SIT314 – Distinction Task 4.2D

Project Status Report: Smart Warehouse Inventory & Delivery Management System

1. Introduction

The project requires the design and implementation of an IoT-based scalable solution that demonstrates event-driven processing, cloud integration, and the ability to handle dynamic workloads. The chosen project is the Smart Warehouse Inventory and Delivery Management System, which will monitor warehouse inventory levels through simulated sensors, process events in real-time using Node-RED, and update an inventory microservice that manages stock data. In its final form, the project will integrate with cloud services (AWS DynamoDB and EC2), extend into order and delivery microservices, and demonstrate scalability through testing and simulation. This report provides a status update on the project's current progress, highlights completed components, outlines challenges encountered, and specifies pending work required to reach the final deliverable.

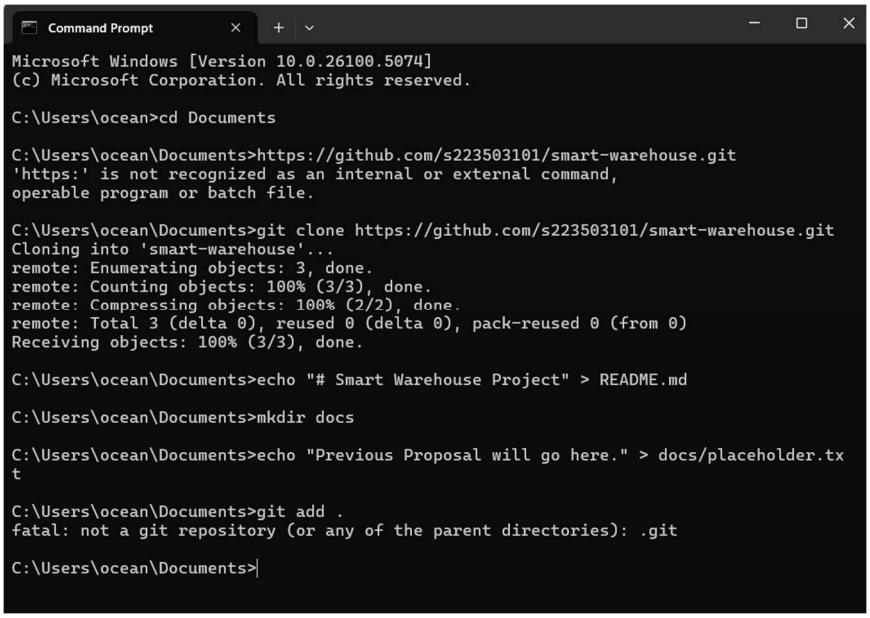
2. Project Objectives

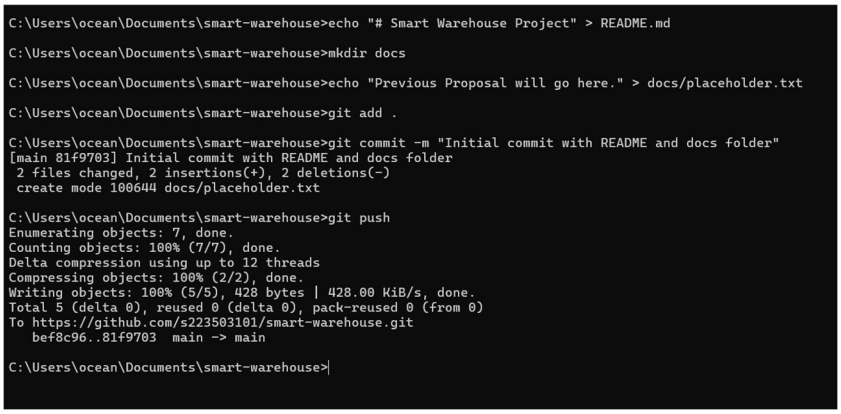
- Sensor Simulation: Implement simulated IoT sensors to generate inventory data.
- Event Processing: Use Node-RED to apply rules, thresholds, and trigger reorder events.
- Inventory Management: Build a microservice to store, update, and retrieve inventory records.
- Cloud Integration (pending): Deploy to AWS infrastructure, using DynamoDB for persistence and EC2/Lambda for scalability.
- Extended Microservices (pending): Add services for order handling and delivery scheduling.
- Testing and Scalability (pending): Conduct functional, integration, and load testing with multiple simulated sensors.

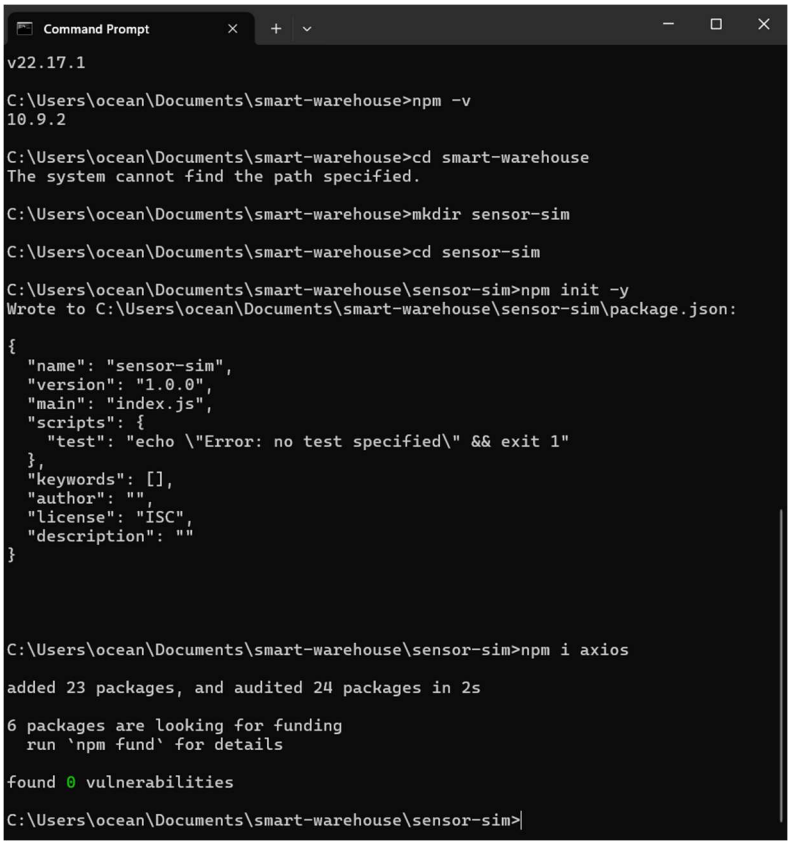
3. Progress Achieved So Far

3.1 Repository Setup

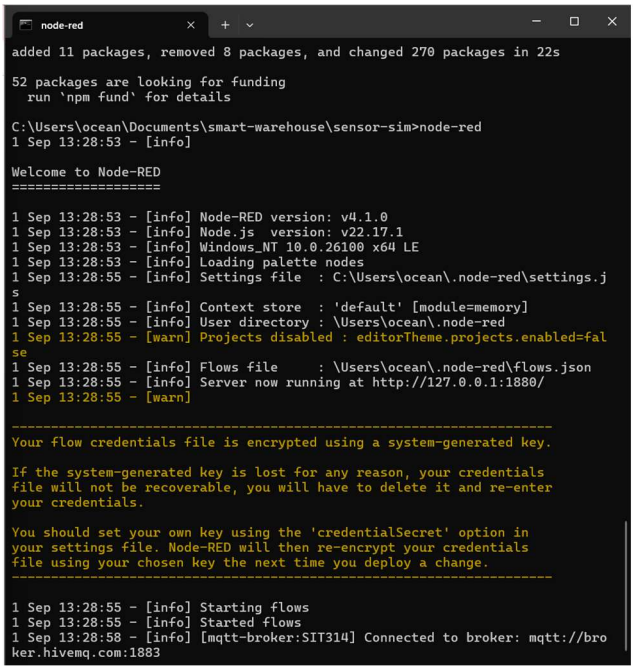
A structured GitHub repository (*smart-warehouse*) has been created and organised into clear modules. The repository contains: sensor-sim/ (Node.js simulator), services/inventory/ (inventory microservice), flows/ (Node-RED flow export), docs/ (documentation files), and README.md (project overview). All updates have been committed and pushed to GitHub, ensuring version control.

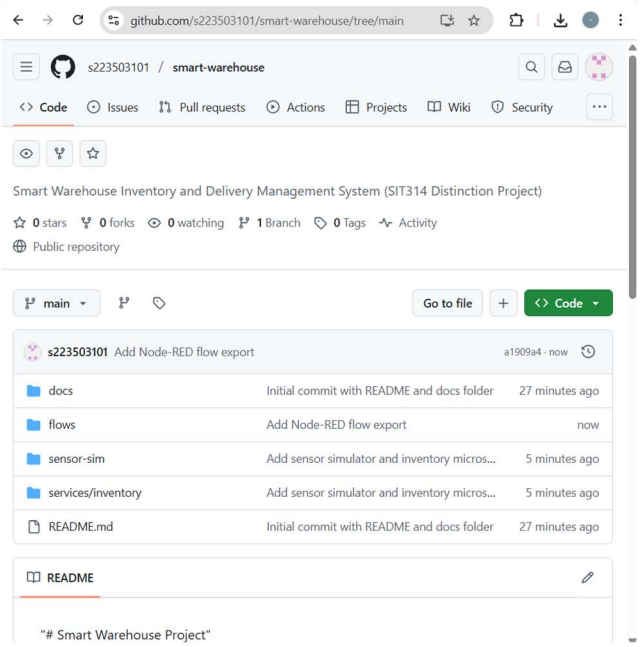

Command Prompt
Microsoft Windows [Version 10.0.26100.5074]
(c) Microsoft Corporation. All rights reserved.
C:\Users\ocean>cd Documents
C:\Users\ocean\Documents>https://github.com/s223503101/smart-warehouse.git
'https:' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\ocean\Documents>git clone https://github.com/s223503101/smart-warehouse.git
Cloning into 'smart-warehouse'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
C:\Users\ocean\Documents>echo "# Smart Warehouse Project" > README.md
C:\Users\ocean\Documents>mkdir docs
C:\Users\ocean\Documents>echo "Previous Proposal will go here." > docs/placeholder.tx
t
C:\Users\ocean\Documents>git add .
fatal: not a git repository (or any of the parent directories): .git
C:\Users\ocean\Documents>


C:\Users\ocean\Documents\smart-warehouse>echo "# Smart Warehouse Project" > README.md
C:\Users\ocean\Documents\smart-warehouse>mkdir docs
C:\Users\ocean\Documents\smart-warehouse>echo "Previous Proposal will go here." > docs/placeholder.txt
C:\Users\ocean\Documents\smart-warehouse>git add .
C:\Users\ocean\Documents\smart-warehouse>git commit -m "Initial commit with README and docs folder"
[main 81f9703] Initial commit with README and docs folder
2 files changed, 2 insertions(+), 2 deletions(-)
create mode 100644 docs/placeholder.txt
C:\Users\ocean\Documents\smart-warehouse>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 428 bytes | 428.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/s223503101/smart-warehouse.git
bef8c96..81f9703 main -> main
C:\Users\ocean\Documents\smart-warehouse>


Command Prompt
v22.17.1
C:\Users\ocean\Documents\smart-warehouse>npm -v
10.9.2
C:\Users\ocean\Documents\smart-warehouse>cd smart-warehouse
The system cannot find the path specified.
C:\Users\ocean\Documents\smart-warehouse>mkdir sensor-sim
C:\Users\ocean\Documents\smart-warehouse>cd sensor-sim
C:\Users\ocean\Documents\smart-warehouse\sensor-sim>npm init -y
Wrote to C:\Users\ocean\Documents\smart-warehouse\sensor-sim\package.json:
{
"name": "sensor-sim",
"version": "1.0.0",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC",
"description": ""
}
C:\Users\ocean\Documents\smart-warehouse\sensor-sim>npm i axios
added 23 packages, and audited 24 packages in 2s
6 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
C:\Users\ocean\Documents\smart-warehouse\sensor-sim>


Command Prompt - node ser
C:\Users\ocean\Documents\smart-warehouse>cd sensor-sim
C:\Users\ocean\Documents\smart-warehouse\sensor-sim>npm init -y
Wrote to C:\Users\ocean\Documents\smart-warehouse\sensor-sim\package.json:
{
"name": "sensor-sim",
"version": "1.0.0",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC",
"description": ""
}
C:\Users\ocean\Documents\smart-warehouse\sensor-sim>npm i axios
added 23 packages, and audited 24 packages in 2s
6 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
C:\Users\ocean\Documents\smart-warehouse\sensor-sim>notepad sensor.js
C:\Users\ocean\Documents\smart-warehouse\sensor-sim>node sensor.js
Sensor simulator started.
Posting to: http://127.0.0.1:1880/sensor every 5s
[POST ERR] connect ECONNREFUSED 127.0.0.1:1880 {"item_id":"SKU-123","stock_l
evel":0,"ts":"2025-09-01T03:26:06.076Z"}
[POST ERR] connect ECONNREFUSED 127.0.0.1:1880 {"item_id":"SKU-123","stock_l
evel":7,"ts":"2025-09-01T03:26:11.134Z"}



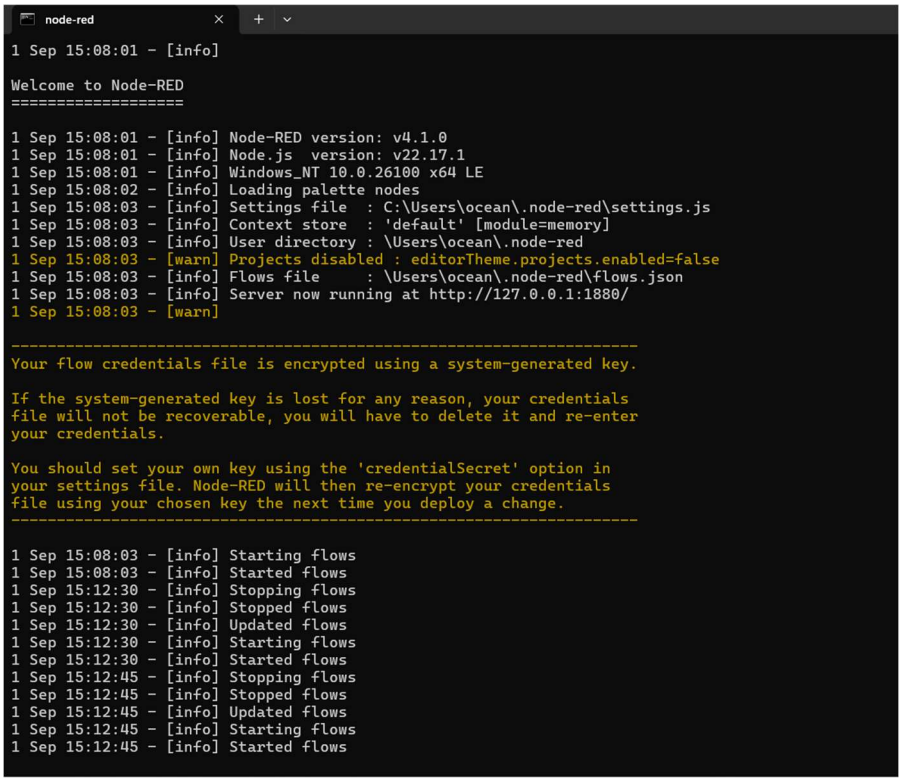


3.2 Sensor Simulator

A Node.js script (sensor.js) was developed to simulate warehouse stock levels. It generates random stock_level values every five seconds and POSTs them to Node-RED at /sensor. The simulator was tested successfully with console logs confirming transmission, e.g. [POST 200] {...}.

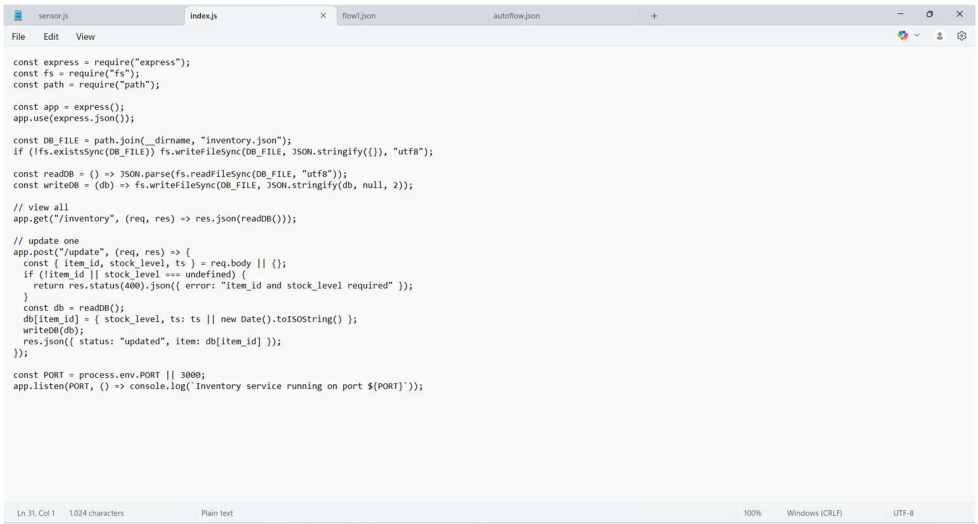
3.3 Node-RED Flow

A Node-RED flow was designed to process incoming sensor data. It parses JSON payloads, applies a threshold check for stock_level < 10, and returns structured JSON responses. Debug nodes were added to visualise incoming messages and inventory responses. The flow was exported (flow1.json) and stored in the repository.



3.4 Inventory Microservice

An Express.js microservice was implemented with two endpoints: POST /update (updates inventory.json) and GET /inventory (retrieves inventory data). The service was validated using curl and browser testing, confirming correct updates and retrieval.



3.5 End-to-End Integration

Node-RED and the inventory service were successfully integrated. When low stock events are detected, Node-RED triggers a POST /update call. Testing confirmed that stock_level=5 triggered a reorder and updated the DB, while stock_level=15 did not update the DB. Verification through GET /inventory confirmed correct persistence of values.

4. Challenges and Solutions

Challenge	Impact	Solution Implemented
Payload parsing in Node-RED – sensor data arrived as strings	Threshold checks failed as stock_level was not numeric	Added JSON node to convert to objects; updated function logic
Flow import/export issues – difficulties using Admin API	Slowed testing and configuration	Used Node-RED UI import/export; committed flow1.json to GitHub
Service integration – Inventory API triggered incorrectly	Database updated regardless of stock level	Added Switch node to ensure /update triggers only when lowStock==true
Debug visibility – unclear data flow	Delayed troubleshooting	Added Debug nodes to monitor incoming messages and inventory responses

5. Next Steps

Although the local prototype is functional, the following tasks are pending:

1. Cloud Deployment – Replace local JSON persistence with AWS DynamoDB and deploy services on AWS EC2.

2. Additional Microservices – Develop Order and Delivery services, integrating them with inventory.

3. Scalability and Testing – Simulate multiple sensors and workloads; conduct load and performance testing to validate system scalability.

4. Security Hardening – Configure IAM roles, API Gateway, and HTTPS endpoints to ensure secure cloud integration.

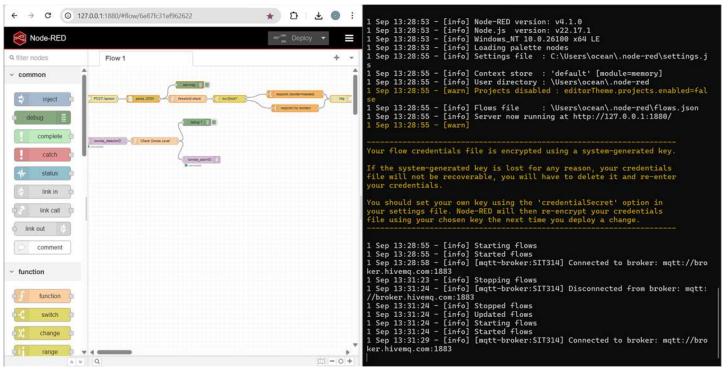
5. Final Deliverables – Produce the comprehensive final report and record a demonstration video.

6. Conclusion

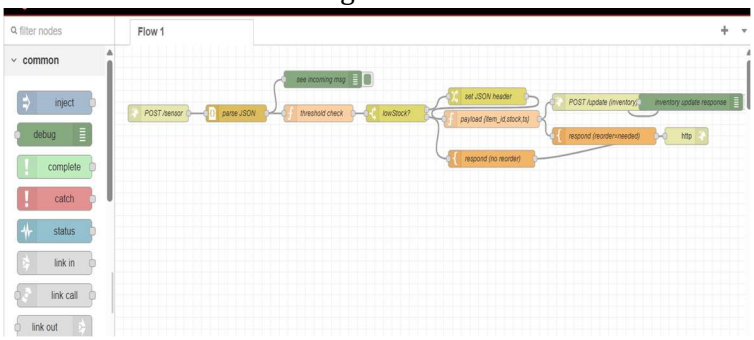
The project has reached a key milestone with a fully working local prototype. The complete cycle of sensor simulation → Node-RED event processing → inventory microservice update has been demonstrated. The foundation is strong, and the next steps will focus on cloud deployment, additional microservices, scalability testing, and final deliverables. The project is on track to meet the SIT314 distinction requirements.

Evidences:

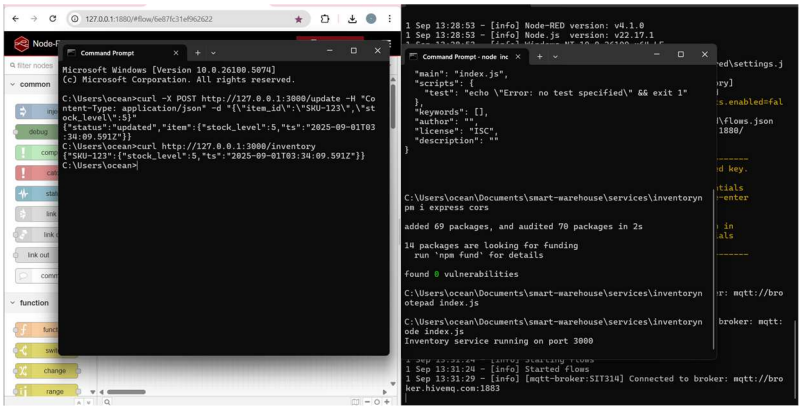
Node-RED Flow and Runtime Logs



Node-RED Event Processing Flow

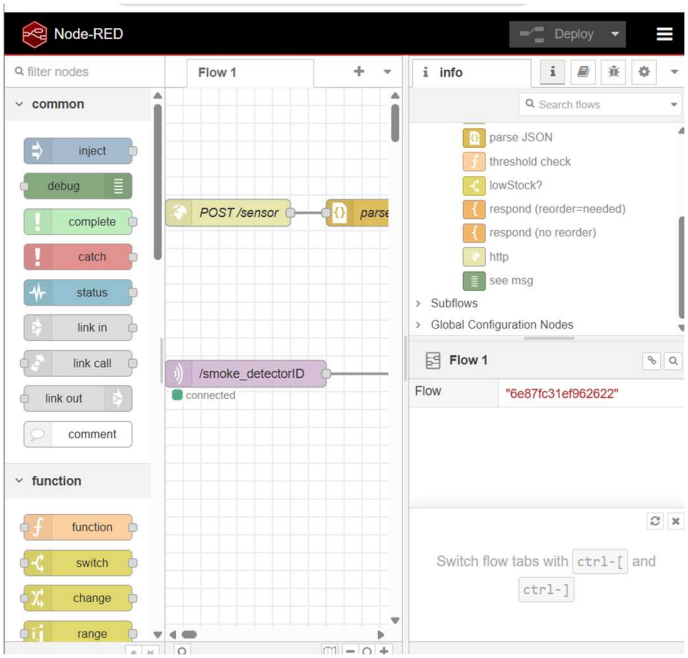


Inventory Microservice Update and Retrieval

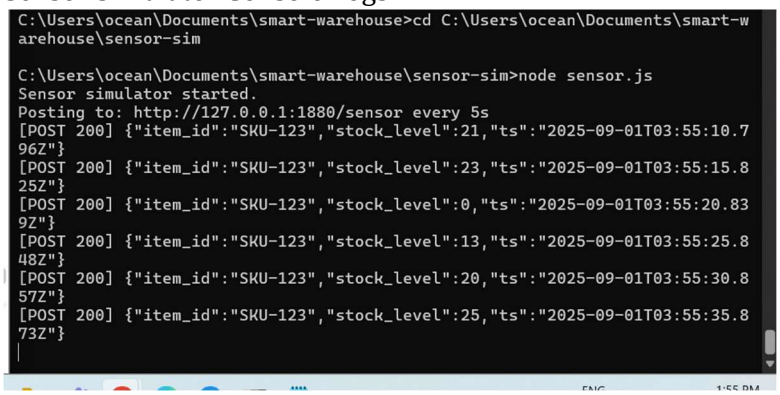


Inventory Management

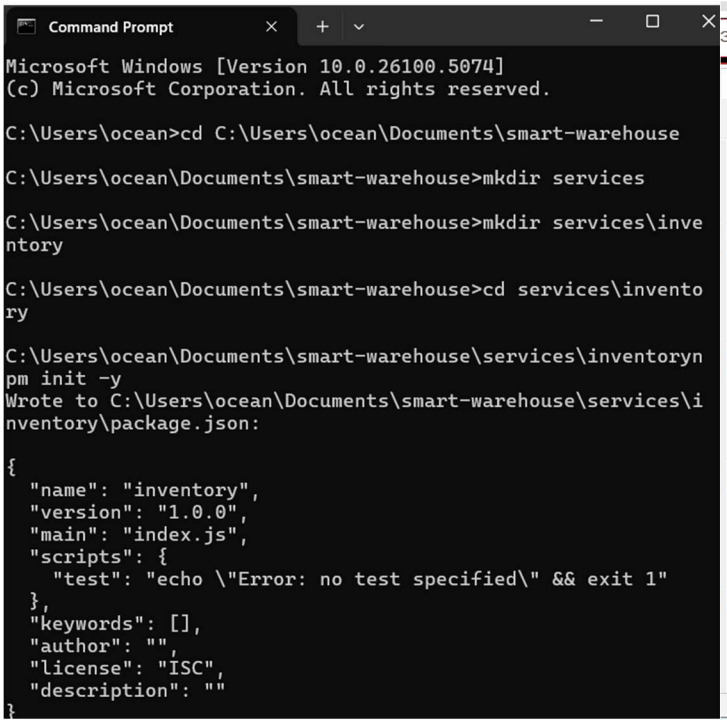
Node-RED Multi-Sensor Integration



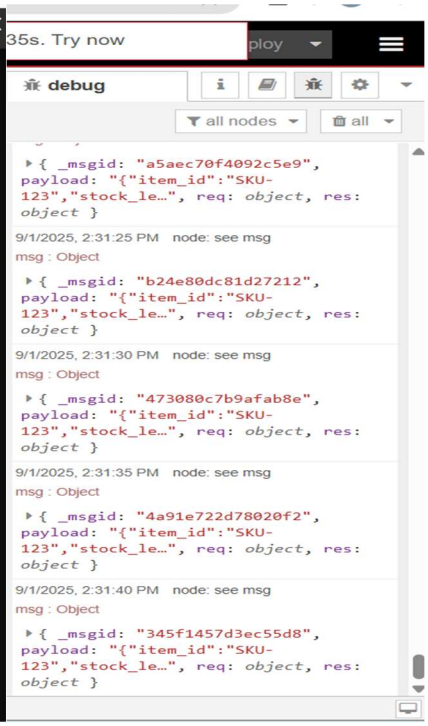
Sensor Simulator Console Logs



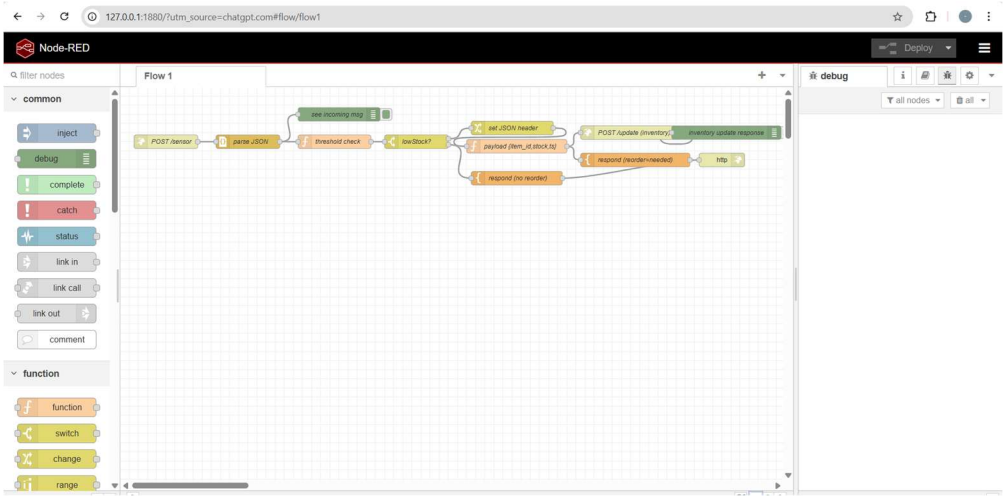
Inventory Microservice Setup



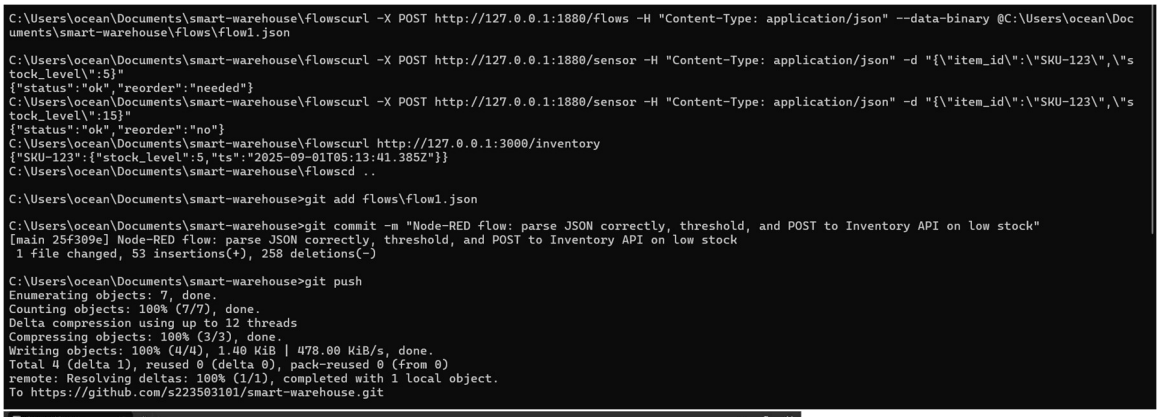
Node-RED Debug Output



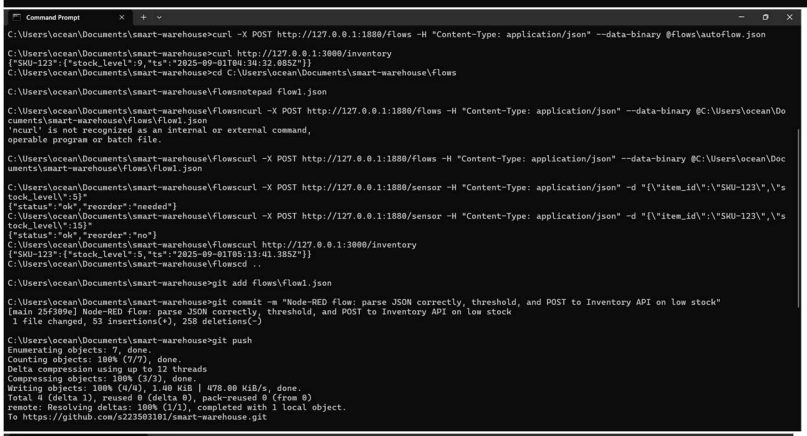
Node-RED End-to-End Event Flow



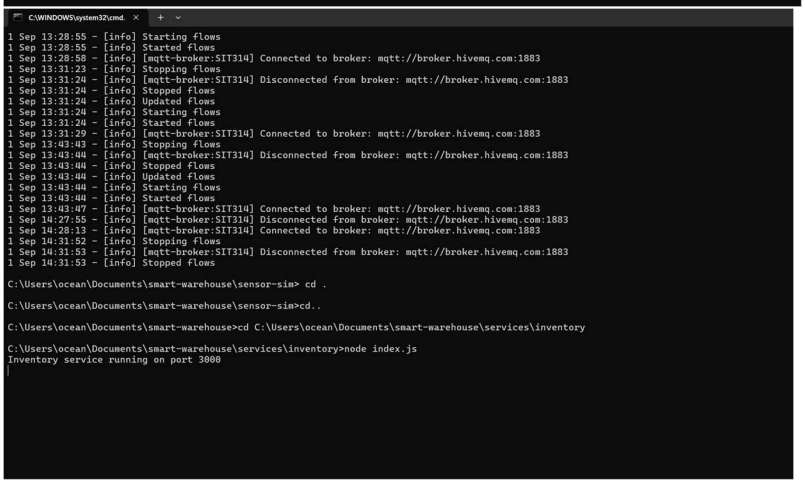
Evidences confirm correct flow execution, Git version control of flow1.json, and the inventory service running on port 3000, aligning with the Project Status objectives



```
C:\Users\ocean\Documents\smart-warehouse\flowscurl -X POST http://127.0.0.1:1880/flows -H "Content-Type: application/json" --data-binary @C:\Users\ocean\Doc
uments\smart-warehouse\flows\flow1.json

C:\Users\ocean\Documents\smart-warehouse\flowscurl -X POST http://127.0.0.1:1880/sensor -H "Content-Type: application/json" -d "{\"item_id\":\"SKU-123\",\"s
tock_level\":5}"
{"status":"ok","reorder":"needed"}
C:\Users\ocean\Documents\smart-warehouse\flowscurl -X POST http://127.0.0.1:1880/sensor -H "Content-Type: application/json" -d "{\"item_id\":\"SKU-123\",\"s
tock_level\":15}"
{"status":"ok","reorder":"no"}
C:\Users\ocean\Documents\smart-warehouse\flowscurl http://127.0.0.1:3000/inventory
{"SKU-123":{"stock_level":5,"ts":"2025-09-01T05:13:41.385Z"}}
C:\Users\ocean\Documents\smart-warehouse\flowscd ..

C:\Users\ocean\Documents\smart-warehouse>git add flows\flow1.json

C:\Users\ocean\Documents\smart-warehouse>git commit -m "Node-RED flow: parse JSON correctly, threshold, and POST to Inventory API on low stock"
[main 25f309e] Node-RED flow: parse JSON correctly, threshold, and POST to Inventory API on low stock
 1 file changed, 53 insertions(+), 258 deletions(-)

C:\Users\ocean\Documents\smart-warehouse>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 1.40 KiB | 478.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/s223503101/smart-warehouse.git
```



```
Command Prompt

C:\Users\ocean\Documents\smart-warehouse>curl -X POST http://127.0.0.1:1880/flows -H "Content-Type: application/json" --data-binary @flows\autoflow.json

C:\Users\ocean\Documents\smart-warehouse>curl http://127.0.0.1:3000/inventory
{"SKU-123":{"stock_level":9,"ts":"2025-09-01T04:34:32.085Z"}}
C:\Users\ocean\Documents\smart-warehouse>cd C:\Users\ocean\Documents\smart-warehouse\flows

C:\Users\ocean\Documents\smart-warehouse\flowsnotepad flow1.json

C:\Users\ocean\Documents\smart-warehouse\flowsncurl -X POST http://127.0.0.1:1880/flows -H "Content-Type: application/json" --data-binary @C:\Users\ocean\Do
cuments\smart-warehouse\flows\flow1.json
'ncurl' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\ocean\Documents\smart-warehouse\Flowscurl -X POST http://127.0.0.1:1880/flows -H "Content-Type: application/json" --data-binary @C:\Users\ocean\Doc
uments\smart-warehouse\flows\flow1.json

C:\Users\ocean\Documents\smart-warehouse\Flowscurl -X POST http://127.0.0.1:1880/sensor -H "Content-Type: application/json" -d "{\"item_id\":\"SKU-123\",\"s
tock_level\":5}"
{"status":"ok","reorder":"needed"}
C:\Users\ocean\Documents\smart-warehouse\flowscurl -X POST http://127.0.0.1:1880/sensor -H "Content-Type: application/json" -d "{\"item_id\":\"SKU-123\",\"s
tock_level\":15}"
{"status":"ok","reorder":"no"}
C:\Users\ocean\Documents\smart-warehouse\flowscurl http://127.0.0.1:3000/inventory
{"SKU-123":{"stock_level":5,"ts":"2025-09-01T05:13:41.385Z"}}
C:\Users\ocean\Documents\smart-warehouse\flowscd ..

C:\Users\ocean\Documents\smart-warehouse>git add flows\flow1.json

C:\Users\ocean\Documents\smart-warehouse>git commit -m "Node-RED flow: parse JSON correctly, threshold, and POST to Inventory API on low stock"
[main 25f309e] Node-RED flow: parse JSON correctly, threshold, and POST to Inventory API on low stock
 1 file changed, 53 insertions(+), 258 deletions(-)

C:\Users\ocean\Documents\smart-warehouse>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 1.40 KiB | 478.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/s223503101/smart-warehouse.git
```



```
C:\WINDOWS\system32\cmd.

1 Sep 13:28:55 - [info] Starting flows
1 Sep 13:28:55 - [info] Started flows
1 Sep 13:28:58 - [info] [mqtt-broker:SIT314] Connected to broker: mqtt://broker.hivemq.com:1883
1 Sep 13:31:23 - [info] Stopping flows
1 Sep 13:31:24 - [info] [mqtt-broker:SIT314] Disconnected from broker: mqtt://broker.hivemq.com:1883
1 Sep 13:31:24 - [info] Stopped flows
1 Sep 13:31:24 - [info] Updated flows
1 Sep 13:31:24 - [info] Starting flows
1 Sep 13:31:24 - [info] Started flows
1 Sep 13:31:29 - [info] [mqtt-broker:SIT314] Connected to broker: mqtt://broker.hivemq.com:1883
1 Sep 13:43:43 - [info] Stopping flows
1 Sep 13:43:44 - [info] [mqtt-broker:SIT314] Disconnected from broker: mqtt://broker.hivemq.com:1883
1 Sep 13:43:44 - [info] Stopped flows
1 Sep 13:43:44 - [info] Updated flows
1 Sep 13:43:44 - [info] Starting flows
1 Sep 13:43:44 - [info] Started flows
1 Sep 13:43:47 - [info] [mqtt-broker:SIT314] Connected to broker: mqtt://broker.hivemq.com:1883
1 Sep 14:27:55 - [info] [mqtt-broker:SIT314] Disconnected from broker: mqtt://broker.hivemq.com:1883
1 Sep 14:28:13 - [info] [mqtt-broker:SIT314] Connected to broker: mqtt://broker.hivemq.com:1883
1 Sep 14:31:52 - [info] Stopping flows
1 Sep 14:31:53 - [info] [mqtt-broker:SIT314] Disconnected from broker: mqtt://broker.hivemq.com:1883
1 Sep 14:31:53 - [info] Stopped flows

C:\Users\ocean\Documents\smart-warehouse\sensor-sim> cd .

C:\Users\ocean\Documents\smart-warehouse\sensor-sim>cd..

C:\Users\ocean\Documents\smart-warehouse>cd C:\Users\ocean\Documents\smart-warehouse\services\inventory

C:\Users\ocean\Documents\smart-warehouse\services\inventory>node index.js
Inventory service running on port 3000
```