**Python II**

**Student ID : C0930321**

**Student Name : Shreejana Shrestha**

**Assignment: 5**

**----------------------------------------------------------------------------------------------------------------**

# # GUI based python application to perform CRUD Operation using SQLITE3

## # Creation of database and table

A screenshot of a computer

Description automatically generated

## # Inserting the record and showing the success message

A screenshot of a computer screen

Description automatically generated

### # Record displayed after insertion of data

A screenshot of a computer program

Description automatically generated

# Displaying all the record on click of display button

A screenshot of a computer screen

Description automatically generated

### # Selecting the record to perform update or delete operation

A screenshot of a computer screen

Description automatically generated

## # Updating of the record

A screenshot of a computer screen

Description automatically generated

### # After update of record that is selected above

A screenshot of a computer screen

Description automatically generated

## # Deleting the record by selecting the row from the listed table

A screenshot of a computer screen

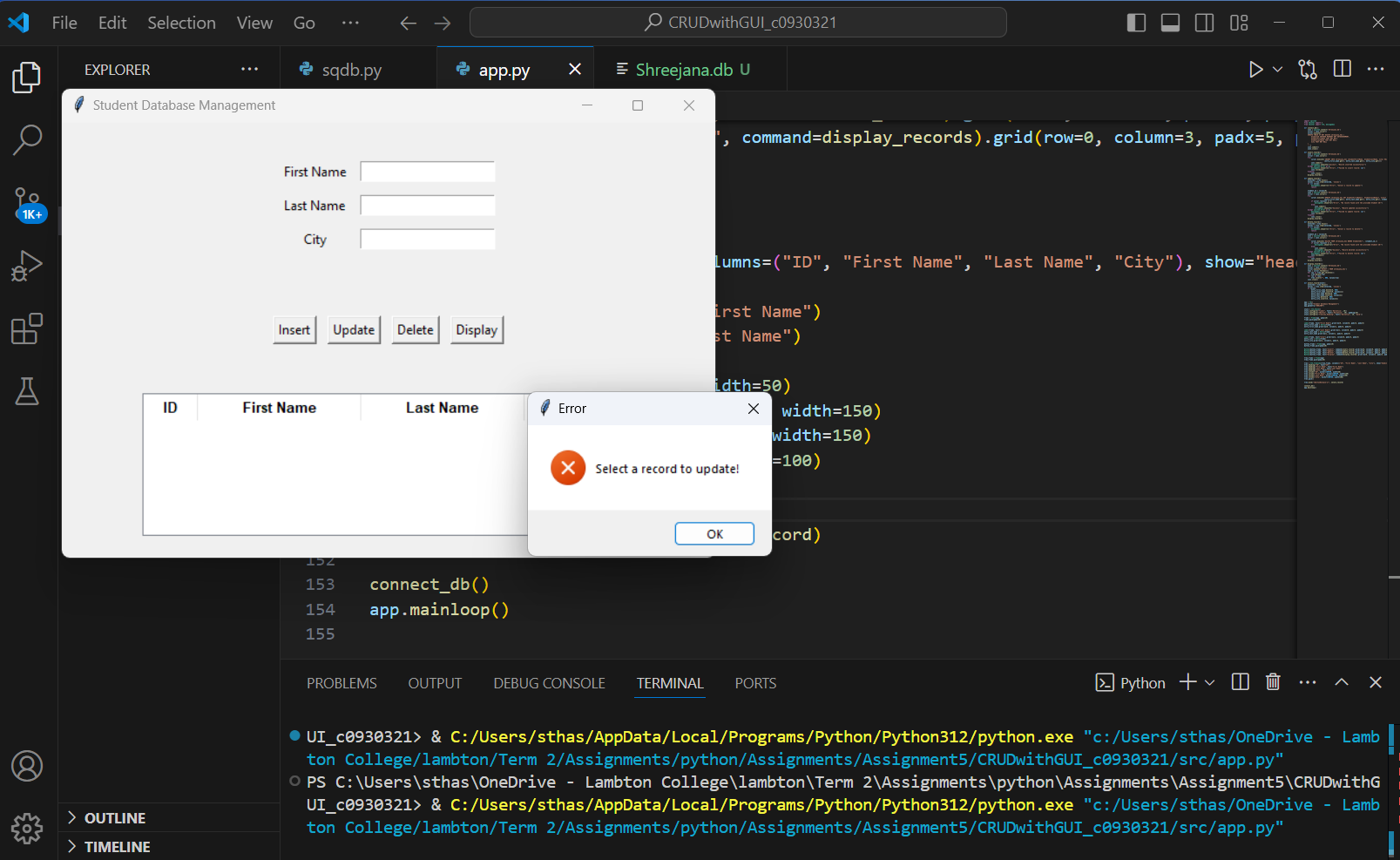
Description automatically generated

### # List of records after deletion

A screenshot of a computer screen

Description automatically generated

## # Error handling and showing message while trying to update / delete without selecting any record



A screenshot of a computer

Description automatically generated

## # All the list of records after CRUD from GUI

A screenshot of a computer

Description automatically generated

# # Source code with brief description

import sqlite3

from tkinter import \*

from tkinter import ttk, messagebox

# database connection and table creation

def connect\_db():

    conn = sqlite3.connect('Shreejana.db')

    cursor = conn.cursor()

    cursor.execute('''

    CREATE TABLE IF NOT EXISTS shreejana\_321 (

        StudentID INTEGER PRIMARY KEY AUTOINCREMENT,

        StudentFirstName TEXT NOT NULL,

        StudentLastName TEXT NOT NULL,

        City TEXT NOT NULL

    )

    ''')

    conn.commit()

    conn.close()

def insert\_record():

    '''

    Takes input from the user (First Name, Last Name, City) and inserts a new record into the database.

    If the insertion is successful, a success message is displayed.

    If there is an error, an error message is displayed, and the transaction is rolled back

    '''

    conn = sqlite3.connect('Shreejana.db')

    cursor = conn.cursor()

    try:

        cursor.execute('INSERT INTO shreejana\_321 (StudentFirstName, StudentLastName, City) VALUES (?, ?, ?)',

                       (entry\_first\_name.get(), entry\_last\_name.get(), entry\_city.get()))

        conn.commit()

        messagebox.showinfo("Success", "Record inserted successfully!")

    except sqlite3.Error as e:

        messagebox.showerror("Error", f"Failed to insert record: {e}")

        conn.rollback()

    finally:

        conn.close()

    display\_records()

def update\_record():

    '''

    Updates an existing record in the database. It retrieves the selected record from the Treeview,

    takes new input values, and updates the corresponding record in the database.

    It handles errors such as no selection and database errors, displaying appropriate messages.

    '''

    selected = tree.focus()

    values = tree.item(selected, 'values')

    if not values:

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

        return

    student\_id = values[0]

    conn = sqlite3.connect('Shreejana.db')

    cursor = conn.cursor()

    try:

        cursor.execute('UPDATE shreejana\_321 SET StudentFirstName=?, StudentLastName=?, City=? WHERE StudentID=?',

                       (entry\_first\_name.get(), entry\_last\_name.get(), entry\_city.get(), student\_id))

        if cursor.rowcount == 0:

            messagebox.showerror("Error", "No record found with the provided Student ID!")

        else:

            conn.commit()

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

    except sqlite3.Error as e:

        messagebox.showerror("Error", f"Failed to update record: {e}")

        conn.rollback()

    finally:

        conn.close()

    display\_records()

def delete\_record():

    '''

    Deletes a selected record from the database.

    It retrieves the selected record from the Treeview and deletes it from the database.

    Appropriate success or error messages are displayed based on the operation's outcome.

    '''

    selected = tree.focus()

    values = tree.item(selected, 'values')

    if not values:

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

        return

    student\_id = values[0]

    conn = sqlite3.connect('Shreejana.db')

    cursor = conn.cursor()

    try:

        cursor.execute('DELETE FROM shreejana\_321 WHERE StudentID=?', (student\_id,))

        if cursor.rowcount == 0:

            messagebox.showerror("Error", "No record found with the provided Student ID!")

        else:

            conn.commit()

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

    except sqlite3.Error as e:

        messagebox.showerror("Error", f"Failed to delete record: {e}")

        conn.rollback()

    finally:

        conn.close()

    display\_records()

def display\_records():

    '''

    Fetches all records from the database and displays them in the Listbox.

    The list is updated each time a record is inserted, updated, or deleted to

    reflect the current state of the database.

    '''

    conn = sqlite3.connect('Shreejana.db')

    cursor = conn.cursor()

    cursor.execute('SELECT \* FROM shreejana\_321')

    rows = cursor.fetchall()

    for row in tree.get\_children():

        tree.delete(row)

    for row in rows:

        tree.insert("", END, values=row)

    conn.close()

def select\_record(event):

    '''

    Handle the selection of a record from the Treeview. When a row is selected,

    it populates the entry fields with the selected record's data for easy updating or deletion.

    '''

    selected = tree.focus()

    values = tree.item(selected, 'values')

    if values:

        entry\_first\_name.delete(0, END)

        entry\_first\_name.insert(0, values[1])

        entry\_last\_name.delete(0, END)

        entry\_last\_name.insert(0, values[2])

        entry\_city.delete(0, END)

        entry\_city.insert(0, values[3])

# GUI section logic

app = Tk()

app.title("Student Database Management")

app.geometry("600x400")

style = ttk.Style()

style.configure("TLabel", font=('Helvetica', 10))

style.configure("TButton", font=('Helvetica', 10), padding=10)

style.configure("Treeview.Heading", font=('Helvetica', 10, 'bold'))

frame = Frame(app, pady=10)

frame.pack(pady=20)

#  input for StudentID, First Name, Last Name, and City using label and entry widgets

Label(frame, text="First Name").grid(row=0, column=0, padx=5, pady=5)

entry\_first\_name = Entry(frame)

entry\_first\_name.grid(row=0, column=1, padx=5, pady=5)

Label(frame, text="Last Name").grid(row=1, column=0, padx=5, pady=5)

entry\_last\_name = Entry(frame)

entry\_last\_name.grid(row=1, column=1, padx=5, pady=5)

Label(frame, text="City").grid(row=2, column=0, padx=5, pady=5)

entry\_city = Entry(frame)

entry\_city.grid(row=2, column=1, padx=5, pady=5)

button\_frame = Frame(app, pady=10)

button\_frame.pack(pady=10)

Button(button\_frame, text="Insert", command=insert\_record).grid(row=0, column=0, padx=5, pady=5)

Button(button\_frame, text="Update", command=update\_record).grid(row=0, column=1, padx=5, pady=5)

Button(button\_frame, text="Delete", command=delete\_record).grid(row=0, column=2, padx=5, pady=5)

Button(button\_frame, text="Display", command=display\_records).grid(row=0, column=3, padx=5, pady=5)

# using treeview widget to display the records in tabular format with column headings

tree\_frame = Frame(app)

tree\_frame.pack(pady=20)

tree = ttk.Treeview(tree\_frame, columns=("ID", "First Name", "Last Name", "City"), show="headings")

tree.heading("ID", text="ID")

tree.heading("First Name", text="First Name")

tree.heading("Last Name", text="Last Name")

tree.heading("City", text="City")

tree.column("ID", anchor=CENTER, width=50)

tree.column("First Name", anchor=CENTER, width=150)

tree.column("Last Name", anchor=CENTER, width=150)

tree.column("City", anchor=CENTER, width=100)

tree.pack()

tree.bind("<ButtonRelease-1>", select\_record)

# establish a connection to the database

connect\_db()

app.mainloop()