FRONT END PYTHON CODE

```
import tkinter as tk
from tkinter import messagebox, ttk
from tkinter import simpledialog
import mysql.connector
from datetime import date


# ----------------------------
# Database Connection
# ----------------------------
def connect_db():
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="Dhanush@2006",
            database="ComplaintManagement"
        )
        return conn
    except mysql.connector.Error as err:
        messagebox.showerror("Database Error", f"Error: {err}")
        return None


# ----------------------------
# Add Complaint
# ----------------------------
def add_complaint():
    user_id = entry_user_id.get()
    description = entry_description.get("1.0", tk.END).strip()
    if not user_id or not description:
        messagebox.showwarning("Input Error", "Please fill all fields")
```

```
        return
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.callproc("AddComplaint", [user_id, description])
            conn.commit()
            messagebox.showinfo("Success", "Complaint added successfully!")
            entry_user_id.delete(0, tk.END)
            entry_description.delete("1.0", tk.END)
            view_complaints()
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Database Error: {err}")
        finally:
            conn.close()

# ----------------------------
# Add Response
# ----------------------------
def add_response():
    complaint_id = entry_complaint_id.get()
    admin_id = entry_admin_id.get()
    response_text = entry_response.get("1.0", tk.END).strip()
    if not complaint_id or not admin_id or not response_text:
        messagebox.showwarning("Input Error", "Please fill all fields")
        return
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.callproc("AddResponse", [complaint_id, admin_id, response_text])
            conn.commit()
```

```
            messagebox.showinfo("Success", "Response added successfully!")
            entry_complaint_id.delete(0, tk.END)
            entry_admin_id.delete(0, tk.END)
            entry_response.delete("1.0", tk.END)
            view_complaints()
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Database Error: {err}")
        finally:
            conn.close()

# ----------------------------
# Delete Complaint
# ----------------------------
def delete_complaint():
    complaint_id = entry_delete_id.get()
    if not complaint_id:
        messagebox.showwarning("Input Error", "Please enter complaint ID")
        return
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.callproc("DeleteComplaint", [complaint_id])
            conn.commit()
            messagebox.showinfo("Deleted", f"Complaint {complaint_id} deleted successfully!")
            entry_delete_id.delete(0, tk.END)
            view_complaints()
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Database Error: {err}")
        finally:
            conn.close()
```

```
# ----------------------------
# View All Complaints
# ----------------------------
def view_complaints():
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        cursor.execute("""
            SELECT c.complaint_id, u.name, c.description, c.status, c.submit_date
            FROM Complaint c
            JOIN User u ON c.user_id = u.user_id
            ORDER BY c.submit_date DESC
        """)
        rows = cursor.fetchall()
        conn.close()
        tree.delete(*tree.get_children())
        for row in rows:
            tree.insert("", tk.END, values=row)

# ----------------------------
# Search Complaints by Status
# ----------------------------
def search_by_status():
    status = combo_status.get()
    if not status:
        messagebox.showwarning("Input Error", "Please select a status")
        return
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.callproc("SearchComplaintsByStatus", [status])
```

```
            for result in cursor.stored_results():
                rows = result.fetchall()
            tree.delete(*tree.get_children())
            for row in rows:
                tree.insert("", tk.END, values=row)
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Database Error: {err}")
        finally:
            conn.close()

# ----------------------------
# Get Complaints by User
# ----------------------------
def complaints_by_user():
    user_id = simpledialog.askinteger("User ID", "Enter User ID:")
    if user_id is None:
        return
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.callproc("GetComplaintsByUser", [user_id])
            for result in cursor.stored_results():
                rows = result.fetchall()
            tree.delete(*tree.get_children())
            for row in rows:
                tree.insert("", tk.END, values=(row[0], f"User {user_id}", row[1], row[3], row[4]))
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Database Error: {err}")
        finally:
            conn.close()
```

```
# ----------------------------
# Get Complaints by Date Range
# ----------------------------
def complaints_by_date():
    start_date = simpledialog.askstring("Start Date", "Enter start date (YYYY-MM-DD):")
    end_date = simpledialog.askstring("End Date", "Enter end date (YYYY-MM-DD):")
    if not start_date or not end_date:
        return
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.callproc("GetComplaintsByDate", [start_date, end_date])
            for result in cursor.stored_results():
                rows = result.fetchall()
            tree.delete(*tree.get_children())
            for row in rows:
                tree.insert("", tk.END, values=row)
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Database Error: {err}")
        finally:
            conn.close()

# ----------------------------
# Count Complaints by Status
# ----------------------------
def count_complaints_by_status():
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.callproc("CountComplaintsByStatus")
```

```
            for result in cursor.stored_results():
                rows = result.fetchall()
            message = "\n".join([f"{row[0]}: {row[1]}" for row in rows])
            messagebox.showinfo("Complaint Status Summary", message)
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Database Error: {err}")
        finally:
            conn.close()

# ----------------------------
# Tkinter GUI Setup
# ----------------------------
root = tk.Tk()
root.title("Complaint Management System")
root.geometry("950x700")
root.config(bg="#e8f0fe")

tk.Label(root, text="Complaint Management System", font=("Arial", 20, "bold"), bg="#e8f0fe",
fg="#333").pack(pady=10)

# --- Add Complaint Frame ---
frame_add = tk.LabelFrame(root, text="Add Complaint", bg="#e8f0fe", padx=10, pady=10)
frame_add.pack(fill="x", padx=20, pady=5)
tk.Label(frame_add, text="User ID:", bg="#e8f0fe").grid(row=0, column=0, padx=5, pady=5)
entry_user_id = tk.Entry(frame_add)
entry_user_id.grid(row=0, column=1, padx=5, pady=5)
tk.Label(frame_add, text="Description:", bg="#e8f0fe").grid(row=1, column=0, padx=5, pady=5)
entry_description = tk.Text(frame_add, height=2, width=40)
entry_description.grid(row=1, column=1, padx=5, pady=5)
tk.Button(frame_add, text="Add Complaint", bg="#4CAF50", fg="white",
command=add_complaint).grid(row=1, column=2, padx=10)

# --- Add Response Frame ---
```

```
frame_response = tk.LabelFrame(root, text="Add Response", bg="#e8f0fe", padx=10, pady=10)

frame_response.pack(fill="x", padx=20, pady=5)

tk.Label(frame_response, text="Complaint ID:", bg="#e8f0fe").grid(row=0, column=0, padx=5, pady=5)

entry_complaint_id = tk.Entry(frame_response)

entry_complaint_id.grid(row=0, column=1, padx=5, pady=5)

tk.Label(frame_response, text="Admin ID:", bg="#e8f0fe").grid(row=1, column=0, padx=5, pady=5)

entry_admin_id = tk.Entry(frame_response)

entry_admin_id.grid(row=1, column=1, padx=5, pady=5)

tk.Label(frame_response, text="Response:", bg="#e8f0fe").grid(row=2, column=0, padx=5, pady=5)

entry_response = tk.Text(frame_response, height=2, width=40)

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

tk.Button(frame_response, text="Add Response", bg="#2196F3", fg="white", command=add_response).grid(row=2, column=2, padx=10)


# --- Delete Complaint Frame ---

frame_delete = tk.LabelFrame(root, text="Delete Complaint", bg="#e8f0fe", padx=10, pady=10)

frame_delete.pack(fill="x", padx=20, pady=5)

tk.Label(frame_delete, text="Complaint ID:", bg="#e8f0fe").grid(row=0, column=0, padx=5, pady=5)

entry_delete_id = tk.Entry(frame_delete)

entry_delete_id.grid(row=0, column=1, padx=5, pady=5)

tk.Button(frame_delete, text="Delete", bg="#f44336", fg="white", command=delete_complaint).grid(row=0, column=2, padx=10)


# --- Search by Status Frame ---

frame_search = tk.LabelFrame(root, text="Search by Status", bg="#e8f0fe", padx=10, pady=10)

frame_search.pack(fill="x", padx=20, pady=5)

tk.Label(frame_search, text="Select Status:", bg="#e8f0fe").grid(row=0, column=0, padx=5, pady=5)

combo_status = ttk.Combobox(frame_search, values=["Pending", "Resolved"])

combo_status.grid(row=0, column=1, padx=5, pady=5)

tk.Button(frame_search, text="Search", bg="#FF9800", fg="white", command=search_by_status).grid(row=0, column=2, padx=10)
```

```
# --- Additional Actions Frame ---
frame_actions = tk.LabelFrame(root, text="Additional Actions", bg="#e8f0fe", padx=10, pady=10)
frame_actions.pack(fill="x", padx=20, pady=5)
tk.Button(frame_actions, text="View Complaints by User", bg="#673AB7", fg="white",
command=complaints_by_user).grid(row=0, column=0, padx=10, pady=5)
tk.Button(frame_actions, text="View Complaints by Date", bg="#009688", fg="white",
command=complaints_by_date).grid(row=0, column=1, padx=10, pady=5)
tk.Button(frame_actions, text="Count Complaints by Status", bg="#795548", fg="white",
command=count_complaints_by_status).grid(row=0, column=2, padx=10, pady=5)

# --- View Complaints Treeview ---
frame_view = tk.LabelFrame(root, text="Complaint Records", bg="#e8f0fe", padx=10, pady=10)
frame_view.pack(fill="both", expand=True, padx=20, pady=5)
columns = ("Complaint ID", "User", "Description", "Status", "Submit Date")
tree = ttk.Treeview(frame_view, columns=columns, show="headings")
for col in columns:
    tree.heading(col, text=col)
    tree.column(col, width=150)
tree.pack(fill="both", expand=True)

# Load initial complaints
view_complaints()

root.mainloop()
```