**Report of mini project**

**Title of project:** **Student Management System.**

**Name of student: Prerna karn bhosale.**

**Roll no:13124.**

**Aim:** **To design and develop a database system to manage student details such as name, roll number, department, marks, and attendance.**

**Use Case: This system is used by school/college administrators to store, view, and manage student records efficiently.**

**Mysql tables:**

**Microsoft Windows [Version 10.0.26200.6725]**

**(c) Microsoft Corporation. All rights reserved.**

**C:\Program Files\MySQL\MySQL Server 9.3\bin>mysql -h localhost -u root -p**

**Enter password: \*\*\*\***

**Welcome to the MySQL monitor. Commands end with ; or \g.**

**Your MySQL connection id is 9**

**Server version: 9.3.0 MySQL Community Server - GPL**

**Copyright (c) 2000, 2025, 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> CREATE DATABASE student\_db;**

**Query OK, 1 row affected (1.339 sec)**

**mysql> USE student\_db;**

**Database changed**

**mysql> CREATE TABLE students(student\_id INT AUTO\_INCREMENT PRIMARY KEY,name VARCHAR(100),roll\_no VARCHAR(20),department VARCHAR(50));**

**Query OK, 0 rows affected (1.630 sec)**

**mysql> CREATE TABLE courses (course\_id INT AUTO\_INCREMENT PRIMARY KEY,course\_name VARCHAR(100),department VARCHAR(50));**

**Query OK, 0 rows affected (0.467 sec)**

**mysql> CREATE TABLE marks (mark\_id INT AUTO\_INCREMENT PRIMARY KEY,student\_id INT,course\_id INT,marks\_obtained INT,FOREIGN KEY (student\_id) REFERENCES students(student\_id),FOREIGN KEY (course\_id) REFERENCES courses(course\_id));**

**Query OK, 0 rows affected (1.053 sec)**

**mysql> CREATE TABLE attendance (att\_id INT AUTO\_INCREMENT PRIMARY KEY,student\_id INT,total\_classes INT,attended INT,FOREIGN KEY (student\_id) REFERENCES students(student\_id));**

**Query OK, 0 rows affected (0.792 sec)**

**Source code in python:**

**import mysql.connector**

**# Connect to MySQL**

**conn = mysql.connector.connect(**

**host="localhost",**

**user="root",**

**password="root",**

**database="student\_db"**

**)**

**# Create a cursor object to execute SQL queries**

**cursor = conn.cursor()**

**# Step 2: Functions for each operation**

**def add\_student():**

**name = input("Enter name: ")**

**roll\_no = input("Enter roll number: ")**

**department = input("Enter department: ")**

**cursor.execute(**

**"INSERT INTO students (name, roll\_no, department) VALUES (%s, %s, %s)",**

**(name, roll\_no, department)**

**)**

**conn.commit()**

**print("✅ Student added successfully!\n")**

**def view\_students():**

**cursor.execute("SELECT \* FROM students")**

**records = cursor.fetchall()**

**print("\n--- Student List ---")**

**for row in records:**

**print(f"ID: {row[0]}, Name: {row[1]}, Roll No: {row[2]}, Department: {row[3]}")**

**print()**

**def update\_student():**

**sid = input("Enter student ID to update: ")**

**name = input("Enter new name: ")**

**dept = input("Enter new department: ")**

**cursor.execute(**

**"UPDATE students SET name=%s, department=%s WHERE student\_id=%s",**

**(name, dept, sid)**

**)**

**conn.commit()**

**print("✅ Record updated successfully!\n")**

**def delete\_student():**

**sid = input("Enter student ID to delete: ")**

**cursor.execute("DELETE FROM students WHERE student\_id=%s", (sid,))**

**conn.commit()**

**print("🗑️ Student deleted successfully!\n")**

**def add\_marks():**

**sid = input("Enter student ID: ")**

**cid = input("Enter course ID: ")**

**marks = input("Enter marks obtained: ")**

**cursor.execute(**

**"INSERT INTO marks (student\_id, course\_id, marks\_obtained) VALUES (%s, %s, %s)",**

**(sid, cid, marks)**

**)**

**conn.commit()**

**print("✅ Marks added successfully!\n")**

**def view\_marks():**

**cursor.execute(**

**"SELECT s.name, c.course\_name, m.marks\_obtained "**

**"FROM marks m "**

**"JOIN students s ON m.student\_id = s.student\_id "**

**"JOIN courses c ON m.course\_id = c.course\_id"**

**)**

**records = cursor.fetchall()**

**print("\n--- Marks Report ---")**

**for row in records:**

**print(f"Student: {row[0]}, Course: {row[1]}, Marks: {row[2]}")**

**print()**

**def add\_attendance():**

**sid = input("Enter student ID: ")**

**total = int(input("Enter total classes: "))**

**attended = int(input("Enter attended classes: "))**

**cursor.execute(**

**"INSERT INTO attendance (student\_id, total\_classes, attended) VALUES (%s, %s, %s)",**

**(sid, total, attended)**

**)**

**conn.commit()**

**print("✅ Attendance added successfully!\n")**

**def view\_attendance():**

**cursor.execute(**

**"SELECT s.name, a.total\_classes, a.attended "**

**"FROM attendance a "**

**"JOIN students s ON a.student\_id = s.student\_id"**

**)**

**records = cursor.fetchall()**

**print("\n--- Attendance Report ---")**

**for row in records:**

**percent = (row[2] / row[1]) \* 100 if row[1] > 0 else 0**

**print(f"Student: {row[0]}, Attendance: {percent:.2f}%")**

**print()**

**# Step 3: Menu-driven program**

**while True:**

**print("""**

**===== Student Management System =====**

**1. Add Student**

**2. View Students**

**3. Update Student**

**4. Delete Student**

**5. Add Marks**

**6. View Marks**

**7. Add Attendance**

**8. View Attendance**

**9. Exit**

**""")**

**choice = input("Enter your choice: ")**

**if choice == '1':**

**add\_student()**

**elif choice == '2':**

**view\_students()**

**elif choice == '3':**

**update\_student()**

**elif choice == '4':**

**delete\_student()**

**elif choice == '5':**

**add\_marks()**

**elif choice == '6':**

**view\_marks()**

**elif choice == '7':**

**add\_attendance()**

**elif choice == '8':**

**view\_attendance()**

**elif choice == '9':**

**print("👋 Exiting...")**

**break**

**else:**

**print("❌ Invalid choice! Try again.")**

**output:**

**PS C:\Users\HP> python "C:\Users\HP\OneDrive\Desktop\student\_management.py"**

**===== Student Management System =====**

**1. Add Student**

**2. View Students**

**3. Update Student**

**4. Delete Student**

**5. Add Marks**

**6. View Marks**

**7. Add Attendance**

**8. View Attendance**

**9. Exit**

**Enter your choice: 1**

**Enter name: prerna**

**Enter roll number: 13124**

**Enter department: computer**

**✅ Student added successfully!**

**===== Student Management System =====**

**1. Add Student**

**2. View Students**

**3. Update Student**

**4. Delete Student**

**5. Add Marks**

**6. View Marks**

**7. Add Attendance**

**8. View Attendance**

**9. Exit**

**Enter your choice: 9**

**👋 Exiting...**

**Front end code:**

import tkinter as tk

from tkinter import ttk, messagebox

import mysql.connector

# ===== MySQL Connection =====

conn = mysql.connector.connect(

    host="localhost",

    user="root",

    password="root",  # Change this if needed

    database="student\_db"

)

cursor = conn.cursor()

# ====== Functions ======

def add\_student():

    name = name\_entry.get()

    roll = roll\_entry.get()

    dept = dept\_entry.get()

    if not name or not roll or not dept:

        messagebox.showerror("Error", "All fields are required!")

        return

    cursor.execute(

        "INSERT INTO students (name, roll\_no, department) VALUES (%s, %s, %s)",

        (name, roll, dept)

    )

    conn.commit()

    messagebox.showinfo("Success", "Student added successfully!")

    clear\_student\_fields()

    view\_students()

def view\_students():

    for row in student\_tree.get\_children():

        student\_tree.delete(row)

    cursor.execute("SELECT \* FROM students")

    for r in cursor.fetchall():

        student\_tree.insert("", tk.END, values=r)

def update\_student():

    selected = student\_tree.selection()

    if not selected:

        messagebox.showerror("Error", "Select a student to update!")

        return

    sid = student\_tree.item(selected)["values"][0]

    name = name\_entry.get()

    dept = dept\_entry.get()

    if not name or not dept:

        messagebox.showerror("Error", "Enter new name and department!")

        return

    cursor.execute(

        "UPDATE students SET name=%s, department=%s WHERE student\_id=%s",

        (name, dept, sid)

    )

    conn.commit()

    messagebox.showinfo("Success", "Student updated successfully!")

    view\_students()

def delete\_student():

    selected = student\_tree.selection()

    if not selected:

        messagebox.showerror("Error", "Select a student to delete!")

        return

    sid = student\_tree.item(selected)["values"][0]

    cursor.execute("DELETE FROM students WHERE student\_id=%s", (sid,))

    conn.commit()

    messagebox.showinfo("Success", "Student deleted successfully!")

    view\_students()

def add\_marks():

    sid = marks\_student\_id.get()

    cid = course\_id.get()

    marks = marks\_obtained.get()

    if not sid or not cid or not marks:

        messagebox.showerror("Error", "All fields are required!")

        return

    cursor.execute(

        "INSERT INTO marks (student\_id, course\_id, marks\_obtained) VALUES (%s, %s, %s)",

        (sid, cid, marks)

    )

    conn.commit()

    messagebox.showinfo("Success", "Marks added successfully!")

    view\_marks()

def view\_marks():

    for row in marks\_tree.get\_children():

        marks\_tree.delete(row)

    cursor.execute(

        "SELECT m.mark\_id, s.name, m.course\_id, m.marks\_obtained "

        "FROM marks m "

        "JOIN students s ON m.student\_id = s.student\_id"

    )

    for r in cursor.fetchall():

        marks\_tree.insert("", tk.END, values=(r[0], r[1], r[2], r[3]))

def add\_attendance():

    sid = att\_student\_id.get()

    total = total\_classes.get()

    attended = attended\_classes.get()

    if not sid or not total or not attended:

        messagebox.showerror("Error", "All fields are required!")

        return

    cursor.execute(

        "INSERT INTO attendance (student\_id, total\_classes, attended) VALUES (%s, %s, %s)",

        (sid, total, attended)

    )

    conn.commit()

    messagebox.showinfo("Success", "Attendance added successfully!")

    view\_attendance()

def view\_attendance():

    for row in att\_tree.get\_children():

        att\_tree.delete(row)

    cursor.execute(

        "SELECT s.name, a.total\_classes, a.attended "

        "FROM attendance a "

        "JOIN students s ON a.student\_id = s.student\_id"

    )

    for r in cursor.fetchall():

        percent = (r[2] / r[1]) \* 100 if r[1] > 0 else 0

        att\_tree.insert("", tk.END, values=(r[0], r[1], r[2], f"{percent:.2f}%"))

def clear\_student\_fields():

    name\_entry.delete(0, tk.END)

    roll\_entry.delete(0, tk.END)

    dept\_entry.delete(0, tk.END)

def exit\_app():

    conn.close()

    root.destroy()

# ===== Tkinter GUI =====

root = tk.Tk()

root.title("Student Management System")

root.geometry("950x700")

notebook = ttk.Notebook(root)

notebook.pack(fill="both", expand=True)

# --- TAB 1: Student Management ---

student\_tab = ttk.Frame(notebook)

notebook.add(student\_tab, text="Students")

tk.Label(student\_tab, text="Name").grid(row=0, column=0, padx=5, pady=5)

tk.Label(student\_tab, text="Roll No").grid(row=1, column=0, padx=5, pady=5)

tk.Label(student\_tab, text="Department").grid(row=2, column=0, padx=5, pady=5)

name\_entry = tk.Entry(student\_tab)

roll\_entry = tk.Entry(student\_tab)

dept\_entry = tk.Entry(student\_tab)

name\_entry.grid(row=0, column=1, padx=5, pady=5)

roll\_entry.grid(row=1, column=1, padx=5, pady=5)

dept\_entry.grid(row=2, column=1, padx=5, pady=5)

tk.Button(student\_tab, text="Add Student", command=add\_student).grid(row=3, column=0, pady=5)

tk.Button(student\_tab, text="Update Student", command=update\_student).grid(row=3, column=1, pady=5)

tk.Button(student\_tab, text="Delete Student", command=delete\_student).grid(row=3, column=2, pady=5)

tk.Button(student\_tab, text="View Students", command=view\_students).grid(row=3, column=3, pady=5)

columns = ("ID", "Name", "Roll No", "Department")

student\_tree = ttk.Treeview(student\_tab, columns=columns, show="headings")

for col in columns:

    student\_tree.heading(col, text=col)

student\_tree.grid(row=4, column=0, columnspan=4, padx=5, pady=5)

# --- TAB 2: Marks ---

marks\_tab = ttk.Frame(notebook)

notebook.add(marks\_tab, text="Marks")

tk.Label(marks\_tab, text="Student ID").grid(row=0, column=0, padx=5, pady=5)

tk.Label(marks\_tab, text="Course ID").grid(row=1, column=0, padx=5, pady=5)

tk.Label(marks\_tab, text="Marks Obtained").grid(row=2, column=0, padx=5, pady=5)

marks\_student\_id = tk.Entry(marks\_tab)

course\_id = tk.Entry(marks\_tab)

marks\_obtained = tk.Entry(marks\_tab)

marks\_student\_id.grid(row=0, column=1, padx=5, pady=5)

course\_id.grid(row=1, column=1, padx=5, pady=5)

marks\_obtained.grid(row=2, column=1, padx=5, pady=5)

tk.Button(marks\_tab, text="Add Marks", command=add\_marks).grid(row=3, column=0, pady=5)

tk.Button(marks\_tab, text="View Marks", command=view\_marks).grid(row=3, column=1, pady=5)

marks\_tree = ttk.Treeview(marks\_tab, columns=("ID", "Name", "Course ID", "Marks"), show="headings")

for col in ("ID", "Name", "Course ID", "Marks"):

    marks\_tree.heading(col, text=col)

marks\_tree.grid(row=4, column=0, columnspan=3, padx=5, pady=5)

# --- TAB 3: Attendance ---

attendance\_tab = ttk.Frame(notebook)

notebook.add(attendance\_tab, text="Attendance")

tk.Label(attendance\_tab, text="Student ID").grid(row=0, column=0, padx=5, pady=5)

tk.Label(attendance\_tab, text="Total Classes").grid(row=1, column=0, padx=5, pady=5)

tk.Label(attendance\_tab, text="Attended").grid(row=2, column=0, padx=5, pady=5)

att\_student\_id = tk.Entry(attendance\_tab)

total\_classes = tk.Entry(attendance\_tab)

attended\_classes = tk.Entry(attendance\_tab)

att\_student\_id.grid(row=0, column=1, padx=5, pady=5)

total\_classes.grid(row=1, column=1, padx=5, pady=5)

attended\_classes.grid(row=2, column=1, padx=5, pady=5)

tk.Button(attendance\_tab, text="Add Attendance", command=add\_attendance).grid(row=3, column=0, pady=5)

tk.Button(attendance\_tab, text="View Attendance", command=view\_attendance).grid(row=3, column=1, pady=5)

att\_tree = ttk.Treeview(attendance\_tab, columns=("Name", "Total", "Attended", "Percent"), show="headings")

for col in ("Name", "Total", "Attended", "Percent"):

    att\_tree.heading(col, text=col)

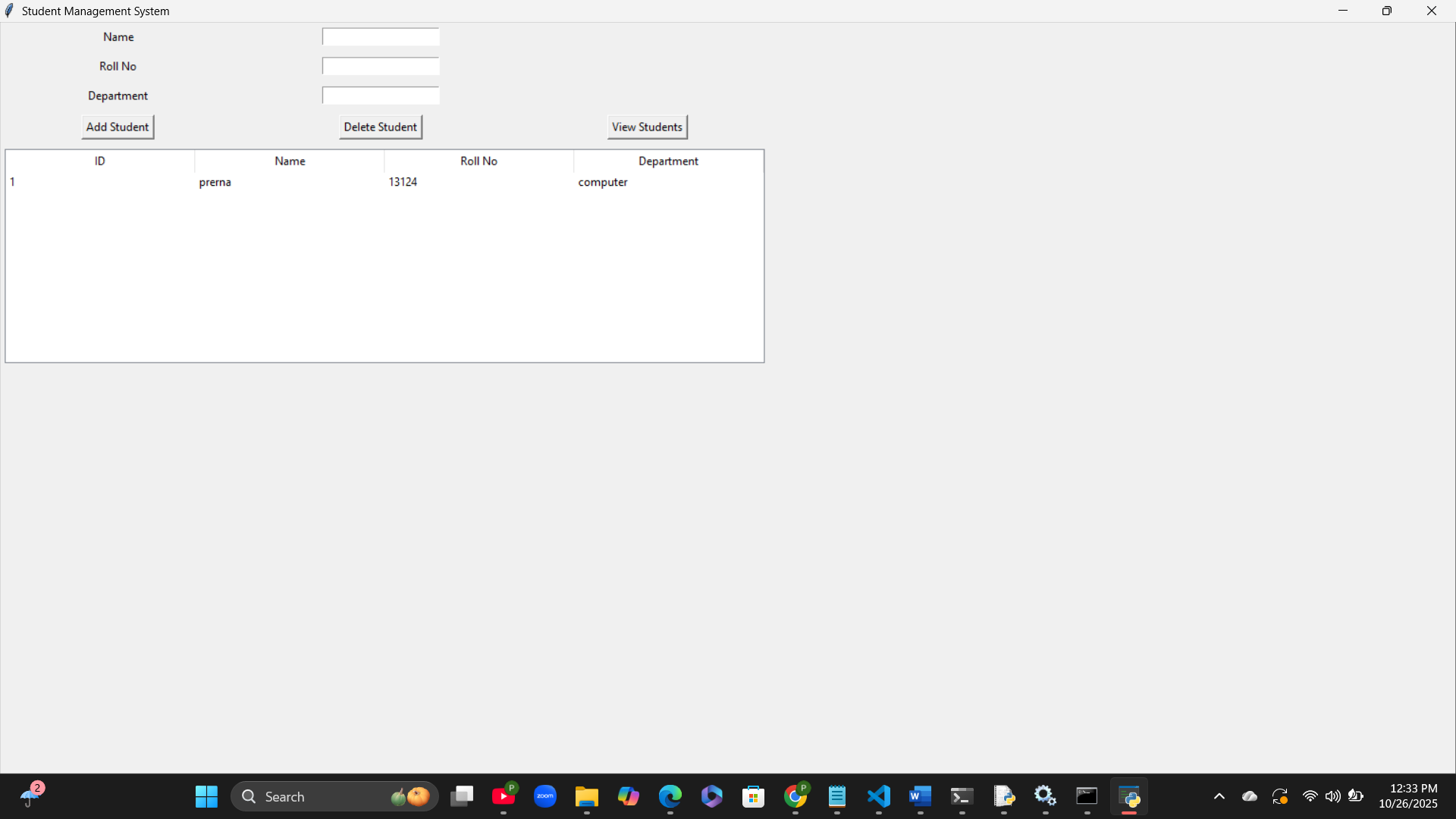
att\_tree.grid(row=4, column=0, columnspan=3, padx=5, pady=5)

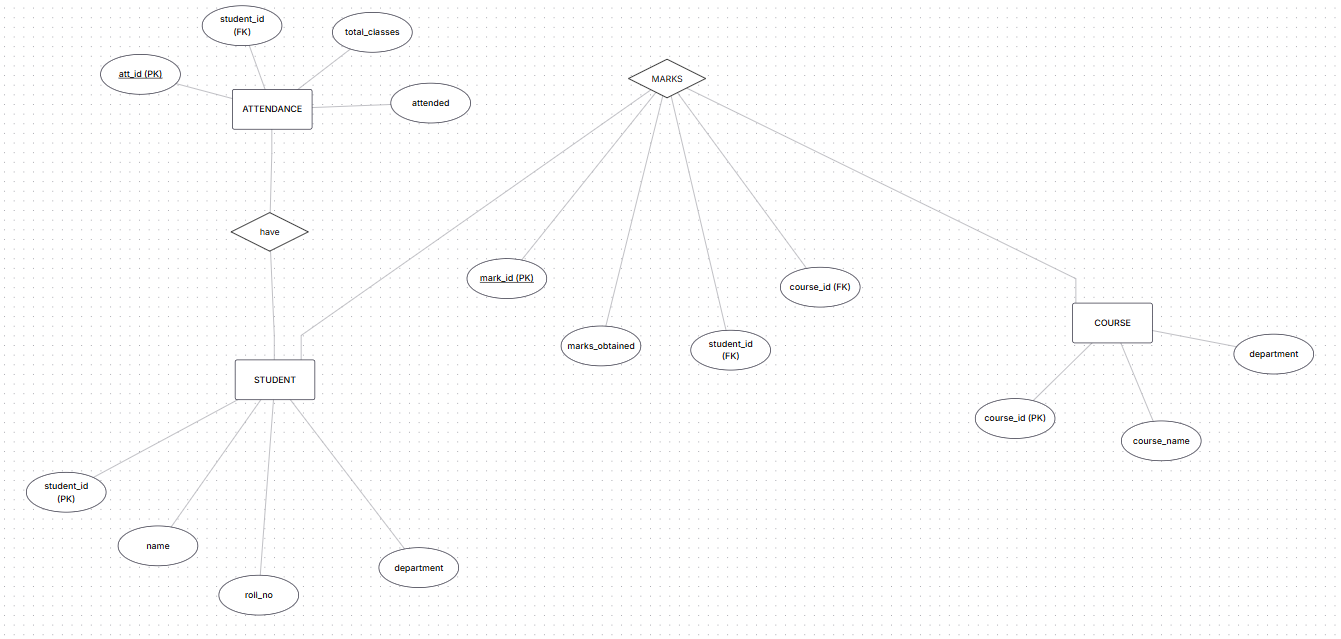
# --- Exit Button ---

tk.Button(root, text="Exit", command=exit\_app, bg="red", fg="white").pack(pady=10)

view\_students()  # Load initial data

root.mainloop()

**Screenshot:**

**ER diagram:**

**ER Diagram (Conceptual Explanation)**

* **STUDENTS** ↔ **MARKS**
  + One student can have **many marks** (one per course).
  + Hence, **1 : M relationship**.
* **COURSES** ↔ **MARKS**
  + One course can have **many students’ marks**.
  + Hence, **1 : M relationship**.
* **STUDENTS** ↔ **ATTENDANCE**
  + Each student can have **one or multiple attendance records**, depending on design.
  + Typically **1 : 1 or 1 : M**.