PROJECT WORK

SMART INVENTORY MANAGEMENT SYSTEM FOR SMALL WAREHOUSES

AIM:- The Smart Inventory Management System in a Cyber-Physical System (CPS) aims to revolutionize warehouse management by automating and optimizing inventory tracking and control. By integrating physical and computational components, the system enables real-time monitoring and control, allowing for swift responses to changing demands. Through advanced data analytics and machine learning, the system improves efficiency, accuracy, and decision-making, ultimately enhancing overall supply chain management. This harmonization of physical and digital elements creates a more agile, responsive, and intelligent inventory management system, capable of streamlining operations and driving business success.

PROBLEM STATEMENT:- "Inefficient and inaccurate inventory management in warehouses leads to stockouts, overstocking, and wasted resources, resulting in decreased productivity, increased costs, and reduced customer satisfaction. Current systems lack real-time visibility, automation, and data-driven insights, making it challenging to respond quickly to changing demands, manage complex supply chains, and optimize inventory levels. A Smart Inventory Management System in a Cyber-Physical System is needed to integrate physical and computational components, enable real-time monitoring and control, and leverage data analytics and machine learning to optimize inventory management and improve overall supply chain performance."

PROJECT DESIGN SPECIFICATION:-

System Requirements:

1. Real-time inventory tracking and monitoring
2. Automated inventory reporting and alerts
3. Data analytics and machine learning for demand forecasting and optimization
4. Integration with existing warehouse management systems (WMS) and enterprise resource planning (ERP) systems
5. Scalability and flexibility to accommodate changing inventory levels and product lines
6. User-friendly interface for warehouse staff and management
7. Cybersecurity measures to protect sensitive data and prevent unauthorized access

Hardware Components:

1. RFID tags or sensors for inventory tracking
2. IoT devices for real-time monitoring and data collection
3. Gateways for data transmission and communication
4. Servers for data storage and processing

Software Components:

1. Inventory management software with real-time tracking and reporting capabilities
2. Data analytics and machine learning algorithms for demand forecasting and optimization

3. Integration APIs for connecting with existing WMS and ERP systems
4. User interface software for warehouse staff and management

System Architecture:

1. Distributed architecture with IoT devices and gateways at the edge
2. Centralized data storage and processing in the cloud or on-premise servers
3. Real-time data transmission and communication between components

Performance Metrics:

1. Inventory accuracy and tracking efficiency
2. Response time for inventory queries and reports
3. Forecasting accuracy and optimization effectiveness
4. System uptime and reliability
5. User adoption and satisfaction rates.

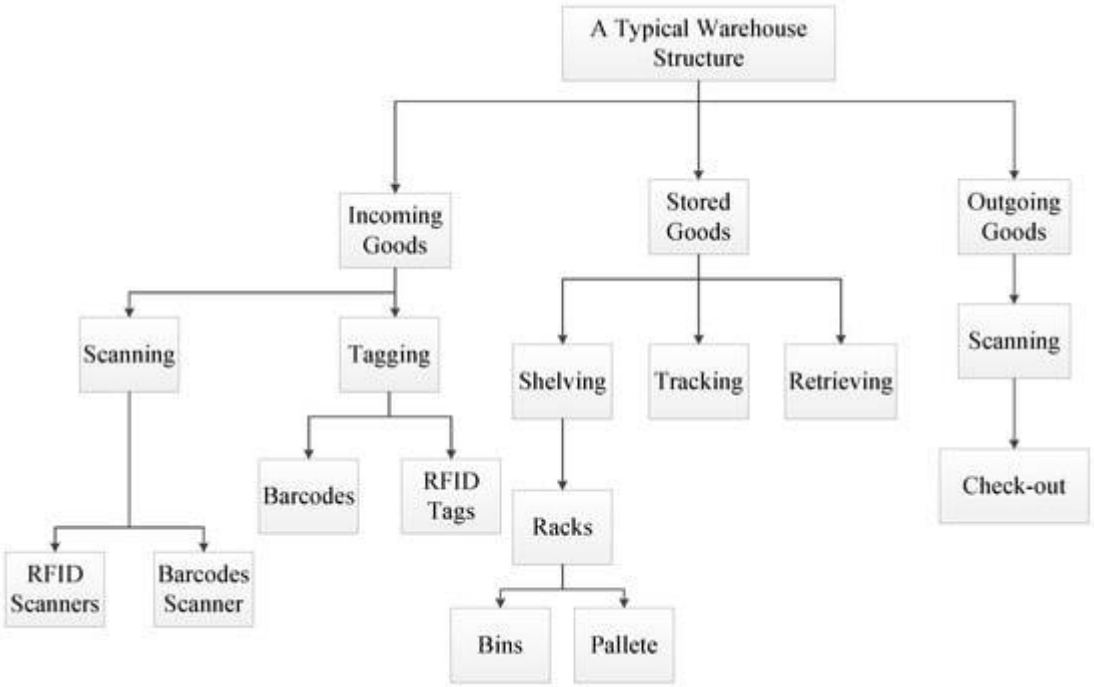
Security Requirements:

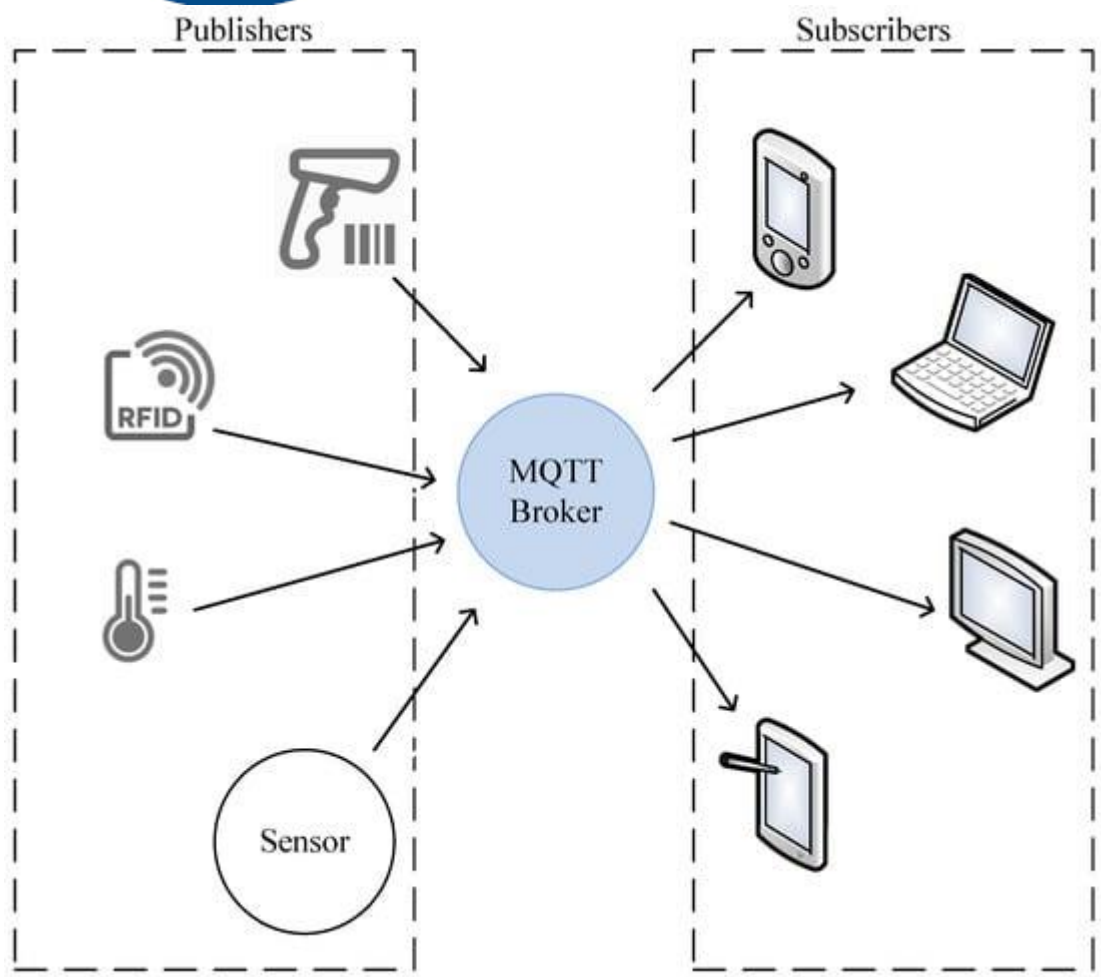
1. Authentication and authorization for user access
2. Data encryption for secure transmission and storage
3. Regular security audits and vulnerability assessments
4. Compliance with industry-specific regulations (e.g. GDPR, HIPAA)

Data Management:

1. Data warehousing for historical inventory data
2. Data analytics for trend analysis and forecasting
3. Data visualization for intuitive reporting and insights
4. Data backup and disaster recovery procedures

Flow chart:-




Publishers
Subscribers
RFID
MQTT
Broker
Sensor


Sensor
Controller
Actuator Block
Actuator Block
IMAGINED WITH AI

Components and working:-

1. RFID Tags: Attached to inventory items, these tags store unique identification information.

2. RFID Readers: Installed throughout the warehouse, these readers detect and read RFID tags.

3. IoT Devices: Sensors and gateways that transmit data to the server.

4. Server: Processes and stores data from IoT devices.

5. Database: Stores inventory information, tracking history, and analytics.

6. Warehouse Management System (WMS): Software that manages inventory, orders, and shipments.

7. Enterprise Resource Planning (ERP) System: Software that integrates WMS with business operations.

8. Barcode Scanners: Used for manual inventory tracking and verification.

9. Mobile Devices: Used for remote monitoring and management.

Working:

1. RFID tags are attached to inventory items.

2. RFID readers detect and read tags, transmitting data to IoT devices.

3. IoT devices transmit data to the server.

4. Server processes data, updates database, and triggers alerts.

5. WMS and ERP systems integrate data for inventory management and business operations.

6. Barcode scanners verify inventory information.

7. Mobile devices access data for remote monitoring.

Functions:

1. Inventory Tracking: Real-time tracking of inventory levels and locations.
2. Automated Reporting: Automated generation of inventory reports.
3. Alerts and Notifications: Alerts for low stock, expired items, or discrepancies.
4. Demand Forecasting: Predictive analytics for inventory replenishment.
5. Inventory Optimization: Optimization of inventory levels and storage.
6. Supply Chain Visibility: Real-time visibility into inventory movement.
7. Remote Monitoring: Remote access to inventory information.

This integrated system enables efficient inventory management, reduces errors, and improves supply chain operations

ASSEMBLING HARDWARE COMPONENTS AND CODING:-

Hardware Assembly:

1. Install RFID readers and antennas in the warehouse.
2. Connect RFID readers to IoT devices (e.g., gateways, routers).
3. Install sensors (e.g., temperature, humidity) and connect to IoT devices.
4. Set up servers and data storage devices.
5. Connect barcode scanners and mobile devices to the system.
6. Ensure power supply and backup systems are in place.

Software and Coding:

1. Choose a programming language (e.g., Python, Java, C++).
2. Develop a database schema for inventory management.
3. Write code for:
   - RFID tag reading and data transmission.
   - IoT device data processing and transmission.
   - Server-side data processing and analytics.
   - WMS and ERP system integration.
   - Mobile app development for remote monitoring.
4. Implement algorithms for:
   - Inventory tracking and reporting.
   - Demand forecasting and optimization.
   - Alerts and notifications.
5. Integrate with existing WMS and ERP systems.
6. Test and debug the system.

Example Code Snippets:

```
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
import paho.mqtt.client as mqtt
import sqlite3
from flask import Flask, request
```

```
import pyusb

# RFID Reader
reader = SimpleMFRC522()

# IoT Device (MQTT)
client = mqtt.Client()
client.connect("(link unavailable)", 1883)
client.loop_start()

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

# Server (Flask)
app = Flask(__name__)

# Barcode Scanner
scanner = pyusb.core.find(idVendor=0x1234, idProduct=0x5678)

# Function to read RFID tag
def read_rfid_tag():
    try:
        id, text = reader.read()
```

```
        return id, text
    finally:
        GPIO.cleanup()

# Function to publish data to MQTT topic
def publish_data(topic, data):
    client.publish(topic, data)

# Function to execute SQL query
def execute_query(query):
    cursor.execute(query)
    conn.commit()

# Function to handle HTTP requests
@app.route("/inventory", methods=["GET", "POST"])
def handle_request():
    if request.method == "POST":
        data = request.json
        execute_query("INSERT INTO inventory VALUES (?, ?)", (data["id"], data["quantity"]))
    return "OK"

# Function to read barcode data
def read_barcode_data():
    data = scanner.read(1024)
```

```
    return data.decode("utf-8")

# Main loop
while True:
    # Read RFID tag
    id, text = read_rfid_tag()
    print(f"RFID Tag: {id}, {text}")

    # Publish data to MQTT topic
    publish_data("inventory/rfid", f"{id},{text}")

    # Read barcode data
    barcode_data = read_barcode_data()
    print(f"Barcode Data: {barcode_data}")

    # Handle HTTP requests
    app.run(debug=True)
```