task2.py

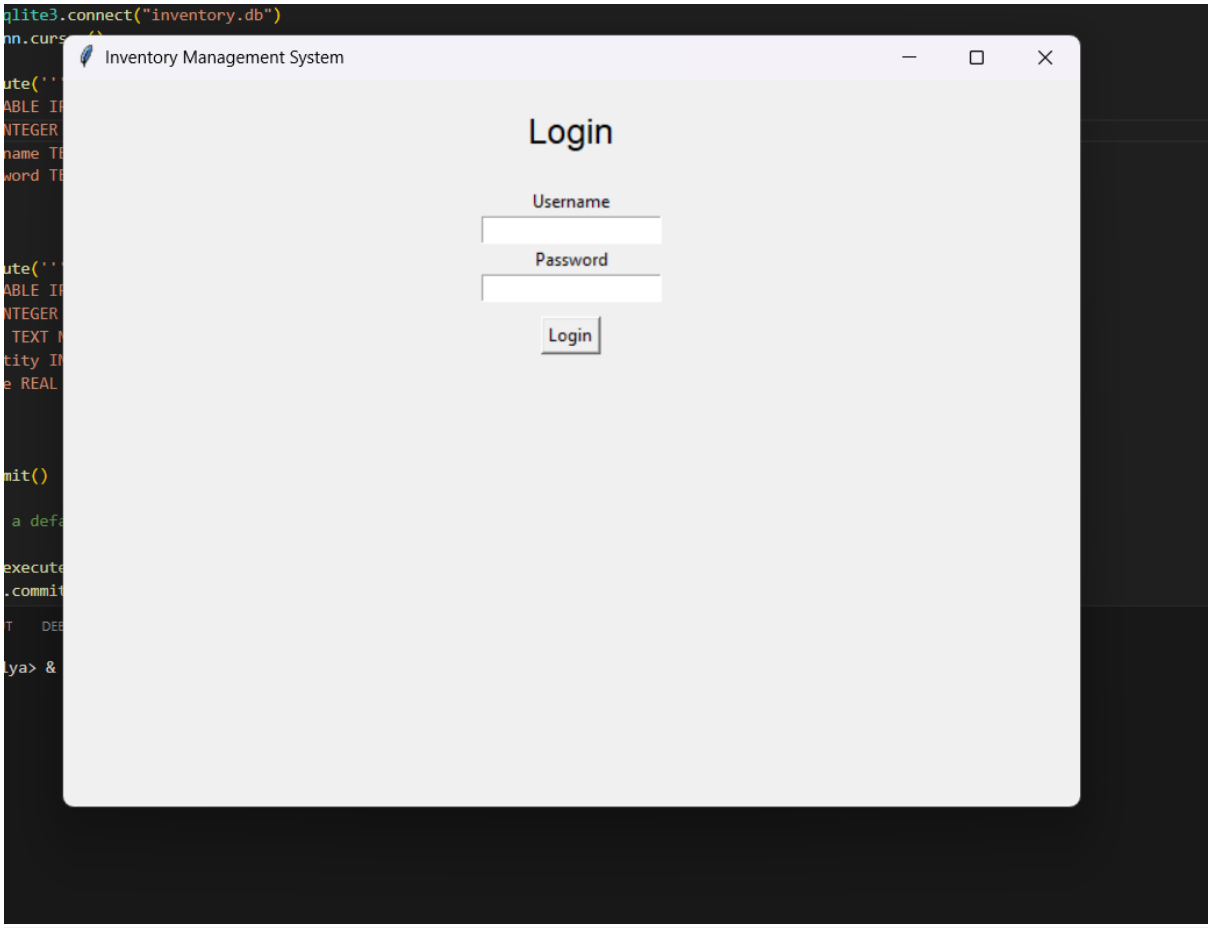
C: > Users > kalya > Desktop > task2.py > ...

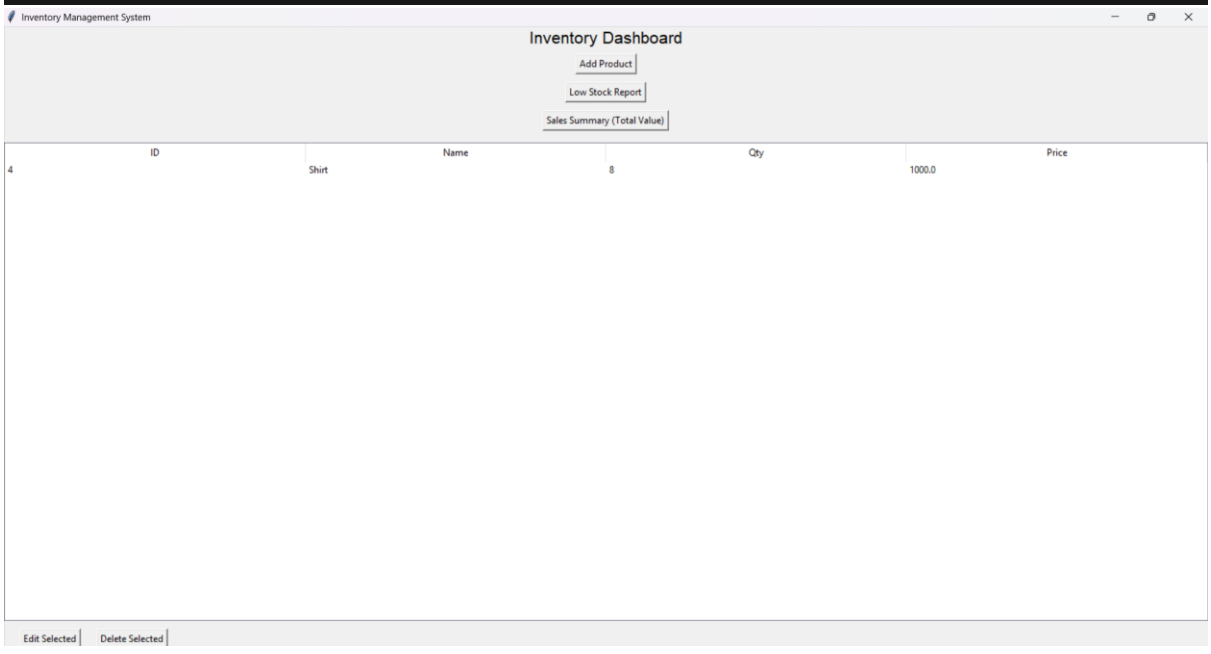
```python
import sqlite3
from tkinter import *
from tkinter import messagebox, ttk

# Database setup
conn = sqlite3.connect("inventory.db")
cur = conn.cursor()

cur.execute('''
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE NOT NULL,
    password TEXT NOT NULL
)
''')

cur.execute('''
CREATE TABLE IF NOT EXISTS products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    quantity INTEGER NOT NULL,
    price REAL NOT NULL
)
''')

conn.commit()

# Insert a default user
try:
    cur.execute("INSERT INTO users (username, password) VALUES (?, ?)", ("admin", "admin123"))
    conn.commit()
except sqlite3.IntegrityError:
    pass

# Main Application
class InventoryApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Inventory Management System")
        self.root.geometry("700x500")
        self.create_login_window()

    def create_login_window(self):
        self.clear_window()
        Label(self.root, text="Login", font=("Arial", 18)).pack(pady=20)
        Label(self.root, text="Username").pack()
        self.username_entry = Entry(self.root)
        self.username_entry.pack()

        Label(self.root, text="Password").pack()
        self.password_entry = Entry(self.root, show='*')
        self.password_entry.pack()

        Button(self.root, text="Login", command=self.login).pack(pady=10)

    def login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        cur.execute("SELECT * FROM users WHERE username=? AND password=?", (username, password))
        if cur.fetchone():
            self.create_main_window()
        else:
            messagebox.showerror("Login Failed", "Invalid credentials!")

    def create_main_window(self):
        self.clear_window()
        Label(self.root, text="Inventory Dashboard", font=("Arial", 16)).pack()

        Button(self.root, text="Add Product", command=self.add_product_window).pack(pady=5)
        Button(self.root, text="Low Stock Report", command=self.low_stock_report).pack(pady=5)
        Button(self.root, text="Sales Summary (Total Value)", command=self.sales_summary).pack(pady=5)

        self.tree = ttk.Treeview(self.root, columns=("ID", "Name", "Qty", "Price"), show='headings')
        for col in self.tree["columns"]:
            self.tree.heading(col, text=col)
        self.tree.pack(expand=True, fill='both', pady=10)

        Button(self.root, text="Edit Selected", command=self.edit_selected_product).pack(side=LEFT, padx=20)
        Button(self.root, text="Delete Selected", command=self.delete_selected_product).pack(side=LEFT)

        self.refresh_product_list()

    def refresh_product_list(self):
        for row in self.tree.get_children():
            self.tree.delete(row)
        cur.execute("SELECT * FROM products")
        for product in cur.fetchall():
            self.tree.insert('', END, values=product)
```

```
    def add_product_window(self):
        top = Toplevel(self.root)
        top.title("Add Product")
        Label(top, text="Product Name").pack()
        name_entry = Entry(top)
        name_entry.pack()
        Label(top, text="Quantity").pack()
        quantity_entry = Entry(top)
        quantity_entry.pack()
        Label(top, text="Price").pack()
        price_entry = Entry(top)
        price_entry.pack()

        def add_product():
            name = name_entry.get()
            try:
                quantity = int(quantity_entry.get())
                price = float(price_entry.get())
                if quantity < 0 or price < 0:
                    raise ValueError
            except ValueError:
                messagebox.showerror("Invalid Input", "Please enter valid numeric values.")
                return

            cur.execute("INSERT INTO products (name, quantity, price) VALUES (?, ?, ?)",
                        (name, quantity, price))
            conn.commit()
            top.destroy()
            self.refresh_product_list()

        Button(top, text="Add", command=add_product).pack()

    def edit_selected_product(self):
        selected = self.tree.focus()
        if not selected:
            messagebox.showwarning("Select Product", "Please select a product to edit.")
            return
        values = self.tree.item(selected, 'values')

        top = Toplevel(self.root)
        top.title("Edit Product")
```

```
        Label(top, text="Product Name").pack()
        name_entry = Entry(top)
        name_entry.insert(0, values[1])
        name_entry.pack()
        Label(top, text="Quantity").pack()
        quantity_entry = Entry(top)
        quantity_entry.insert(0, values[2])
        quantity_entry.pack()
        Label(top, text="Price").pack()
        price_entry = Entry(top)
        price_entry.insert(0, values[3])
        price_entry.pack()

        def save_changes():
            name = name_entry.get()
            try:
                quantity = int(quantity_entry.get())
                price = float(price_entry.get())
                if quantity < 0 or price < 0:
                    raise ValueError
            except ValueError:
                messagebox.showerror("Invalid Input", "Please enter valid numeric values.")
                return

            cur.execute("UPDATE products SET name=?, quantity=?, price=? WHERE id=?",
                        (name, quantity, price, values[0]))
            conn.commit()
            top.destroy()
            self.refresh_product_list()

        Button(top, text="Save Changes", command=save_changes).pack()

    def delete_selected_product(self):
        selected = self.tree.focus()
        if not selected:
            messagebox.showwarning("Select Product", "Please select a product to delete.")
            return
        values = self.tree.item(selected, 'values')
        confirm = messagebox.askyesno("Delete Confirmation", f"Delete product: {values[1]}?")
        if confirm:
            cur.execute("DELETE FROM products WHERE id=?", (values[0],))
            conn.commit()
```

```
175
176     def low_stock_report(self):
177         top = Toplevel(self.root)
178         top.title("Low Stock Report")
179         Label(top, text="Products with Low Stock (< 5)").pack()
180
181         tree = ttk.Treeview(top, columns=("ID", "Name", "Qty"), show='headings')
182         for col in tree["columns"]:
183             tree.heading(col, text=col)
184         tree.pack(fill='both', expand=True)
185
186         cur.execute("SELECT id, name, quantity FROM products WHERE quantity < 5")
187         for row in cur.fetchall():
188             tree.insert('', END, values=row)
189
190     def sales_summary(self):
191         cur.execute("SELECT SUM(quantity * price) FROM products")
192         total_value = cur.fetchone()[0]
193         messagebox.showinfo("Sales Summary", f"Total Inventory Value: ${total_value:.2f}")
194
195     def clear_window(self):
196         for widget in self.root.winfo_children():
197             widget.destroy()
198
199 # Start the app
200 if __name__ == "__main__":
201     root = Tk()
202     app = InventoryApp(root)
203     root.mainloop()
```


glite3.connect("inventory.db")
Inventory Management System
Login
Username
Password
Login


Inventory Management System
Inventory Dashboard
Add Product
Low Stock Report
Sales Summary (Total Value)
ID
Name
Qty
Price
4
Shirt
8
1000.0
Edit Selected
Delete Selected