Project Report

On

“Information Store Management System”

Submitted by:

Pratik Vijay Manjarekar

(Roll no: 65)

Guided by:

Ms. Preeti Ramtekkar

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING

R.M.D Sinhgad School of Engineering ,Warje, Pune-58

Savitribai Phule Pune University

2024-25

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING

R.M.D Sinhgad School of Engineering ,Warje, Pune-58

Savitribai Phule Pune University

CERTIFICATE

This is to certify that **"Information Store Management System"** embodies the original work done by **Pratik Vijay Manjarekar** during this project submission as a partial fulfillment of the requirement for the Mini Project in subject Database Management System of Third year Computer Engineering students of Savitribai Phule Pune University during the academic year 2024-2025

Date: 21/10/2024

Place: Pune

Project Guide	Head of Department	Principal
(Ms. Preeti Ramtekkar)	(Dr. Vina M. Lomte)	(Dr.V.V. Dixit)

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project guide Ms. Preeti Ramtekkar for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project. We also thank our other staff members who have helped in successful completion of the project.

Name of the students

Pratik Vijay Manjarekar

Contents

1. INTRODUCTION

Project Name: **Information Store Management System**

Platform: Visual Studio code

Programming Language: Python Tkinter

Database: Mysql

The Information Store Management System (ISMS) is a software application designed to manage, store, and retrieve various types of information efficiently. In an era where data is paramount, this system helps organizations streamline their data management processes, ensuring that information is organized and accessible to authorized users.

Objectives :

The application serves as a management system to store, update, delete, and search student information (e.g., roll number, name, class, contact details).

2. Scope of the project

The system will manage various types of information (e.g., documents, records, reports).

It will include data storage, and reporting features.

3. Requirement Analysis

3.1 Non-Functional Requirements:

Non-Functional requirements are the ones that specify criteria that can be used to judge the operation of a system, rather than specific behaviors.

The ones concerned for the project are as:

- **Security**

  Security is the degree of protection against danger, damage, loss and crime. In order to ensure security, a login is maintained which is user name and password secured and hence forth is accessible by only certain trustworthy people from admin.

- **Maintainability**

  Maintainability refers to the ease with which a product can be maintained in order to:

  - Isolate requirements of their cause
  - Meet new requirements
  - Make future management easier or,
  - Cope with a changed environment

- **Performance**

  Good performance is maintained by providing a reliable and high quality service to the customer as customer satisfaction is the top most priority.

- **Testability**

Testability refers the capability of a system to be tested. As this system depicts a real life scenario it can be easily tested, as to how it is able to store customer details.

- **Usability**

  Usability is the ease of use and learns ability. System is made very user friendly using interactive GUI so that it is easy to use, understand and maintain as well.

- **Robustness**

  Robustness is the ability of a system to cope with errors during execution. As the system has been tested for all sorts of invalid and unexpected inputs and exception handling is implemented, hence it is Robust in nature.

- **Accessibility**

  Accessibility can have viewed as the “ability to access” and benefit from some system or entity. System is accessible only to admin for special functions and hence is easily accessible and easy to maintain.

- **Portability**

  It describes that how easy it is to reuse the system. It requires generalized abstraction between the application logic and system interfaces. This system is quite portable as a few minor changes and it can cater to various management systems.

4. System Requirements

H/w and S/w requirement: -

Hardware Requirements

RAM	8 GB or more (16 GB recommended for large organizations)
Hard disk	500 GB hard drive or SSD (1 TB recommended for larger databases)
Processor	Intel Core i3 or higher

Software Requirements

Database	MySQL
Framework	Visual Studio Code
Operating System	WINDOWS

5. ER Diagram

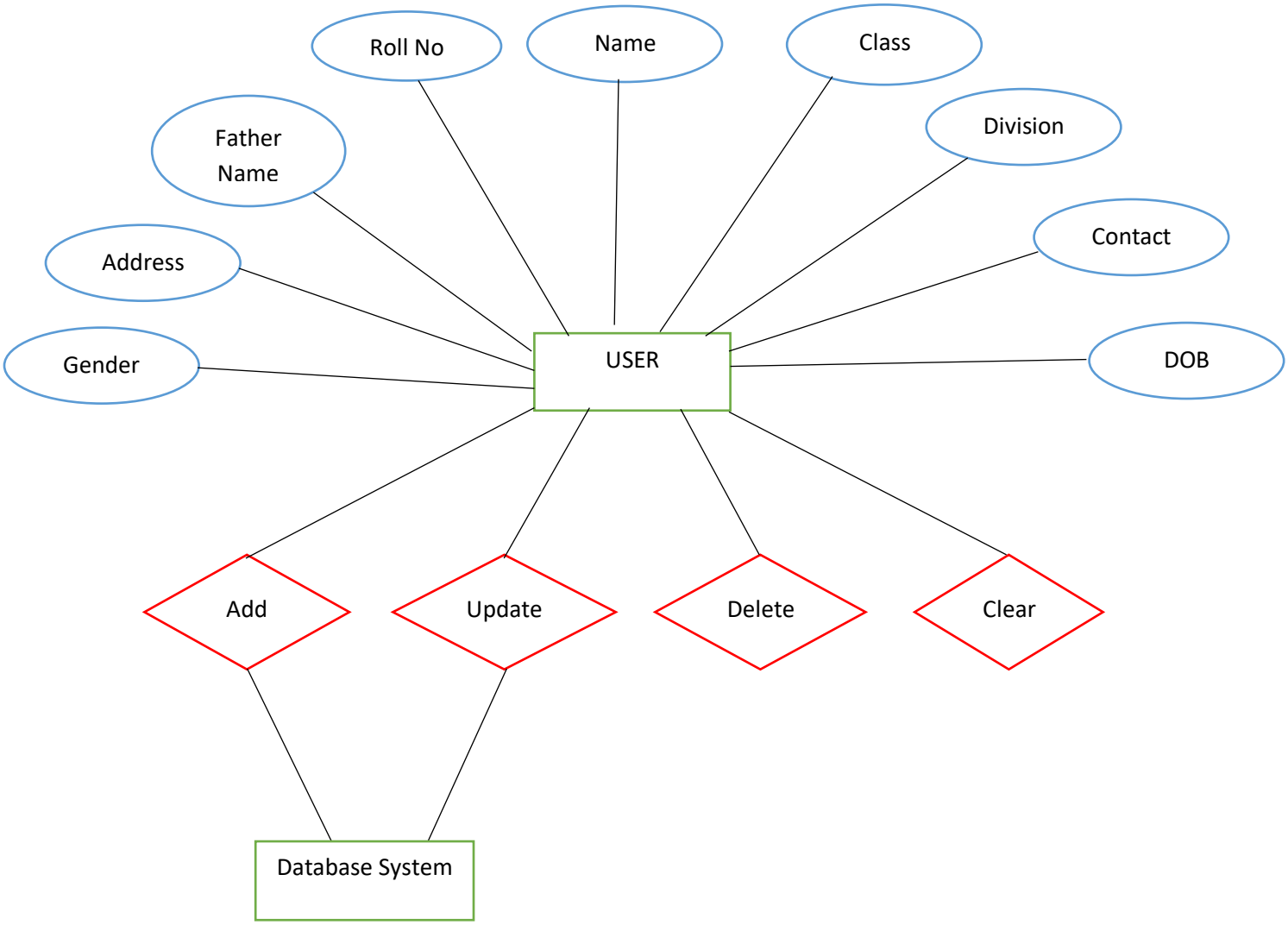

Roll No
Name
Class
Division
Father Name
Contact
Address
Gender
USER
DOB
Add
Update
Delete
Clear
Database System

TABLE STRUCTURES

Field	Type	Null	Key	Default	Extra
rollno	varchar(225)	NO	PRI	NULL	
name	varchar(225)	NO		NULL	
class	varchar(225)	NO		NULL	
division	varchar(225)	NO		NULL	
contact	varchar(225)	NO		NULL	
fathername	varchar(225)	NO		NULL	
address	varchar(225)	NO		NULL	
gender	varchar(225)	NO		NULL	
dob	varchar(225)	NO		NULL	

6. Coding

```
import tkinter as tk

from tkinter import ttk

from tkinter import messagebox

import pymysql


win = tk.Tk()

win.geometry("1350x700")

win.title("Information Store Management System")


label=tk.Label(win, text="Information Store Management
System",font=("Airal",30,"bold"),border=12,relief=tk.GROOVE,bg="lightgray")

label.pack(side=tk.TOP,fill=tk.X)


#================== Frame 1 =========================#


detail_frame = tk.LabelFrame(win,text="Enter
Details",font=("Airal",25),bd=12,relief=tk.GROOVE,bg="lightgray")

detail_frame.place(x=20,y=90,width=420,height=575)


data_frame = tk.LabelFrame(win,bd=12,bg="lightgray",relief=tk.GROOVE)
```

```
data_frame.place(x=475,y=90,width=810,height=575)


#===================Variables=========================#

rollno = tk.StringVar()

name = tk.StringVar()

class_var = tk.StringVar()

division = tk.StringVar()

contact = tk.StringVar()

fathername = tk.StringVar()

address = tk.StringVar()

gender = tk.StringVar()

dob = tk.StringVar()


search_by = tk.StringVar()


#=====================================================#

#=========== Entry ============#


rollno_lbl = tk.Label(detail_frame,text="Roll no ",font=("Airal",15),bg="lightgray")

rollno_lbl.grid(row=0,column=0,padx=2,pady=2)
```

```
rollno_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=rollno)
rollno_ent.grid(row=0,column=1,padx=2,pady=2)

#-------------------------------------------------------------#

name_lbl = tk.Label(detail_frame,text="Name ",font=("Airal",15),bg="lightgray")
name_lbl.grid(row=1,column=0,padx=2,pady=2)

name_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=name)
name_ent.grid(row=1,column=1,padx=2,pady=2)

#-------------------------------------------------------------#

class_lbl = tk.Label(detail_frame,text="Class ",font=("Airal",15),bg="lightgray")
class_lbl.grid(row=2,column=0,padx=2,pady=2)

class_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=class_var)
class_ent.grid(row=2,column=1,padx=2,pady=2)

#-------------------------------------------------------------#
```

```
div_lbl = tk.Label(detail_frame,text="Division ",font=("Airal",15),bg="lightgray")

div_lbl.grid(row=3,column=0,padx=2,pady=2)


div_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=division)

div_ent.grid(row=3,column=1,padx=2,pady=2)


#-------------------------------------------------------------#


contact_lbl = tk.Label(detail_frame,text="Contact ",font=("Airal",15),bg="lightgray")

contact_lbl.grid(row=4,column=0,padx=2,pady=2)


contact_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=contact)

contact_ent.grid(row=4,column=1,padx=2,pady=2)


#-------------------------------------------------------------#


fathername_lbl = tk.Label(detail_frame,text="Father Name
",font=("Airal",15),bg="lightgray")

fathername_lbl.grid(row=5,column=0,padx=2,pady=2)


fathername_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=fathername)
```

```
fathername_ent.grid(row=5,column=1,padx=2,pady=2)

#-------------------------------------------------------------#

address_lbl = tk.Label(detail_frame,text="Address ",font=("Airal",15),bg="lightgray")
address_lbl.grid(row=6,column=0,padx=2,pady=2)

address_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=address)
address_ent.grid(row=6,column=1,padx=2,pady=2)

#-------------------------------------------------------------#

gender_lbl = tk.Label(detail_frame,text="Gender ",font=("Airal",15),bg="lightgray")
gender_lbl.grid(row=7,column=0,padx=2,pady=2)

gender_ent =
ttk.Combobox(detail_frame,font=("Airal",15),state="readonly",textvariable=gender)
gender_ent['values']=("Male","Female","Other")
gender_ent.grid(row=7,column=1,padx=2,pady=2)

#-------------------------------------------------------------#
```

```
dob_lbl = tk.Label(detail_frame,text="DOB ",font=("Airal",15),bg="lightgray")
dob_lbl.grid(row=8,column=0,padx=2,pady=2)

dob_ent = tk.Entry(detail_frame,bd=7,font=("Airal",15),textvariable=dob)
dob_ent.grid(row=8,column=1,padx=2,pady=2)

#========================================#

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

def fetch_data():
    conn = pymysql.connect(host="localhost",user="root",password="",database="sms1")
    curr = conn.cursor()
    curr.execute("SELECT * FROM data")
    rows = curr.fetchall()
    if len(rows)!=0:
        student_table.delete(*student_table.get_children())
        for row in rows:
            student_table.insert('',tk.END,values=row)
        conn.commit()
```

```
        conn.close()


def add_fun():

    if rollno.get()== "" or name.get()== "" or class_var.get()== "" or division.get()== "" or
contact.get()== "" or fathername.get()== "" or address.get()== "" or gender.get()== "" or
dob.get()== "":

        messagebox.showerror("Error!","Please fill all the fields!")

    else:

        conn =
pymysql.connect(host="localhost",user="root",password="",database="sms1")

        curr = conn.cursor()

        curr.execute("INSERT INTO data
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s)",(rollno.get(),name.get(),class_var.get(),divi
sion.get(),contact.get(),fathername.get(),address.get(),gender.get(),dob.get()))

        conn.commit()

        conn.close()


        fetch_data() #---->This will fetch data after adding (means UPDATE)


def get_cursor(event):

    '''This function fetch the data for selected row'''

    cursor_row = student_table.focus()

    content = student_table.item(cursor_row)
```

```
    row = content["values"]
    rollno.set(row[0])
    name.set(row[1])
    class_var.set(row[2])
    division.set(row[3])
    contact.set(row[4])
    fathername.set(row[5])
    address.set(row[6])
    gender.set(row[7])
    dob.set(row[8])


def clear():
    '''This is function will clear the entry boxes'''
    rollno.set("")
    name.set("")
    class_var.set("")
    division.set("")
    contact.set("")
    fathername.set("")
    address.set("")
    gender.set("")
```

```
    dob.set("")



def update_fun():

    '''This function update data according to user'''

    conn = pymysql.connect(host="localhost",user="root",password="",database="sms1")

    curr = conn.cursor()

    curr.execute("UPDATE data SET name=%s, class=%s, division=%s, contact=%s,
fathername=%s, address=%s, gender=%s, dob=%s where rollno
=%s",(name.get(),class_var.get(),division.get(),contact.get(),fathername.get(),address.g
et(),gender.get(),dob.get(),rollno.get()))

    conn.commit()

    conn.close()

    fetch_data()

    clear()


def delete():

    conn = pymysql.connect(host="localhost",user="root",password="",database="sms1")

    curr = conn.cursor()

    curr.execute("DELETE FROM data WHERE rollno = %s", rollno.get())

    conn.commit()

    conn.close()
```

```
    fetch_data() #---->This will fetch data after deleting (means UPDATE)

    clear()


def search_data():

    conn = pymysql.connect(host="localhost",user="root",password="",database="sms1")

    curr = conn.cursor()

    query = ""

    if search_by.get() == "Name":

        query = "SELECT * FROM data WHERE name LIKE
'%{}%'".format(search_ent.get())

    elif search_by.get() == "Roll No":

        query = "SELECT * FROM data WHERE rollno LIKE
'%{}%'".format(search_ent.get())

    elif search_by.get() == "Contact":

        query = "SELECT * FROM data WHERE contact LIKE
'%{}%'".format(search_ent.get())

    elif search_by.get() == "Father's Name":

        query = "SELECT * FROM data WHERE fathername LIKE
'%{}%'".format(search_ent.get())

    elif search_by.get() == "Class":

        query = "SELECT * FROM data WHERE class LIKE
'%{}%'".format(search_ent.get())

    elif search_by.get() == "Division":
```

```
        query = "SELECT * FROM data WHERE division LIKE
'%{}%'".format(search_ent.get())
    elif search_by.get() == "DOB":
        query = "SELECT * FROM data WHERE dob LIKE
'%{}%'".format(search_ent.get())
    curr.execute(query)
    rows = curr.fetchall()
    if len(rows)!=0:
        student_table.delete(*student_table.get_children())
        for row in rows:
            student_table.insert('',tk.END,values=row)
        conn.commit()
        conn.close()
    else:
        messagebox.showerror("Error!","No data found!")
#================ Frame 2 ================#
#================ Button =================#


btn_frame = tk.LabelFrame(detail_frame,bd=10,relief=tk.GROOVE,bg="lightgray")
btn_frame.place(x=25,y=390,width=342,height=120)


#---------------------------------------------#
```

```
add_lbl =
tk.Button(btn_frame,text="Add",bd=7,font=("Airal",13),width=15,bg="lightgray",comman
d=add_fun)

add_lbl.grid(row=0,column=0,padx=2,pady=2)


update_lbl =
tk.Button(btn_frame,text="Update",bd=7,font=("Airal",13),width=15,bg="lightgray",comm
and=update_fun)

update_lbl.grid(row=0,column=1,padx=2,pady=2)


delete_lbl =
tk.Button(btn_frame,text="Delete",bd=7,font=("Airal",13),width=15,bg="lightgray",comm
and=delete)

delete_lbl.grid(row=1,column=0,padx=2,pady=2)


clear_lbl =
tk.Button(btn_frame,text="Clear",bd=7,font=("Airal",13),width=15,bg="lightgray",comma
nd=clear)

clear_lbl.grid(row=1,column=1,padx=2,pady=2)


#================== Frame 3 =====================#

#================== Search =====================#
```

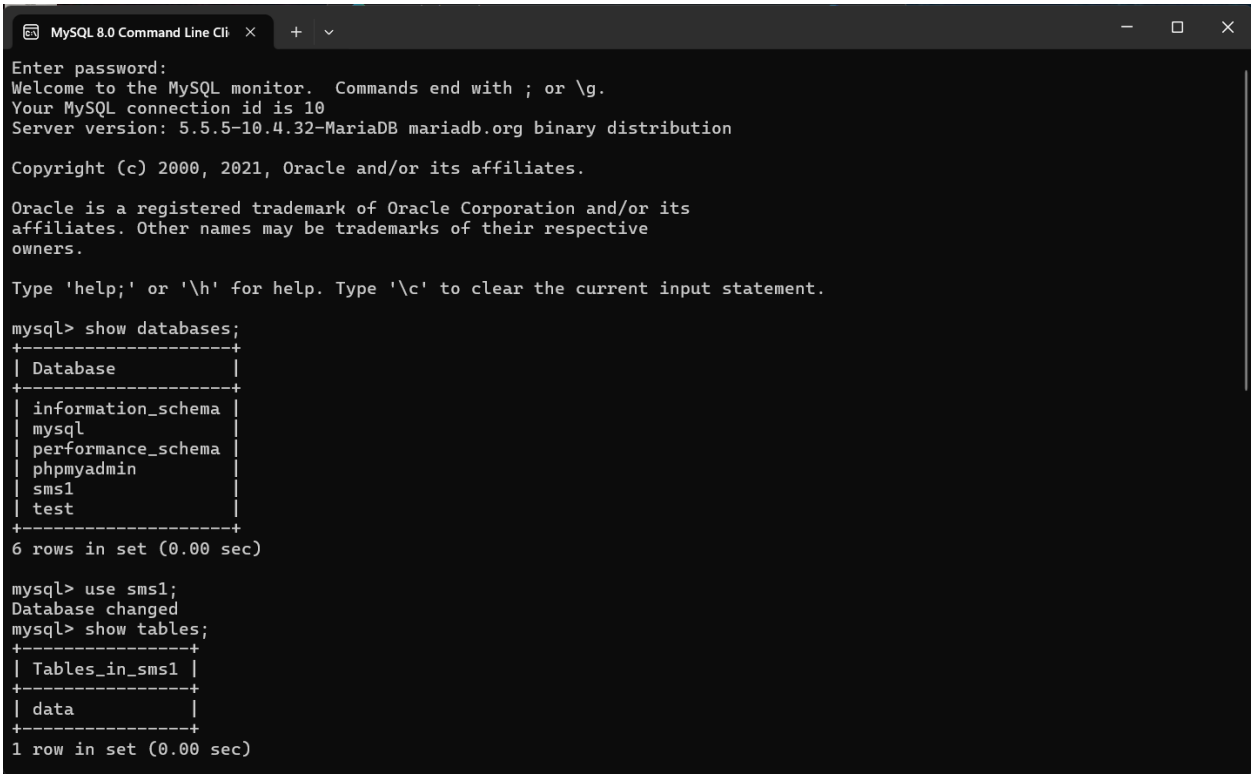
```
search_frame = tk.Frame(data_frame,bd=10,relief=tk.GROOVE,bg="lightgray")

search_frame.pack(side=tk.TOP,fill=tk.X)


#-------------------------------------------#

search_lbl = tk.Label(search_frame,text="Search",font=("Airal",14),bg="lightgray")

search_lbl.grid(row=0,column=0,padx=2,pady=2)


search_in =
ttk.Combobox(search_frame,font=("Airal",14),state="readonly",textvariable=search_by)

search_in['values']=("Name","Roll No","Contact","Father's
Name","Class","Division","DOB")

search_in.grid(row=0,column=1,padx=2,pady=2)


search_ent = tk.Entry(search_frame,bd=7,font=("Airal",15),textvariable=rollno)

search_ent.grid(row=0,column=2,padx=2,pady=2)


search_btn =
tk.Button(search_frame,text="Search",font=("Arial",10),bg="lightgrey",bd=9,width=9,com
mand=search_data)

search_btn.grid(row=0,column=4,padx=2,pady=2)
```

```
showall_btn = tk.Button(search_frame,text="Show
All",font=("Arial",10),bg="lightgrey",bd=9,width=9,command=fetch_data)

showall_btn.grid(row=0,column=5,padx=2,pady=2)


#==================================================#


#===================== DATABASE
Frame==============================#


main_frame = tk.Frame(data_frame,bg="lightgrey",bd=11,relief=tk.GROOVE)

main_frame.pack(fill=tk.BOTH,expand=True)

'''using tree view & tree view comes with ttk'''


y_scroll = tk.Scrollbar(main_frame,orient=tk.VERTICAL)

x_scroll = tk.Scrollbar(main_frame,orient=tk.HORIZONTAL)


student_table = ttk.Treeview(main_frame,columns=("Roll
No.","Name","Class","Division","Contact","Father's
Name","Address","Gender","DOB"),yscrollcommand=y_scroll.set,xscrollcommand=x_sc
roll.set)


y_scroll.config(command=student_table.yview)

x_scroll.config(command=student_table.xview)
```

```
y_scroll.pack(side=tk.RIGHT,fill=tk.Y)

x_scroll.pack(side=tk.BOTTOM,fill=tk.X)


student_table.heading("Roll No.",text="Roll No.")

student_table.heading("Name",text="Name")

student_table.heading("Class",text="Class")

student_table.heading("Division",text="Division")

student_table.heading("Contact",text="Contact")

student_table.heading("Father's Name",text="Father's Name")

student_table.heading("Address",text="Address")

student_table.heading("Gender",text="Gender")

student_table.heading("DOB",text="DOB")


student_table['show']='headings'


student_table.column("Roll No.",width=100)

student_table.column("Name",width=100)

student_table.column("Class",width=100)

student_table.column("Division",width=100)

student_table.column("Contact",width=100)
```

```
student_table.column("Father's Name",width=100)

student_table.column("Address",width=150)

student_table.column("Gender",width=100)

student_table.column("DOB",width=100)


student_table.pack(fill=tk.BOTH,expand=True)


fetch_data()


student_table.bind("<ButtonRelease-1>",get_cursor)


#==================================================#

win.mainloop()
```

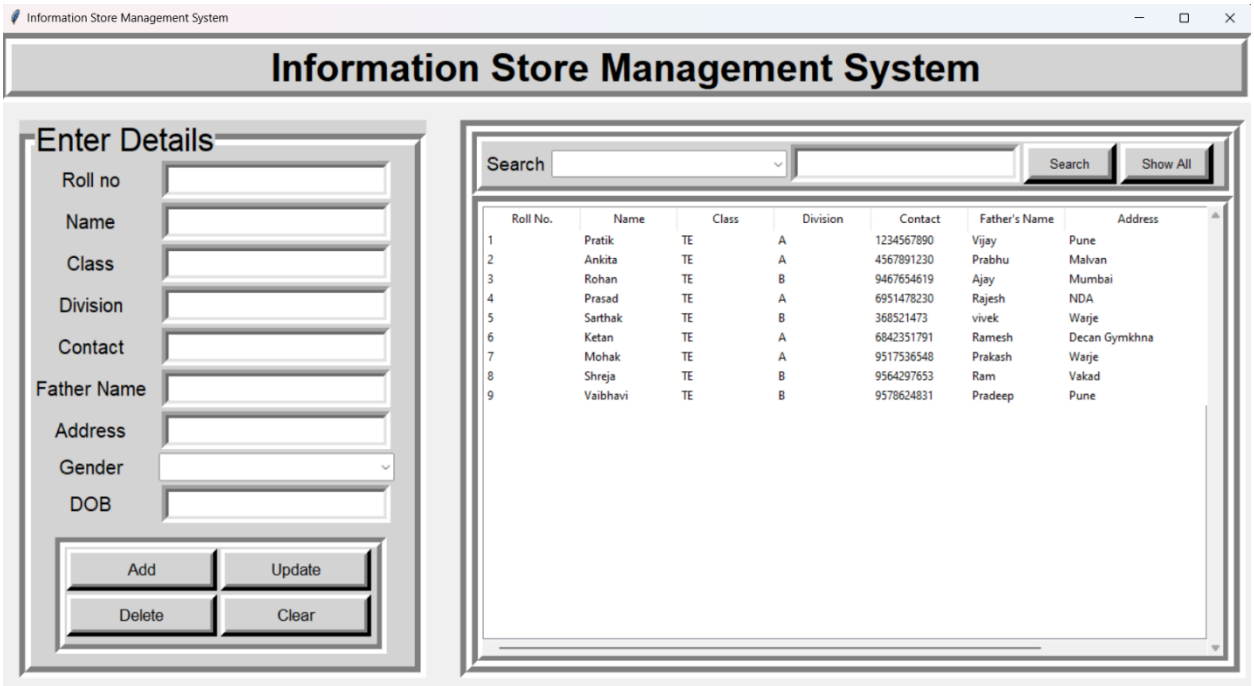
7. Screenshots



```
MySQL 8.0 Command Line Cli

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.5.5-10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sms1               |
| test               |
+--------------------+
6 rows in set (0.00 sec)

mysql> use sms1;
Database changed
mysql> show tables;
+----------------+
| Tables_in_sms1 |
+----------------+
| data           |
+----------------+
1 row in set (0.00 sec)
```

MySQL 8.0 Command Line Cli

```
mysql> desc data;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| rollno     | varchar(225) | NO   | PRI | NULL    |       |
| name       | varchar(225) | NO   |     | NULL    |       |
| class      | varchar(225) | NO   |     | NULL    |       |
| division   | varchar(225) | NO   |     | NULL    |       |
| contact    | varchar(225) | NO   |     | NULL    |       |
| fathername | varchar(225) | NO   |     | NULL    |       |
| address    | varchar(225) | NO   |     | NULL    |       |
| gender     | varchar(225) | NO   |     | NULL    |       |
| dob        | varchar(225) | NO   |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
9 rows in set (0.05 sec)

mysql> select * from data;
+--------+----------+-------+----------+------------+------------+---------------+--------+------------+
| rollno | name     | class | division | contact    | fathername | address       | gender | dob        |
+--------+----------+-------+----------+------------+------------+---------------+--------+------------+
| 1      | Pratik   | TE    | A        | 1234567890 | Vijay      | Pune          | Male   | 27/05/2005 |
| 2      | Ankita   | TE    | A        | 4567891230 | Prabhu     | Malvan        | Female | 15/02/1995 |
| 3      | Rohan    | TE    | B        | 9467654619 | Ajay       | Mumbai        | Male   | 26/06/2000 |
| 4      | Prasad   | TE    | A        | 6951478230 | Rajesh     | NDA           | Male   | 16/07/2003 |
| 5      | Sarthak  | TE    | B        | 368521473  | vivek      | Warje         | Male   | 24/03/2003 |
| 6      | Ketan    | TE    | A        | 6842351791 | Ramesh     | Decan Gymkhna | Male   | 23/08/2002 |
| 7      | Mohak    | TE    | A        | 9517536548 | Prakash    | Warje         | Male   | 4681359725 |
| 8      | Shreja   | TE    | B        | 9564297653 | Ram        | Vakad         | Female | 6/03/2000  |
| 9      | Vaibhavi | TE    | B        | 9578624831 | Pradeep    | Pune          | Female | 4/01/1999  |
+--------+----------+-------+----------+------------+------------+---------------+--------+------------+
9 rows in set (0.00 sec)

mysql>
```



8. Conclusion

The Information Store Management System this application serves as a management system to store, update, delete, and search student information (e.g., roll number, name, class, contact details).

The GUI consists of labeled entry fields for student details, buttons for actions (Add, Update, Delete, Clear), and a search functionality.

It connects to a MySQL database using pymysql to perform CRUD (Create, Read, Update, Delete) operations on student data.

A table (using ttk.Treeview) displays the student records retrieved from the database, allowing users to view all records or search for specific entries.

This Information Store Management System is a robust application that integrates a user-friendly interface with a backend database for managing student information efficiently. It allows for seamless data manipulation, ensuring that users can easily maintain and retrieve records as needed.

9. References

1. Silberschatz A. , Korth H. , Sudarshan S , Database System Concepts , McGraw Hill Publication, ISBN-0-07-120413-X, 6th Edition.

2. www.mysqltutorial.org