# STUDENT MANAGEMENT SYSTEM CODE

PROGRAM:

```
import sqlite3
import tkinter as tk
from tkinter import ttk, messagebox

# ====== CONFIGURATION =======
DEBUG_MODE = True  # Set False to silence debug prints
DB_NAME = "students.db"
ADMIN_USERNAME = "admin"
ADMIN_PASSWORD = "admin123"
def log_debug(msg):
    if DEBUG_MODE:
        print("[DEBUG]", msg)
# ====== DATABASE SETUP =======
conn = sqlite3.connect(DB_NAME)
cursor = conn.cursor()
# DROP and recreate the table fresh on each run — for testing only!
cursor.execute("DROP TABLE IF EXISTS students")
cursor.execute("""
    CREATE TABLE students (
        student_id TEXT PRIMARY KEY,
        name TEXT NOT NULL,
        contact TEXT NOT NULL,
        grade TEXT NOT NULL
    )
""")
conn.commit()
log_debug("Dropped and recreated 'students' table.")
# ====== TKINTER SETUP =======
root = tk.Tk()
root.title("Student Management System")
root.geometry("950x600")
selected_student_id = None
# ====== FUNCTIONS =======
def clear_fields():
    id_var.set("")
    name_var.set("")
    contact_var.set("")
    grade_var.set("")
    search_var.set("")
    global selected_student_id
    selected_student_id = None
    tree.selection_remove(tree.selection())
def refresh_treeview():
    tree.delete(*tree.get_children())
    try:
        cursor.execute("SELECT * FROM students")
        rows = cursor.fetchall()
        for row in rows:
            tree.insert('', tk.END, values=row)
        clear_fields()
        log_debug(f"Refreshed treeview with {len(rows)} students.")
    except Exception as e:
        messagebox.showerror("Error", "Could not refresh data.")
        log_debug(f"Error refreshing data: {e}")
def add_student():
    sid = id_var.get().strip()
    name = name_var.get().strip()
    contact = contact_var.get().strip()
    grade = grade_var.get().strip()

    if not (sid and name and contact and grade):
        messagebox.showwarning("Missing Info", "Please fill all fields.")
        return
    try:
        cursor.execute("SELECT 1 FROM students WHERE student_id=?", (sid,))
        if cursor.fetchone():
            messagebox.showerror("Duplicate ID", "Student ID already exists.")
            return

        cursor.execute("INSERT INTO students (student_id, name, contact, grade) VALUES
(?, ?, ?, ?)",
                       (sid, name, contact, grade))
        conn.commit()
        refresh_treeview()
        messagebox.showinfo("Success", "Student added successfully.")
        log_debug(f"Added student: {sid}, {name}, {contact}, {grade}")
    except sqlite3.Error as e:
        messagebox.showerror("Database Error", f"An error occurred: {e}")
        log_debug(f"Add error: {e}")

def select_student(event):
    global selected_student_id
    selected = tree.focus()
    if selected:
        values = tree.item(selected)['values']
        if values:
            id_var.set(values[0])
            name_var.set(values[1])
            contact_var.set(values[2])
            grade_var.set(values[3])
            selected_student_id = values[0]
            log_debug(f"Selected student: {values}")

def update_student():
    global selected_student_id
    new_id = id_var.get().strip()
    name = name_var.get().strip()
    contact = contact_var.get().strip()
    grade = grade_var.get().strip()
    if not selected_student_id:
        messagebox.showwarning("No Selection", "Select a student to update.")
        return

    if not (new_id and name and contact and grade):
        messagebox.showwarning("Missing Info", "Please fill all fields.")
        return

    try:
        if new_id != selected_student_id:
            cursor.execute("SELECT 1 FROM students WHERE student_id=?", (new_id,))
            if cursor.fetchone():
                messagebox.showerror("Error", "New Student ID already exists.")
                return
            cursor.execute("DELETE FROM students WHERE student_id=?", (selected_student_id,))
            cursor.execute("INSERT INTO students (student_id, name, contact, grade) VALUES (?, ?, ?,
?)",
                           (new_id, name, contact, grade))
        else:
            cursor.execute("UPDATE students SET name=?, contact=?, grade=? WHERE
student_id=?",
                           (name, contact, grade, new_id))
        conn.commit()
        refresh_treeview()
        messagebox.showinfo("Success", "Student updated.")
        log_debug(f"Updated student: {new_id}, {name}, {contact}, {grade}")
    except sqlite3.Error as e:
        messagebox.showerror("Error", f"Update failed: {e}")
        log_debug(f"Update error: {e}")

def delete_student():
    selected = tree.focus()
    if not selected:
        messagebox.showwarning("No Selection", "Please select a student to delete.")
        return
    try:
        sid = tree.item(selected)['values'][0]
        cursor.execute("DELETE FROM students WHERE student_id=?", (sid,))
        conn.commit()
        refresh_treeview()
        messagebox.showinfo("Deleted", f"Student '{sid}' deleted successfully.")
        log_debug(f"Deleted student: {sid}")
    except Exception as e:
        messagebox.showerror("Error", f"Delete failed: {e}")
        log_debug(f"Delete error: {e}")

def search_students():
    keyword = search_var.get().strip()
    if not keyword:
        messagebox.showwarning("Input Required", "Enter a Student ID or Name to search.")
        return
    tree.delete(*tree.get_children())
    try:
        cursor.execute("SELECT * FROM students WHERE student_id LIKE ? OR name LIKE ?",
                       (f'%{keyword}%', f'%{keyword}%'))
        results = cursor.fetchall()
        if results:
            for row in results:
                tree.insert('', tk.END, values=row)
            log_debug(f"Search results for '{keyword}': {results}")
        else:
            messagebox.showinfo("No Results", "No matching student found.")
            log_debug("Search found no results.")
    except Exception as e:
        messagebox.showerror("Error", f"Search failed: {e}")
        log_debug(f"Search error: {e}")

def login_admin():
    user = username_var.get().strip()
    pwd = password_var.get().strip()
    if user == ADMIN_USERNAME and pwd == ADMIN_PASSWORD:
        enable_ui()
        login_frame.destroy()
        messagebox.showinfo("Login Successful", "Welcome, Admin!")
        log_debug("Admin logged in.")
    else:
        messagebox.showerror("Access Denied", "Invalid username or password.")
        log_debug("Login failed.")

def enable_ui():
    for widget in button_frame.winfo_children():
        widget.config(state=tk.NORMAL)
    for widget in entry_frame.winfo_children():
        if isinstance(widget, tk.Entry):
            widget.config(state=tk.NORMAL)
    search_btn.config(state=tk.NORMAL)
    tree.config(selectmode='browse')
# ====== GUI ELEMENTS ======
# Login Frame (top-right)
login_frame = tk.Frame(root)
login_frame.pack(anchor='ne', padx=10, pady=5)

username_var = tk.StringVar()
password_var = tk.StringVar()

tk.Label(login_frame, text="Admin Login").grid(row=0, column=0, columnspan=2)
tk.Label(login_frame, text="Username").grid(row=1, column=0)
tk.Entry(login_frame, textvariable=username_var).grid(row=1, column=1)
tk.Label(login_frame, text="Password").grid(row=2, column=0)
tk.Entry(login_frame, textvariable=password_var, show="*").grid(row=2, column=1)
tk.Button(login_frame, text="Login", command=login_admin).grid(row=3, column=0,
columnspan=2, pady=5)
# Entry Frame
entry_frame = tk.LabelFrame(root, text="Student Details")
```

```
entry_frame.pack(padx=10, pady=10, fill='x')

id_var = tk.StringVar()
name_var = tk.StringVar()
contact_var = tk.StringVar()
grade_var = tk.StringVar()
tk.Label(entry_frame, text="Student ID").grid(row=0, column=0, padx=5, pady=5)
tk.Entry(entry_frame, textvariable=id_var, state='disabled').grid(row=0, column=1, padx=5)

tk.Label(entry_frame, text="Name").grid(row=0, column=2, padx=5, pady=5)
tk.Entry(entry_frame, textvariable=name_var, state='disabled').grid(row=0, column=3, padx=5)

tk.Label(entry_frame, text="Contact").grid(row=1, column=0, padx=5, pady=5)
tk.Entry(entry_frame, textvariable=contact_var, state='disabled').grid(row=1, column=1, padx=5)

tk.Label(entry_frame, text="Grade").grid(row=1, column=2, padx=5, pady=5)
tk.Entry(entry_frame, textvariable=grade_var, state='disabled').grid(row=1, column=3, padx=5)
# Buttons Frame
button_frame = tk.Frame(root)
button_frame.pack(pady=10)
tk.Button(button_frame, text="Add Student", width=15, command=add_student,
state='disabled').grid(row=0, column=0, padx=5)
tk.Button(button_frame, text="Update Student", width=15, command=update_student,
state='disabled').grid(row=0, column=1, padx=5)
tk.Button(button_frame, text="Delete Student", width=15, command=delete_student,
state='disabled').grid(row=0, column=2, padx=5)
tk.Button(button_frame, text="View All", width=15, command=refresh_treeview,
state='disabled').grid(row=0, column=3, padx=5)
tk.Button(button_frame, text="Clear", width=15, command=clear_fields,
state='disabled').grid(row=0, column=4, padx=5)
# Search Frame
search_frame = tk.Frame(root)
search_frame.pack(pady=5)
search_var = tk.StringVar()
tk.Entry(search_frame, textvariable=search_var, width=30).grid(row=0, column=0, padx=5)
search_btn = tk.Button(search_frame, text="Search", command=search_students, state='disabled')
search_btn.grid(row=0, column=1)
# Treeview for displaying students
tree = ttk.Treeview(root, columns=("ID", "Name", "Contact", "Grade"), show='headings',
selectmode='none')
tree.heading("ID", text="Student ID")
tree.heading("Name", text="Name")
tree.heading("Contact", text="Contact")
tree.heading("Grade", text="Grade")
tree.column("ID", width=100)
tree.column("Name", width=150)
tree.column("Contact", width=120)
tree.column("Grade", width=80)
tree.bind("<<TreeviewSelect>>", select_student)
tree.pack(padx=10, pady=10, fill='both', expand=True)
root.mainloop()
conn.close()
```