**FRESH GUARD TROLLEY**

**CODE FOR DATABASE SETUP**

import sqlite3

# Connect to the SQLite database (or create it)

conn = sqlite3.connect('smart\_trolley.db')

c = conn.cursor()

# Create a table for products

c.execute('''CREATE TABLE IF NOT EXISTS products

(id INTEGER PRIMARY KEY, name TEXT, expiry\_date TEXT, barcode TEXT)''')

# Insert sample data

products = [

('Milk', '2023-12-01', '123456789012'),

('Bread', '2023-11-15', '234567890123'),

('Butter', '2023-10-20', '345678901234')

]

c.executemany('INSERT INTO products (name, expiry\_date, barcode) VALUES (?, ?, ?)', products)

conn.commit()

# Close the connection

conn.close()

**CODE FOR BARCODE READING AND PRODUCT INFORMATION RETRIVAL**

import cv2

from pyzbar import pyzbar

import sqlite3

# Function to read barcodes using OpenCV and pyzbar

def read\_barcodes(frame):

barcodes = pyzbar.decode(frame)

barcode\_data = None

for barcode in barcodes:

x, y, w, h = barcode.rect

barcode\_info = barcode.data.decode('utf-8')

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

font = cv2.FONT\_HERSHEY\_SIMPLEX

cv2.putText(frame, barcode\_info, (x + 10, y - 10), font, 0.5, (0, 255, 0), 2)

barcode\_data = barcode\_info

return frame, barcode\_data

# Function to get product information from the database

def get\_product\_info(barcode):

conn = sqlite3.connect('smart\_trolley.db')

c = conn.cursor()

c.execute('SELECT name, expiry\_date FROM products WHERE barcode=?', (barcode,))

product = c.fetchone()

conn.close()

return product

# Main function to capture video and process barcodes

def main():

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if not ret:

break

frame, barcode = read\_barcodes(frame)

if barcode:

product\_info = get\_product\_info(barcode)

if product\_info:

name, expiry\_date = product\_info

cv2.putText(frame, f'Product: {name}', (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 2)

cv2.putText(frame, f'Expiry: {expiry\_date}', (50, 100), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 2)

cv2.imshow('Smart Trolley', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

if \_\_name\_\_ == "\_\_main\_\_":

main()