



MURUGAPPA POLYTECHNIC COLLEGE

Sathyamurthy Nagar, Chennai-600 062.

(An Academically Autonomous Institution)



DEPARTMENT OF COMPUTER ENGINEERING

MINI PROJECT REPORT

ON

STUDENT MANAGEMENT SYSTEM

Submitted by

NAME

REGISTER NO

GANESHAN M

2113213

STAFF INCHARGE

HOD

ABSTRACT

Schools and Universities are the foundation of knowledge and an educational body on which students rely upon. Therefore, they need to maintain a proper database of its students to keep all the updated records and easily share information with students. Most schools and Universities count on an advanced software tool known as 'Student Information System (SIS)' to keep all their student records. Over the recent years, the performance and efficiency of the education industry have been enhanced by using the Student Management System. This tool has productively taken over the workload of the admin department with its well-organized, easy, and reliable.

Student Management System (SMS) is a solution tool that is designed to track, maintain and manage all the data generated by a School, including the grades of a student, their attendance, their interpersonal activities records, etc.,"

The Student Management System software is created to help manage the student's admissions activities, starting from initial communication to course enrolment.

HARDWARE REQUIREMENTS :

- Processor : RYZEN 3
- Hard Disk : 500 GB
- RAM : 8 GB

SOFTWARE REQUIREMENTS :

- Operating System : Windows 11
- Languages : PYTHON
- Software : VISUAL STUDIO CODE & PYTHON
- Extension : RUNNER & PYTHON DEBUGGING

Program

Main.py

```
from db import Database
from tkinter import ttk
from tkinter import messagebox
from tkinter import *
import tkinter as tk

db=Database("mpc")
root = Tk()
root.title("Student Management System")
root.geometry("1920x1080+0+0")
root.config(bg="#2c3e50")
root.state("zoomed")

Rollno=StringVar()
Name=StringVar()
DOB=StringVar()
Gender=StringVar()
Mark10th=StringVar()
JoinDate=StringVar()
dept=StringVar()
contact=StringVar()
address=StringVar()

# Entries Frame
entries_frame = Frame(root, bg="#4F3F84")
entries_frame.pack(side=TOP, fill=X)
title = Label(entries_frame, text="Student Management System", font=("Calibri", 18, "bold"), bg="#4F3F84", fg="white")
title.grid(row=0, columnspan=2, padx=10, pady=20, sticky="w")
lblrollno = Label(entries_frame, text="Rollno", font=("Calibri", 16), bg="#4F3F84", fg="white")
lblrollno.grid(row=1, column=0, padx=10, pady=10, sticky="w")
txtrollno = Entry(entries_frame, textvariable=Rollno, font=("Calibri", 16), width=30)
txtrollno.grid(row=1, column=1, padx=10, pady=10, sticky="w")
lblname = Label(entries_frame, text="Name", font=("Calibri", 16), bg="#4F3F84", fg="white")
lblname.grid(row=1, column=2, padx=10, pady=10, sticky="w")
txtname = Entry(entries_frame, textvariable=Name, font=("Calibri", 16), width=30)
txtname.grid(row=1, column=3, padx=10, pady=10, sticky="w")
lbldob = Label(entries_frame, text="DOB", font=("Calibri", 16), bg="#4F3F84", fg="white")
lbldob.grid(row=1, column=4, padx=10, pady=10, sticky="w")
txtdob = Entry(entries_frame, textvariable=DOB, font=("Calibri", 16), width=30)
txtdob.grid(row=1, column=5, padx=10, pady=10, sticky="w")
lblGender = Label(entries_frame, text="Gender", font=("Calibri", 16), bg="#4F3F84", fg="white")
lblGender.grid(row=2, column=0, padx=10, pady=10, sticky="w")
comboGender = ttk.Combobox(entries_frame, font=("Calibri", 16), width=28, textvariable=Gender, state="readonly")
comboGender['values'] = ("Male", "Female")
comboGender.grid(row=2, column=1, padx=10, sticky="w")
lblMark_10th = Label(entries_frame, text="Mark10th", font=("Calibri", 16), bg="#4F3F84", fg="white")
lblMark_10th.grid(row=2, column=2, padx=10, pady=10, sticky="w")
txtMark_10th = Entry(entries_frame, textvariable=Mark10th, font=("Calibri", 16), width=30)
```

```

txtMark_10th.grid(row=2, column=3, padx=10, pady=10, sticky="w")
lblJoinDate = Label(entries_frame, text="JoinDate", font=("Calibri", 16), bg="#4F3F84", fg="white")
lblJoinDate.grid(row=2, column=4, padx=10, pady=10, sticky="w")
txtJoinDate= Entry(entries_frame, textvariable=JoinDate, font=("Calibri", 16), width=30)
txtJoinDate.grid(row=2, column=5, padx=10, pady=10, sticky="w")
lbldept = Label(entries_frame, text="Department", font=("Calibri", 16), bg="#4F3F84", fg="white")
lbldept.grid(row=3, column=0, padx=10, pady=10, sticky="w")
txtdept = Entry(entries_frame, textvariable=dept, font=("Calibri", 16), width=30)
txtdept.grid(row=3, column=1, padx=10, pady=10, sticky="w")
lblcontact = Label(entries_frame, text="Contact", font=("Calibri", 16), bg="#4F3F84", fg="white")
lblcontact.grid(row=3, column=2, padx=10, pady=10, sticky="w")
txtcontact= Entry(entries_frame, textvariable=contact, font=("Calibri", 16), width=30)
txtcontact.grid(row=3, column=3, padx=10, pady=10, sticky="w")
lbladdress = Label(entries_frame, text="Address", font=("Calibri", 16), bg="#4F3F84", fg="white")
lbladdress.grid(row=3, column=4, padx=10, pady=10, sticky="w")
txtaddress= Entry(entries_frame, textvariable=address, font=("Calibri", 16), width=30)
txtaddress.grid(row=3, column=5, padx=10, pady=10, sticky="w")

def getdata(e):
    selected_row=tv.focus()
    data = tv.item(selected_row)
    global row
    row =data["values"]
    Rollno.set(row[0])
    Name.set(row[1])
    DOB.set(row[2])
    Gender.set(row[3])
    Mark10th.set(row[4])
    JoinDate.set(row[5])
    dept.set(row[6])
    contact.set(row[7])
    address.set(row[8])

def displayAll():
    tv.delete(*tv.get_children())
    for row in db.fetch():
        tv.insert("", END, values=row)

def add_student():
    if txtrollno.get() == "" or txtname.get() == "" or txtdob.get() == "" or comboGender.get() == "" or
    txtMark_10th.get() == "" or txtJoinDate.get() == "" or txtdept.get() == "" or txtcontact.get() == "" or
    txtaddress.get() == "":
        messagebox.showerror("Error in Input", "Please Fill All the Details")
    return

db.insert(txtrollno.get(),txtname.get(),txtdob.get(),comboGender.get(),txtMark_10th.get(),txtJoinDate.get(),t
xtdept.get(),txtcontact.get(),txtaddress.get())
messagebox.showi
nfo("Success", "Record Inserted")
clearAll()
displayAll()

def update_student():
    if txtrollno.get() == "" or txtname.get() == "" or txtdob.get() == "" or comboGender.get() == "" or
    txtMark_10th.get() == "" or txtJoinDate.get() == "" or txtdept.get() == "" or txtcontact.get() == "" or
    txtaddress.get() == "":
        messagebox.showerror("Error in Input", "Please Fill All the Details")
    return

db.update(txtrollno.get(),txtname.get(),txtdob.get(),comboGender.get(),txtMark_10th.get(),txtJoinDate.get(),
txtdept.get(),txtcontact.get(),txtaddress.get())
messagebox.showinfo("Success", "Record Update")
clearAll()

```

```

displayAll()

def delete_student():
    db.remove(row[0])
    clearAll()
    displayAll()

def clearAll():
    Rollno.set("")
    Name.set("")
    DOB.set("")
    Gender.set("")
    Mark10th.set("")
    JoinDate.set("")
    dept.set("")
    contact.set("")
    address.set("")

btn_frame = Frame(entries_frame, bg="#4F3F84")
btn_frame.grid(row=6, column=0, columnspan=4, padx=10, pady=10, sticky="w")
btnAdd = Button(btn_frame, command=add_student, text="Add Details", width=15, font=("Calibri", 16,
"bold"), fg="white",
                bg="#16a085", bd=0).grid(row=0, column=0)
btnEdit = Button(btn_frame, command=update_student, text="Update Details", width=15, font=("Calibri",
16, "bold"),
                fg="white", bg="#2980b9",
                bd=0).grid(row=0, column=1, padx=10)
btnDelete = Button(btn_frame, command=delete_student, text="Delete Details", width=15, font=("Calibri",
16, "bold"),
                fg="white", bg="#c0392b",
                bd=0).grid(row=0, column=2, padx=10)
btnClear = Button(btn_frame, command=clearAll, text="Clear Details", width=15, font=("Calibri", 16, "bold"),
fg="white",
                bg="#f39c12",
                bd=0).grid(row=0, column=3, padx=10)

tree_frame = Frame(root, bg="#ecf0f1")
tree_frame.place(x=0, y=294, width=1550, height=520)

style = ttk.Style()
style.configure("mystyle.Treeview", font=('Calibri', 18),
                rowheight=50) # Modify the font of the body
style.configure("mystyle.Treeview.Heading", font=('Calibri', 18)) # Modify the font of the headings

# Create the Treeview widget
tv = ttk.Treeview(tree_frame, columns=(1, 2, 3, 4, 5, 6, 7, 8, 9), style="mystyle.Treeview")

# Define headings and set column widths
tv.heading(1, text="Rollno", anchor="center")
tv.column(1, width=100, anchor="center")
tv.heading(2, text="Name", anchor="center")
tv.column(2, width=200, anchor="center") # Adjusted width
tv.heading(3, text="DOB", anchor="center")
tv.column(3, width=150, anchor="center") # Adjusted width
tv.heading(4, text="Gender", anchor="center")
tv.column(4, width=100, anchor="center") # Adjusted width
tv.heading(5, text="Mark10th", anchor="center")
tv.column(5, width=100, anchor="center") # Adjusted width
tv.heading(6, text="JoinDate", anchor="center")
tv.column(6, width=150, anchor="center") # Adjusted width
tv.heading(7, text="Department", anchor="center")
tv.column(7, width=150, anchor="center") # Adjusted width
tv.heading(8, text="Contact", anchor="center")

```

```

tv.column(8, width=150 ,anchor="center") # Adjusted width
tv.heading(9, text="Address" ,anchor="center")
tv.column(9, width=250 ,anchor="center") # Adjusted width

tv['show'] = 'headings'
tv.bind("<ButtonRelease-1>",getdata)

tv.pack(fill=tk.BOTH, expand=True) # Adjusted the fill and expand options
displayAll()
root.mainloop()

```

db.py

```

import mysql.connector

class Database:
    def __init__(self, db):
        try:
            self.con = mysql.connector.connect(
                host="localhost",
                user="root",
                password="root",
                database=db)
            self.cur = self.con.cursor()
            sql = """
            CREATE TABLE IF NOT EXISTS student(
                rollno int Primary Key,
                name varchar(30),
                DOB varchar(30),
                gender varchar(30),
                mark_10th varchar(30),
                join_date varchar(30),
                dept varchar(30),
                contact varchar(10),
                address varchar(30)
            )
            """
            self.cur.execute(sql)
            self.con.commit()

            if self.con.is_connected():
                print("Successfully connected...")
        except Exception as e:
            print("Connection error:", e)

    def insert(self, rollno, name, DOB, gender, mark_10th, join_date, dept, contact, address):
        try:
            sql = "INSERT INTO student VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
            values = (rollno, name, DOB, gender, mark_10th, join_date, dept, contact, address)
            self.cur.execute(sql, values)
            self.con.commit()
            print("Data inserted successfully.")
        except Exception as e:
            print("Insert error:", e)

    def fetch(self):
        self.cur.execute("SELECT * from student")
        rows = self.cur.fetchall()
        return rows

    def remove(self, rollno):
        try:

```

```

sql = "DELETE FROM student WHERE rollno = %s"
self.cur.execute(sql, (rollno,))
self.con.commit()
print("Record with rollno =", rollno, "removed successfully.")
except Exception as e:
    print("Remove error:", e)

def update(self, rollno, name, DOB, gender, mark_10th, join_date, dept, contact, address):
    try:
        sql = "UPDATE student SET name=%s, DOB=%s, gender=%s, mark_10th=%s, join_date=%s,
dept=%s, contact=%s, address=%s WHERE rollno=%s"
        values = (name, DOB, gender, mark_10th, join_date, dept, contact, address, rollno)
        self.cur.execute(sql, values)
        self.con.commit()
        print("Record with rollno =", rollno, "updated successfully.")
    except Exception as e:
        print("Update error:", e)

```

OUTPUT SCREENSHOT

Student Management System

Rollno Name DOB

Gender Mark10th JoinDate

Department Contact Address

Rollno	Name	DOB	Gender	Mark10th	JoinDate	Department	Contact	Address
221061	K.ARUN	18-09-2006	Male	450	12-2-2022	CSE	4598726148	ANNA NAGAR
221108	A.U.SURYA	4-3-2005	Male	499	12-2-2022	CSE	148652478	AVADI
221109	S.SYED ASLAM	3-9-2005	Male	500	12-2-2022	CSE	178524944	AVADI

CONCLUSION:

In conclusion, using a student management system in school can benefit students and staff significantly. Therefore, by streamlining administrative tasks, reducing errors, and centralizing student data, a student management system can increase the efficiency of school operations and save time.

