INVENTORY-APP

App.py :

from flask import Flask, render\_template, request, jsonify, send\_file

import pymysql

import pandas as pd

from fpdf import FPDF

import os

app = Flask(\_\_name\_\_)

EXPORT\_FOLDER = 'exports'

os.makedirs(EXPORT\_FOLDER, exist\_ok=True)

def get\_db\_connection():

    return pymysql.connect(

        host="localhost",

        user="root",

        password="Aneesa@09",

        database="inventory\_db",

        cursorclass=pymysql.cursors.DictCursor

    )

@app.route('/')

def index():

    return render\_template('index.html')

@app.route('/search')

def search():

    query = request.args.get('q', '').lower()

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT \* FROM products

        WHERE LOWER(name) LIKE %s OR LOWER(barcode) LIKE %s OR LOWER(category) LIKE %s

    """, (f'%{query}%', f'%{query}%', f'%{query}%'))

    products = cursor.fetchall()

    conn.close()

    return jsonify(products)

@app.route('/api/summary')

def summary():

    conn = get\_db\_connection()

    cursor = conn.cursor()

    # 1. Category Counts

    cursor.execute("SELECT category, COUNT(\*) as count FROM products GROUP BY category")

    category\_counts = cursor.fetchall()

    # 2. Low Stock Count

    cursor.execute("SELECT COUNT(\*) as count FROM products WHERE quantity <= threshold")

    low\_stock = cursor.fetchone()

    # 3. Activity Log Summary

    cursor.execute("SELECT action, COUNT(\*) as count FROM activity\_logs GROUP BY action")

    activity = cursor.fetchall()

    # 4. Stock Levels for Heatmap

    cursor.execute("SELECT name AS item, category, quantity AS stock FROM products")

    stock\_levels = cursor.fetchall()

    conn.close()

    return jsonify({

        'category\_counts': category\_counts,

        'low\_stock\_count': low\_stock['count'],

        'activity\_logs': activity,

        'stock\_levels': stock\_levels

    })

@app.route('/export/excel')

def export\_excel():

    conn = get\_db\_connection()

    df = pd.read\_sql\_query("SELECT \* FROM products", conn)

    path = os.path.join(EXPORT\_FOLDER, "products.xlsx")

    df.to\_excel(path, index=False)

    conn.close()

    return send\_file(path, as\_attachment=True)

@app.route('/export/pdf')

def export\_pdf():

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("""

        SELECT activity\_logs.\*, products.name

        FROM activity\_logs JOIN products ON activity\_logs.product\_id = products.id

    """)

    logs = cursor.fetchall()

    pdf = FPDF()

    pdf.add\_page()

    pdf.set\_font("Arial", size=12)

    pdf.cell(200, 10, txt="Inventory Activity Logs", ln=True, align='C')

    pdf.ln(10)

    for log in logs:

        text = f"{log['timestamp']} - {log['name']} - {log['action']} - Qty: {log['quantity']}"

        pdf.cell(200, 10, txt=text, ln=True)

    path = os.path.join(EXPORT\_FOLDER, "logs.pdf")

    pdf.output(path)

    conn.close()

    return send\_file(path, as\_attachment=True)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

STATIC/CHART.JS

// Search Function

document.getElementById('searchInput').addEventListener('input', async function () {

  const query = this.value;

  const res = await fetch(`/search?q=${encodeURIComponent(query)}`);

  const products = await res.json();

  const tbody = document.querySelector('#resultsTable tbody');

  tbody.innerHTML = '';

  products.forEach(p => {

    const row = document.createElement('tr');

    row.innerHTML = `

      <td>${p.name}</td>

      <td>${p.barcode}</td>

      <td>${p.category}</td>

      <td>${p.stock}</td>

      <td>${p.reorder\_level}</td>

    `;

    tbody.appendChild(row);

  });

});

// Load Charts

async function loadCharts() {

  const res = await fetch('/api/summary');

  const data = await res.json();

  // Category Chart

  const catCtx = document.getElementById('categoryChart').getContext('2d');

  new Chart(catCtx, {

    type: 'pie',

    data: {

      labels: data.category\_counts.map(c => c.category),

      datasets: [{

        label: 'Stock by Category',

        data: data.category\_counts.map(c => c.count),

        backgroundColor: ['#26a69a', '#4db6ac', '#80cbc4', '#b2dfdb'],

      }]

    },

    options: {

      plugins: {

        title: {

          display: true,

          text: '📊 Product Categories'

        }

      }

    }

  });

  // Low Stock Chart

  const lowCtx = document.getElementById('lowStockChart').getContext('2d');

  new Chart(lowCtx, {

    type: 'doughnut',

    data: {

      labels: ['Low Stock', 'Sufficient Stock'],

      datasets: [{

        data: [data.low\_stock\_count, 100 - data.low\_stock\_count], // assume max 100 products

        backgroundColor: ['#ff7043', '#b2dfdb']

      }]

    },

    options: {

      plugins: {

        title: {

          display: true,

          text: '⚠️ Low Stock Overview'

        }

      }

    }

  });

  // Activity Logs Chart

  const actCtx = document.getElementById('activityChart').getContext('2d');

  new Chart(actCtx, {

    type: 'bar',

    data: {

      labels: data.activity\_logs.map(a => a.action),

      datasets: [{

        label: 'Log Count',

        data: data.activity\_logs.map(a => a.count),

        backgroundColor: '#26a69a'

      }]

    },

    options: {

      plugins: {

        title: {

          display: true,

          text: '📝 Inventory Activities'

        }

      },

      scales: {

        y: { beginAtZero: true }

      }

    }

  });

}

window.onload = loadCharts;

STATIC/STYLE.CSS

body {

  font-family: 'Segoe UI', sans-serif;

  margin: 0;

  padding: 0;

  background-color: #f2f4f6;

  color: #333;

}

.container {

  max-width: 1100px;

  margin: 40px auto;

  padding: 20px;

  background-color: #ffffff;

  border-radius: 12px;

  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.08);

}

h1 {

  text-align: center;

  color: #006d6d;

  margin-bottom: 30px;

}

input[type="text"] {

  width: 100%;

  padding: 12px 16px;

  margin-bottom: 20px;

  border: 2px solid #cfd8dc;

  border-radius: 8px;

  font-size: 16px;

}

table {

  width: 100%;

  border-collapse: collapse;

  margin-bottom: 40px;

}

th, td {

  padding: 12px 16px;

  text-align: left;

  border-bottom: 1px solid #e0e0e0;

}

th {

  background-color: #e0f2f1;

  color: #00695c;

}

.charts {

  display: grid;

  grid-template-columns: 1fr 1fr 1fr;

  gap: 30px;

  margin-bottom: 40px;

}

canvas {

  background-color: #ffffff;

  padding: 10px;

  border-radius: 10px;

  border: 1px solid #e0e0e0;

}

.export-buttons {

  text-align: center;

}

button {

  padding: 12px 20px;

  margin: 0 10px;

  background-color: #00897b;

  color: white;

  font-size: 16px;

  border: none;

  border-radius: 8px;

  cursor: pointer;

  transition: background-color 0.3s;

}

button:hover {

  background-color: #00695c;

}

TEMPLATES/INDEX.HTML

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Inventory Dashboard</title>

  <style>

    .chart-container {

      width: 100%;

      height: 400px;

      margin-bottom: 2rem;

    }

  </style>

  <!-- Chart.js and matrix plugin -->

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <script src="https://cdn.jsdelivr.net/npm/chartjs-chart-matrix@1.1.0/dist/chartjs-chart-matrix.min.js"></script>

</head>

<body>

  <div class="chart-container">

    <canvas id="categoryChart"></canvas>

  </div>

  <div class="chart-container">

    <canvas id="lowStockChart"></canvas>

  </div>

  <div class="chart-container">

    <canvas id="activityChart"></canvas>

  </div>

  <div class="chart-container">

    <canvas id="stockHeatmap"></canvas>

  </div>

  <script>

    fetch('/api/summary')

      .then(res => res.json())

      .then(data => {

        // Category Chart

        new Chart(document.getElementById('categoryChart'), {

          type: 'bar',

          data: {

            labels: data.category\_counts.map(row => row.category),

            datasets: [{

              label: 'Count',

              data: data.category\_counts.map(row => row.count),

              backgroundColor: 'rgba(75, 192, 192, 0.5)'

            }]

          }

        });

        // Low Stock Chart

        new Chart(document.getElementById('lowStockChart'), {

          type: 'doughnut',

          data: {

            labels: ['Low Stock', 'Sufficient Stock'],

            datasets: [{

              label: 'Stock Status',

              data: [data.low\_stock\_count, 100 - data.low\_stock\_count],

              backgroundColor: ['#ff6384', '#36a2eb']

            }]

          }

        });

        // Activity Chart

        new Chart(document.getElementById('activityChart'), {

          type: 'pie',

          data: {

            labels: data.activity\_logs.map(row => row.action),

            datasets: [{

              label: 'Actions',

              data: data.activity\_logs.map(row => row.count),

              backgroundColor: ['#ffcd56', '#4bc0c0', '#9966ff']

            }]

          }

        });

        // Stock Heatmap

        function getColorForStock(stock) {

          if (stock <= 10) return '#ff4d4d';    // red

          if (stock <= 20) return '#ff9933';    // orange

          if (stock <= 50) return '#ffff66';    // yellow

          return '#66cc66';                     // green

        }

        const heatmapData = data.stock\_levels.map((item) => ({

          x: item.category,

          y: item.item,

          v: item.stock,

          backgroundColor: getColorForStock(item.stock)

        }));

        new Chart(document.getElementById('stockHeatmap'), {

          type: 'matrix',

          data: {

            datasets: [{

              label: 'Stock Heatmap',

              data: heatmapData,

              backgroundColor: ctx => ctx.dataset.data[ctx.dataIndex].backgroundColor,

              width: 40,

              height: 40,

            }]

          },

          options: {

            responsive: true,

            maintainAspectRatio: false,

            scales: {

              x: {

                type: 'category',

                labels: [...new Set(data.stock\_levels.map(d => d.category))],

                title: { display: true, text: 'Category' }

              },

              y: {

                type: 'category',

                labels: [...new Set(data.stock\_levels.map(d => d.item))],

                title: { display: true, text: 'Item' }

              }

            },

            plugins: {

              tooltip: {

                callbacks: {

                  label: ctx => `Stock: ${ctx.raw.v}`

                }

              },

              legend: { display: false }

            }

          }

        });

      });

  </script>

</body>

</html>

INVENTORY\_DB.SQL

CREATE DATABASE inventory\_db;

USE inventory\_db;

-- Products Table

CREATE TABLE products (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(100),

    barcode VARCHAR(50),

    category VARCHAR(50),

    quantity INT,

    threshold INT

);

-- Activity Logs Table

CREATE TABLE activity\_logs (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    product\_id INT,

    action VARCHAR(50),

    quantity INT,

    timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

    FOREIGN KEY (product\_id) REFERENCES products(id)

);

SEED\_DATA.PY

import pymysql

def get\_db\_connection():

    return pymysql.connect(

        host="localhost",

        user="root",

        password="Aneesa@09",

        database="inventory\_db",

        cursorclass=pymysql.cursors.DictCursor

    )

conn = get\_db\_connection()

cursor = conn.cursor()

# Sample product insert

sample\_products = [

    ("Apple", "123456", "Fruit", 100, 20),

    ("Banana", "789012", "Fruit", 120, 30),

    ("Milk", "345678", "Dairy", 50, 10)

]

cursor.executemany("""

    INSERT INTO products (name, barcode, category, quantity, threshold)

    VALUES (%s, %s, %s, %s, %s)

""", sample\_products)

conn.commit()

print("Sample data inserted successfully!")

cursor.close()

conn.close()