Smart Pharmaceutical Inventory System – Final Year Project Document

---

1. Project Overview

The Smart Pharmaceutical Inventory System (SPIS) is a full-stack, cloud-connected, edge-enabled application designed to manage pharmaceutical stock efficiently while monitoring storage conditions in real-time. The system integrates inventory management, POS, IoT sensors, computer vision, live streaming, and motion detection to provide a robust solution for pharmacy management.

Goals

- Efficient inventory and POS management.
- Real-time monitoring of storage conditions.
- Automated alerts for temperature violations and stock shortages.
- Optional computer vision for vial counting.
- Optional motion detection for security and monitoring.

---

2. Workflow

2.1 User and Device Onboarding

1. User creates an account on the app.
2. Each fridge/Pi camera device comes with a unique `device_id` and `token`.
3. User enters the device ID and token in the app.
4. Backend verifies token and pairs device to the user.
5. Raspberry Pi configures itself with user ID and cloud endpoints.

2.2 Stock Receiving

1. Scan new medicine barcode.
2. Enter details: quantity, expiry, manufacturing date, required storage temperature.
3. Backend assigns batch ID and suggests optimal fridge.
4. Restocking: scan barcode, update quantity and details.

2.3 Storage & Monitoring

1. Edge Pi reads temperature/humidity from sensors.
2. Pi runs optional CV inference for vial counting.
3. Pi sends data to backend (HTTP/MQTT).
4. Backend compares temperature with medicine requirements.
5. Dashboard displays current fridge status and alerts.

6. Optional motion detection triggers alerts if unexpected activity is detected.

2.4 POS

1. Scan medicine barcode at point of sale.
2. Apply LIFO logic to determine batch for sale.
3. Update stock in database.
4. Print receipt or digital confirmation.

2.5 Optional Computer Vision

- Camera mounted above tray.
- Detect empty vs filled slots (12×12 grid initially).
- Count remaining vials and reconcile with database.

2.6 Optional Motion Detection

- Pi camera detects movement inside fridge.
- Sends alerts for unauthorized access.
- Optional recording or snapshot storage.

---

3. System Architecture

Services Overview

1. **Backend 1 – Inventory & POS**
2. Node.js + TypeScript + NestJS/Express
3. MongoDB for inventory & batches
4. InfluxDB for sensor data (optional)

5. Handles alerts, stock logic, and batch suggestions

6. **Backend 2 – Edge Pi + Sensors + CV**

7. Python + FastAPI/Flask
8. Reads temperature/humidity sensors
9. Runs CV inference for vial counting
10. Optional motion detection

11. Sends processed data to Backend 1

12. **Backend 3 – Streaming Service**

13. Node.js + TypeScript
14. Connects to Python backend camera feed

15. Provides live streaming via WebSocket/MJPEG

16. **Frontend – Dashboard & POS**

17. React + TypeScript
18. Displays inventory, POS interface, fridge monitoring, alerts, live stream

Data Flow

```
[Pi Sensors + Camera] ---> Backend 2 (Python) ---> Backend 1 (Node.js) --->
Frontend Dashboard
                                    |                                ^
                                    |---[CV stock count & motion detection]-->
                                    |---[Temp data & alerts]------------>

Backend 3 (Node.js) <-- camera feed -- Backend 2 (Python)
```

---

4. Technology Stack

Component	Technology
Backend 1	Node.js, TypeScript, NestJS/Express, MongoDB, InfluxDB (optional)
Backend 2	Python, FastAPI/Flask, OpenCV, TensorFlow Lite (optional), sensor libraries
Backend 3	Node.js, TypeScript, WebSocket/MJPEG streaming
Frontend	React, TypeScript, Tailwind/Material UI/Ant Design
Edge Device	Raspberry Pi 4 (4GB), Pi Camera/USB Webcam, DS18B20/USB temp sensor, optional DHT22 humidity sensor
Cloud/Infra	Docker, GitHub Actions (CI/CD), optional AWS Free Tier for extended storage

---

5. Logic & Algorithms

Inventory Logic

- Stock receiving & restocking: barcode scanning, batch creation, optimal fridge suggestion.
- Batching & sorting: based on expiry, MFG, quantity, and fridge temp.
- POS: LIFO selection based on expiry.

Sensor Logic

- Read temp/humidity every X seconds.
- Compare against thresholds per medicine batch.
- Trigger alerts if exceeded.

• Send data to cloud database.

Computer Vision

• Grid-based detection (12×12 slots)
• Compare current frame vs empty reference frame
• Count filled slots
• Optional: deep learning model (YOLOv5 Nano) for robust vial detection

Motion Detection

• Frame difference / background subtraction
• Trigger alert if unexpected movement
• Optional snapshot storage for review

Alerting Logic

• Threshold-based alerts for temp violations
• Stock discrepancy alerts if CV count != database
• Motion alerts
• Notifications: email, dashboard highlights

---

6. Hardware Requirements & Pricing (India Q2 2025)

Item	Qty	Estimated Cost (INR)
Raspberry Pi 4 (4GB)	1	4500
Pi Camera Module v2	1	900
DS18B20 Temperature Sensor	1 per fridge	150
DHT22 Humidity Sensor (optional)	1 per fridge	300
USB Webcam (optional alternative)	1	800
Misc. Wiring, SD Card, Power	1 set	1000
Total per Fridge		~6850 – 7400 INR

Software costs: all open-source / free (Node.js, Python, React, MongoDB, OpenCV, TensorFlow Lite, Flask/FastAPI)

---

7. Software Requirements

• Node.js v20+ and npm
• Python 3.11+ (with pip)

- MongoDB Community Edition
- InfluxDB (optional)
- OpenCV, TensorFlow Lite
- Flask / FastAPI
- Docker & Docker Compose (optional)
- React + TypeScript development environment

---

8. Deployment & Scaling

- **Edge:** Pi handles sensors + optional CV + streaming
- **Cloud/Server:** Backend 1 handles inventory, alerts, POS, and data aggregation
- **Streaming:** Backend 3 serves live video feed
- **CI/CD:** GitHub Actions for automated builds & deployments
- **Scalability:** Docker/K3s can be added to scale multiple fridges / warehouses

---

9. Optional Features / Stretch Goals

- Motion detection with snapshots or recording
- Deep learning-based vial detection for robust CV
- Multi-fridge orchestration in cloud
- Predictive analytics for stock depletion / temperature trends

---

10. Summary

This project demonstrates: - Full-stack application design (React + Node + Python) - Edge computing and IoT integration - Computer vision & ML for real-world inventory tasks - Cloud-native architecture and streaming services - Alerts and monitoring for pharmaceutical storage compliance

It’s a comprehensive project showcasing development, DevOps, IoT, and ML skills in one solution.

End of Document