**Smart Inventory Planning Assistant for Small Shops**
**Date:** December 2025

**Team:** Bhumika R, Prakruthi BR
**Course:** Bachelor of Computer Applications (BCA)

---

## Introduction:

Inventory planning is a very important part of running any business.
However, small shop owners usually manage stock manually, which is time-consuming and error prone.
Because of this, they often face over-stocking, under-stocking, and financial loss.

Our project solves this problem by providing a smart, AI-based inventory planning system that helps shop owners decide:

- What items to restock
- How much to restock
- When to restock

---

## Overview:

The Smart Inventory Planning Agent helps small shop owners decide what items to restock, how much to restock, and when to restock, based on current stock levels and sales rate. It uses a Planner Pattern and a local open-source LLM to analyze stock and sales data and explain decisions. The system works in both offline and online modes, making it simple and cost-effective.

---

## Problem Statement:

Small shop owners often manage inventory manually, which can lead to stock shortages or overstocking. Existing inventory software is often expensive, complex, or requires constant internet connectivity. There is a need for a lightweight, intelligent, offline-capable inventory assistant that is suitable for small businesses and simplifies inventory planning.

---

## Objectives:

- To design a smart inventory planning system for small shops
- To apply the Agentic AI Planner Pattern for decision-making
- To analyze stock levels and sales data for restocking decisions
- To use a local open-source LLM to explain inventory decisions

# Inputs and Outputs:

### Inputs

- Product list
- Current stock quantity
- Minimum stock threshold
- Daily or weekly sales data

**Example:**
Rice – Stock: 20, Min: 40
Sugar – Stock: 10, Min: 25
Oil – Stock: 15, Min: 30

### Outputs

**Level 1 – Immediate Attention (Low Stock)**
Sugar → Reorder 30 units
Rice → Reorder 50 units
*Explanation:* Stock is below minimum level and recent sales are high.

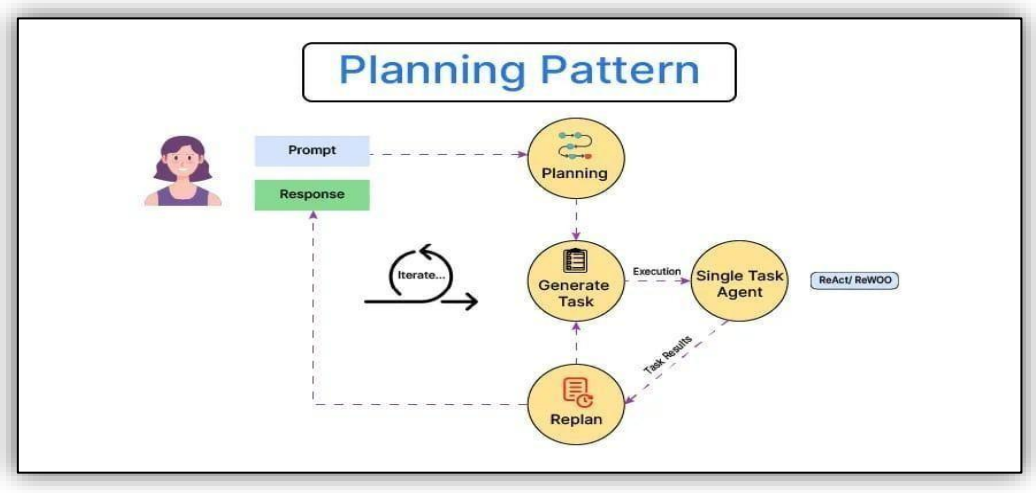
**Level 2 – Monitor Stock**
Oil → Stock sufficient, no reorder required

**Level 3 – Stable Items**
Items with steady stock and low sales – No action required.

---

# Design Overview

The system is designed using a Planner Pattern where inventory decisions are made stepbystep. It reads inventory and sales data, compares current stock with minimum thresholds, and plans restocking actions accordingly. A local open-source LLM is used to generate simple explanations for each decision. The final results are displayed to the user through a simple web interface.




:

**Task Explanations**

- **User Prompt:**
  The user provides inventory details such as product list, stock levels, and minimum thresholds to initiate the planning process.
- **Planning:**
  The LLM analyses the input data and determines an overall strategy for inventory restocking.
- **Generate Task:**
  The planner breaks the inventory goal into smaller ordered tasks like stock analysis and reorder planning.
- **Single Task Agent (Execution):**
  Each generated task is executed individually to analyse stock or calculate reorder quantities.
- **Replan:**
  The system updates or modifies the plan if new conditions such as stock changes or demand variations are detected.
- **Response:**
  The final optimized inventory restocking plan is generated and presented to the user.

## System Architecture:

Shop Owner (User)

↓

Planner Agent (Local LLM)

↓

Executor Agent (Stock Update)

↓

Feedback Agent (Analysis & Learning)

↓

Replanning (Local LLM)

The architecture consists of a user interface, backend server, database, and agent modules. Each agent performs a specific role in the inventory planning cycle.

---

### Technology Used

- Frontend: HTML, JavaScript
- Backend: Python (Flask)
- Database: SQLite / MySQL
- AI Logic: Rule-based planning Agentic AI
- Model: Large Language Model (LLM)

  Offline: Ollama (LLaMA/ Mistral – Open Source)

  Online: API – based LLM

---

## Planner Pattern Explanation:

The Planner Pattern ensures decisions are made logically and transparently:

1. Read inventory and sales data
2. Compare stock with minimum threshold
3. Plan reorder quantities
4. Explain decisions using LLM

This approach improves clarity and reliability of inventory decisions.

## Database Design / Schema:

### Table: Products

| Field | Type | Description |
|---|---|---|
| product_id | INT (PK) | Unique product ID |
| product_name | VARCHAR | Product name |
| category | VARCHAR | Category |
| min_stock | INT | Minimum stock |

**Table 2: Inventory**

| Field Name | Data Type | Description |
|---|---|---|
| inventory_id | INT (PK) | Inventory record ID |
| product_id | INT (FK) | Linked product ID |
| quantity | INT | Current stock quantity |

**Table 3: Sales**

| Field Name | Data Type | Description |
|---|---|---|
| sale_id | INT (PK) | Sales record ID |
| product_id | INT (FK) | Product sold |
| quantity_sold | INT | Quantity sold |
| sale_date | DATE | Date of sale |

**Example Scenario:**

Consider a grocery shop selling rice:

- Product: Rice
- Minimum stock level: 40 units
- Current stock: 20 units
- Average daily sales: 10 units

The Planner Agent (LLM) detects that stock is below the minimum threshold and plans a reorder of 50 units. The Executor Agent updates the inventory after

purchase. The Feedback Agent later observes high sales velocity and recommends increasing the minimum stock for future planning.

---

## Alternative Design Pattern:

## Reflection Pattern:

The Reflective Pattern is an Agentic AI design pattern where the system continuously evaluates its own behaviour and learns from previous actions to improve future performance. Unlike planning-based systems that focus on future steps, reflective systems focus on selfevaluation and adaptation. This pattern is inspired by human learning, where mistakes and successes are analysed to make better decisions in the future.

## Working Flow:

Input Data

↓

Action Execution

↓

Outcome Observation

↓

Reflection & Evaluation

↓

Knowledge Update

↓

(Loop continues)

This feedback loop allows the system to self-correct and evolve over time.

## Comparison with Planner Pattern:

| **Aspect** | **Reflective Pattern** | **Planner Pattern** |
|---|---|---|
| Learning style | Past-based | Future-based |
| Decision timing | After action | Before action |
| Planning steps | No | Yes |
| Best for | Improvement | Prediction & planning |

---

## Roles and Responsibilities:

**Member 1(Prakruthi BR) – Backend & Agent Logic Developer**

- Problem definition and requirement analysis
- Implementation of Planner Agent and Executor Agent
- Integration of local LLM with backend
- API development using Python and Flask

**Member 2(Bhumika R) – Database, UI & Analysis Developer**

- Design and implementation of database schema
- Development of Feedback (Reflective) Agent
- Frontend user interface design
- Documentation, testing, and GitHub management

---

## How Does Our Prototype Work?

This prototype is designed to help small shop owners decide when to reorder products and how much to order, using Agentic AI with a Planner pattern.

---

Step-by-Step Working of the Prototype

1️⃣ User Opens the Website (UI Layer)

- The shop owner opens the Smart Inventory Planner web page.
- The interface shows:
    - Product name
    - Current stock
    - Weekly sales
    - Buttons for:

        - Check Minimum Stock
        - Generate Inventory Plan
    - Option to choose Online LLM or Offline LLM

This makes the system easy to use for non-technical users.

---

2️⃣ Check Minimum Stock (Database Agent)

- When the user enters a product name (e.g., *Rice*) and clicks Check Minimum Stock:
    - The system checks the inventory database.
    - It finds the predefined minimum stock level and unit (kg, packets, liters, etc.).
- The result is shown as:

*Minimum stock required: 40 kg*

This helps the user understand safe stock levels before analysis.

---

3️⃣ User Enters Stock and Sales (Flexible Input)

- The shop owner enters:
    - Current stock (e.g., 20 kg)
    - Weekly sales (e.g., 15 kg)
- The system automatically:
    - Extracts the number
    - Identifies the unit
- This allows natural input without forcing technical formats.

---

4️⃣ Planner Agent Makes the Decision (Core Logic)

- The Planner Agent compares:

    - Current stock
    - Minimum required stock
    - Weekly sales rate
- Based on these values, it decides:
    - Reorder → stock is insufficient
    - No Reorder → stock is sufficient
- It also calculates the suggested order quantity.

This agent behaves like an experienced store manager.

---

5️⃣ LLM Agent Explains the Decision (Human-Friendly AI)

- The decision is sent to the LLM Agent.
- Based on the selected mode:
    - Online Mode → GPT-3.5-Turbo (OpenAI)
    - Offline Mode → Mistral (Ollama)
- The LLM generates explanations in:
    - Simple English
    - Numbered points
    - Practical shop-friendly language

Example explanation:

1. Your current stock is below the safe limit.
2. This product sells regularly every week.
3. If you delay ordering, customers may face shortage.
4. Ordering now ensures smooth sales and profit.

This builds trust in the AI decision.

---

6️⃣ User Action (Executor Agent)

The shop owner chooses one action:

- Accept
- Modify
- Ignore

Each action is handled by the Executor Agent.

---

7️⃣ Action Result Page (Feedback Agent)

Based on the selected action:

- The system shows a color-coded confirmation page:

| Action | Colour | Meaning |
|---|---|---|
| Accept | 🟢 Green | Decision accepted |
| Modify | 🟡 Yellow | Quantity adjusted |
| Ignore | 🔴 Red | Decision skipped |

Example message:

Action Accepted
You decided to order 25 kg of Rice.
This will help avoid shortages and keep customers satisfied

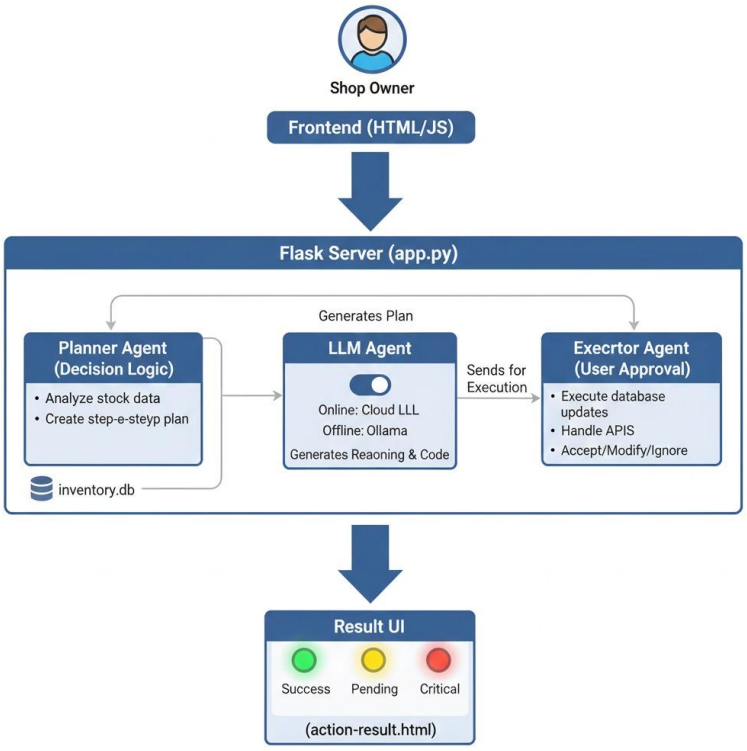
---

8️⃣ Replanning (Optional Replanner Agent)

- If the user modifies or ignores:

- The Replanner Agent adjusts future suggestions.
- The system learns from user behaviour.

- This shows Agentic AI behaviour




---

## Conclusion:

The Smart Inventory Planning Agent Using Planner Pattern successfully demonstrates how Agentic AI concepts can be applied to solve real-world inventory management problems. By using a Planner Pattern, the system efficiently breaks down the inventory planning goal into ordered tasks such as stock analysis, demand evaluation, and reorder decision-making. The integration of a Large Language Model (LLM) as the brain enables intelligent reasoning and adaptive planning, while support for both online and offline modes make the system practical and cost-effective. Overall, this project improves inventory decision accuracy and provides a strong foundation for future enhancements in intelligent planning systems.

---

## Source Code

## GitHub repo link:

Prakruthi BR: https://github.com/prakruthidevanga/Agentic_AI_Projects

Bhumika R: https://github.com/Bhumikaravi/Agentic_AI_Project