Predictive Inventory Agent (PIA) Using OpenWebUI + LangGraph + Deepseek R1 / LLaMa 3.3

Solution Overview

This document outlines the architecture and integration flow for building a Predictive Inventory Agent (PIA) using the following technologies:

- User Interface: OpenWebUI
- Workflow Orchestration: LangGraph
- LLM Model: Deepseek R1 or LLaMa 3.3
- Retrieval Framework: LangChain-powered RAG
- Vector Database: Milvus
- Source Database: PostgreSQL
- Embedding Model: BGE / Instructor-XL / OpenAI Embeddings

The goal is to enable intelligent, predictive inventory responses based on sales and stock data using Retrieval-Augmented Generation (RAG) and LLM-based reasoning.

Architecture Diagram (Logical Flow)

1. User Query Initiation: The user submits a predictive inventory query via OpenWebUI.
2. LangGraph Entry Node: Receives the query and initiates workflow.

3. RAG Retrieval Node: Fetches relevant knowledge using two methods:

   - Vector Search from Milvus (contextual inventory knowledge)

   - SQL Queries from PostgreSQL (live inventory and sales data)

4. Decision Node: Validates if sufficient information has been gathered; loops back if needed.

5. LLM Node: The LLM (Deepseek R1 or LLaMa 3.3) predicts future demand and suggests restocking decisions.

6. LangGraph Exit Node: Outputs the summarized, actionable response back to OpenWebUI.

Example PostgreSQL Schema

```
-- Inventory Table
CREATE TABLE inventory (
    item_id SERIAL PRIMARY KEY,
    item_name TEXT,
    stock_qty INTEGER,
    reorder_point INTEGER,
    lead_time_days INTEGER
);

-- Sales Table
CREATE TABLE sales (
    sale_id SERIAL PRIMARY KEY,
    item_id INTEGER REFERENCES inventory(item_id),
    quantity_sold INTEGER,
    sale_date DATE
);
```

Vector Database (Milvus)

- Documents embedded: Product descriptions, SKUs, inventory policies
- Embedding Model: OpenAI, BGE, or Instructor models
- Integration: LangChain VectorStore abstraction

LangGraph Workflow

Node Type	Description
Entry Node	Receives query from OpenWebUI
RAG Node	Performs hybrid retrieval from Milvus & PostgreSQL
LLM Node	Uses Deepseek/LLaMa to infer predictions
Decision Node	Validates and loops back if retrieval is insufficient
Exit Node	Sends final response to UI

Sample User Query & Response

Query: "Do we need to reorder Amoxicillin 250mg in the next 10 days?"

Response:

"Based on current stock of 180 units, average daily sales of 15 units, and a lead time of 7 days, reordering within 3 days is advisable to avoid stock-out."

Deployment Strategy

- Containers: All components dockerized
- Orchestration: Kubernetes for scalability (optional)
- LangGraph API: gRPC or REST
- Milvus & PostgreSQL: Stateful services
- LLM Hosting: Local Ollama or vLLM

Security & Governance

- Secure data access via API tokens and RBAC
- PostgreSQL access over SSL
- Audit trails of LLM responses (optional logging)

Conclusion

This integrated solution empowers business users to predict inventory requirements and automate restocking decisions

using conversational interfaces powered by LangGraph and state-of-the-art open-source LLMs. It combines real-time data, semantic search, and AI reasoning into a seamless, user-friendly experience.