Tutorial Report: Real-Time Barcode Inventory Management System using OpenCV and SQLite

1. Introduction

This project implements a **real-time barcode inventory management system** using a webcam. It scans barcodes, updates an inventory database, and prints live status updates. The system supports **multi-threaded processing** for efficient handling of barcode events and ensures persistent storage with **SQLite**.

---

2. System Overview

The application follows this modular workflow:

1. **Barcode Scanning** – Live video feed captures barcodes using OpenCV and decodes them via pyzbar.
2. **Inventory Update** – Based on user action (a to add, s to subtract), the database is updated.
3. **Threaded Queue Handling** – A background thread manages all inventory updates to avoid UI delays.
4. **Console Reporting** – Current inventory is printed after every scan and update.

---

3. Requirements

External Libraries

Install using pip:

pip install opencv-python pyzbar

Built-in Libraries

- sqlite3
- queue
- threading

---

4. Code & Explanation

```
import cv2
import sqlite3
from pyzbar.pyzbar import decode
import threading
import queue
```

4.2 Initialize SQLite Database

```
def init_db():
    conn = sqlite3.connect("barcode_data.db")
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS barcodes (
            sku TEXT PRIMARY KEY,
            quantity INTEGER,
            product_name TEXT,
            price REAL
        )
    """)
    conn.commit()
    return conn, cursor
```

Explanation:
Creates or opens a local SQLite database to store inventory data, with unique SKUs as primary keys.

4.3 Barcode Processing Thread

```
def process_barcodes(barcode_queue, cursor, conn):
    while True:
        action, sku = barcode_queue.get()
        if action == 'EXIT':
            break

        cursor.execute("SELECT * FROM barcodes WHERE sku=?", (sku,))
        result = cursor.fetchone()

        if action == "ADD":
            if result:
                cursor.execute("UPDATE barcodes SET quantity=quantity
            else:
                name = input(f"New SKU {sku}. Enter product name: ")
                price = float(input(f"Enter price for {name}: "))
                cursor.execute("INSERT INTO barcodes VALUES (?, ?, ?,
```

```
        if action == "ADD":
            if result:
                cursor.execute("UPDATE barcodes SET quantity=quantity
            else:
                name = input(f"New SKU {sku}. Enter product name: ")
                price = float(input(f"Enter price for {name}: "))
                cursor.execute("INSERT INTO barcodes VALUES (?, ?, ?,
        elif action == "SUBTRACT":
            if result:
                quantity = result[1]
                if quantity > 1:
                    cursor.execute("UPDATE barcodes SET quantity=quar
                else:
                    cursor.execute("DELETE FROM barcodes WHERE sku=?"

        conn.commit()
        display_inventory(cursor)
```

Explanation:
Background thread handles all database updates to avoid race conditions. New products prompt the user for metadata.

4.4 Display Inventory

```
def display_inventory(cursor):
    print("\nCurrent Inventory:")
    print("{:<15} {:<10} {:<20} {:<10}".format("SKU", "Quantity", "
    print("-" * 60)
    for row in cursor.execute("SELECT * FROM barcodes"):
        print("{:<15} {:<10} {:<20} {:<10.2f}".format(*row))
```

Explanation:
Formats and prints a clean table view of all current inventory records.

4.5 Main Video Loop

```
def start_scanner():
    conn, cursor = init_db()
    barcode_queue = queue.Queue()
    thread = threading.Thread(target=process_barcodes, args=(barcod
    thread.start()

    cap = cv2.VideoCapture(0)
    last_detected = None

    print("Press 'a' to add, 's' to subtract, 'q' to quit...")

    while True:
        ret, frame = cap.read()
        if not ret:
            break
```

```
        barcodes = decode(frame)
        for barcode in barcodes:
            sku = barcode.data.decode("utf-8")
            cv2.rectangle(frame, (barcode.rect.left, barcode.rect.t
                          (barcode.rect.left + barcode.rect.width,
                          (0, 255, 0), 2)
            cv2.putText(frame, sku, (barcode.rect.left, barcode.rec
                        cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0),
            last_detected = sku

        cv2.imshow("Barcode Scanner", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord('a') and last_detected:
            barcode_queue.put(('ADD', last_detected))
        elif key == ord('s') and last_detected:
            barcode_queue.put(('SUBTRACT', last_detected))
```

```
        cv2.imshow("Barcode Scanner", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord('a') and last_detected:
            barcode_queue.put(('ADD', last_detected))
        elif key == ord('s') and last_detected:
            barcode_queue.put(('SUBTRACT', last_detected))
        elif key == ord('q'):
            barcode_queue.put(('EXIT', None))
            break

    cap.release()
    cv2.destroyAllWindows()
    thread.join()
    conn.close()
```

Explanation:
The webcam captures barcodes in real time. Detected barcodes are decoded and visually displayed. User input determines whether to add or subtract quantity.

---

5. Output Example

Detected: SKU 123456789012

Added new product: SKU 123456789012 (Sample Product) to the database.

Current Inventory:

SKU Quantity Product Name Price

123456789012 1 Sample Product 9.99

---

6. Usage Instructions

Step 1: Run the scanner

start_scanner()

Step 2: Use keys to manage inventory

Key	Action
a	Add one unit of detected product
s	Subtract one unit
q	Quit scanner

---

7. Features Summary

Feature	Description
Real-time barcode scanning	Using OpenCV and pyzbar
Persistent inventory	Stored in barcode_data.db
SKU management	Add or remove product quantities
Threaded processing	Keeps UI responsive
Console interface	Prints live inventory

---

8. Enhancement Ideas

Feature	How to Add
GUI inventory manager	Use Tkinter or PyQt
Cloud sync	Integrate with Firebase, Google Sheets, or MongoDB Atlas
Export to CSV	Use pandas to export DB to CSV

Feature	How to Add
REST API	Add Flask server to access DB from browser