Multi-Agent Smart Supply Chain Optimizer

Introduction:

Modern supply chains are complex systems that involve real-time decision-making across multiple domains such as inventory management, supplier coordination, route optimization, and risk management. Traditional single-agent or rule-based systems are not flexible enough to handle these dynamic trade-offs in real time.

To overcome these challenges, the **Multi-Agent Smart Supply Chain Optimizer** uses a **multi-agent architecture** where specialized agents collaborate, communicate, and reason together to make optimal supply chain decisions. Each agent has a well-defined responsibility, individual memory, and participates in bidirectional communication through a shared scratchpad.

Project Objectives:

- Automate and optimize supply chain planning.
- Minimize costs while maintaining delivery reliability.
- Respond dynamically to disruptions such as supplier delays or weather events.
- Maintain transparency through structured decision logs.
- Demonstrate explainability through JSON logs and agent reasoning.
- Showcase agent collaboration, communication, and memory persistence.

System Architecture:

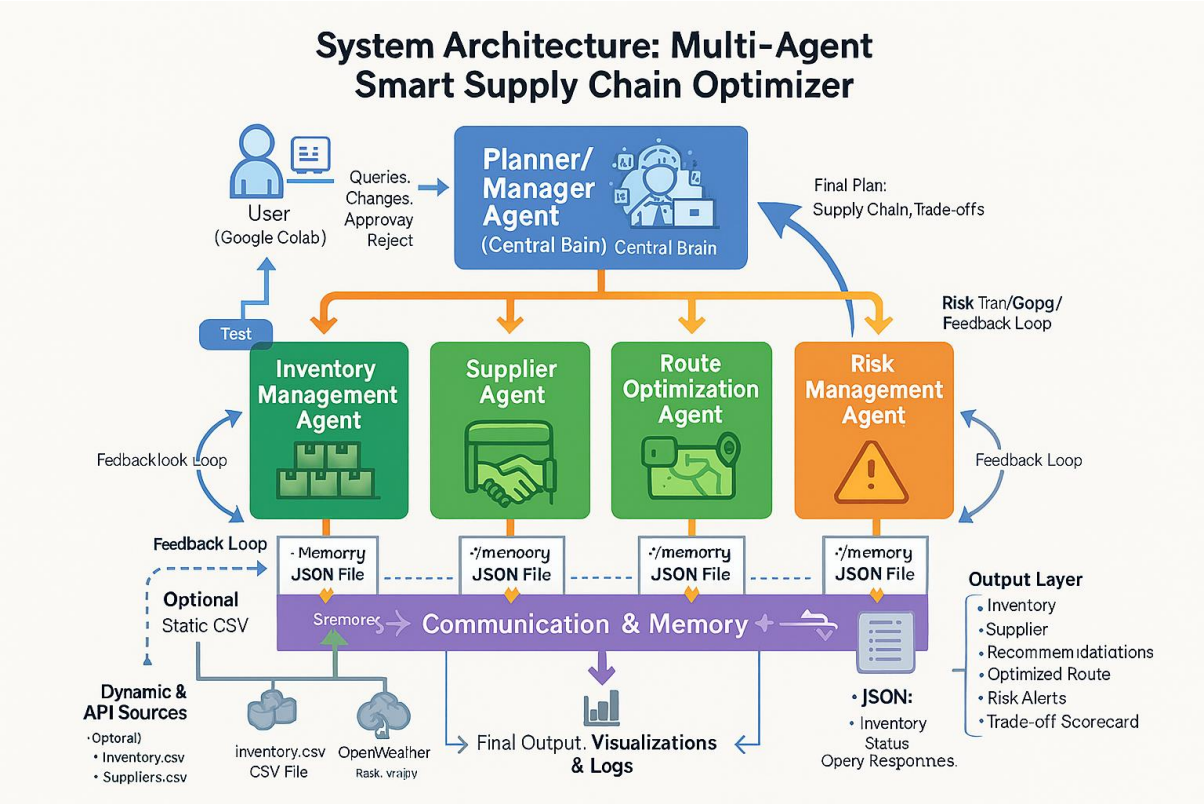
Architectural Overview:

The system is built on a **multi-agent framework** comprising five specialized agents:

- **Inventory Management Agent** – Monitors stock, predicts shortages.
- **Supplier Agent** – Evaluates suppliers based on price, reliability, and lead time.
- **Route Optimization Agent** – Plans efficient transport routes using OpenRouteService.
- **Risk Management Agent** – Monitors environmental, weather, and logistical risks.
- **Planner/Manager Agent** – Integrates all agent outputs, resolves conflicts, and produces the final supply chain plan.

Agents communicate **bidirectionally** through a shared communication bus (CommunicationBus) and store reasoning, conversations, and results in individual JSON memory files.

System Diagram (Description):



Workflow:

Step-by-Step Flow:

- **User Interaction**
  - The user communicates through the **Planner Agent**.
  - Example query: *"Analyze supply chain status and suggest improvements."*
- **Inventory Analysis**
  - The **Inventory Management Agent** reads inventory.csv, identifies low stock and critical items.
- **Supplier Analysis**
  - The **Supplier Agent** checks suppliers.csv for vendors matching the critical items and evaluates cost, lead time, and reliability.
- **Route Optimization**

- The **Route Optimization Agent** calculates the best route (using OpenRouteService or Haversine distance for static mode).

- **Risk Assessment**
  - The **Risk Management Agent** checks weather or disruption data (via OpenWeatherMap or static mock JSON).
- **Planner Integration**
  - The **Planner Agent** collects all outputs, resolves trade-offs (cost vs. speed vs. risk), and provides a **final supply chain plan**.
- **Logging & Memory Updates**
  - Each agent saves reasoning and conversations in its respective JSON memory.
  - Logs are also stored in a centralized session file (session_summary_<timestamp>.json).

Multi-Agent Roles and Memory Design:

Agent	Function	Individual Memory Example
Inventory Management Agent	Detects low stock and predicts shortages.	Stores historical reorder levels and item consumption trends.
Supplier Agent	Evaluates suppliers and selects best options.	Remembers vendor reliability and past negotiation data.
Route Optimization Agent	Determines optimal delivery routes.	Retains route efficiency scores and past delays
Risk Management Agent	Evaluates environmental and supply-chain risks.	Stores previous weather events, strike records, and mitigations.
Planner/Manager Agent	Integrates all outputs and resolves trade-offs.	Keeps records of decisions, trade-off weights, and user preferences.

Agent Communication and Collaboration:

All agents communicate through a shared CommunicationBus which maintains structured message logs.

Example JSON Log:


```
{
 "timestamp": "2025-10-05T14:30:55",
 "from_agent": "Supplier Agent",
 "to_agent": "Planner Agent",
 "message_type": "proposal",
 "payload": {
  "supplier_id": "SUP002",
  "lead_time_days": 3,
  "price_per_unit": 25.50
 },
 "reasoning_steps": [
  "Supplier reliability > 0.9",
  "Lead time < average threshold"
 ],
 "confidence": 0.88
}
```


Feedback Loop Example (System Dynamics):

Example Scenario:

Flood Disruption → Route Recalculation → Supplier Adjustment → Final Plan Update

- **Risk Agent** detects flood alert → Sends alert to Route Agent.
- **Route Agent** recalculates alternate safe route.
- **Inventory Agent** updates ETA for critical deliveries.
- **Supplier Agent** checks alternate vendors for delayed routes.
- **Planner Agent** re-balances the plan with all updated information.

Data Sources:

Source	Description	Type
`inventory.csv`	Item stock, usage rate, reorder thresholds, and coordinates.	Local CSV
`suppliers.csv`	Supplier info, price, lead time, reliability.	Local CSV
OpenRouteService API	Used for real route optimization.	Free API
OpenWeatherMap API	Used for risk/weather monitoring.	Free API (Limited)

Example Input Data:

inventory.csv (Sample)

item_id	item_name	current_stock	monthly_usage	reorder_point	lead_time_days	location_lat	location_lon
SKU001	Laptop	45	20	30	7	40.7128	-74.0060
SKU003	USB Cable	300	100	150	2	40.6892	-74.0445

suppliers.csv (Sample)

supplier_id	supplier_name	item_id	price_per_unit	lead_time_days	reliability_score
SUP001	TechCorp Solutions	SKU001	899.99	7	0.92
SUP002	Global Electronics	SKU003	12.99	2	0.95

Example Outputs:

Human-Readable Output:

Analysis Completed:

Restock 2 critical items immediately.

Recommended supplier: Global Electronics (Reliability 0.95)

Optimal route: 128.5 km, cost $122.30

Risk level: Medium — monitor conditions.

Final plan ensures on-time delivery and balanced trade-offs.

JSON Output Example:


```
{
 "inventory_status": {"SKU003": {"qty": 50, "predicted_shortage_days": 2}},
 "supplier_recommendation": {"supplier_id": "SUP002", "score": 0.91},
 "optimized_route": {"distance_km": 128.5, "duration_hr": 3.5, "cost": 122.3},
 "risk_alerts": [{"type": "flood", "impact": "medium"}],
 "final_plan": {"supplier": "SUP002", "route": "Route B", "action": "Expedite delivery"}
}
```


Explainability and Logs:

All agent actions are recorded as structured JSON logs:


```
{
 "agent": "Planner Manager Agent",
 "decision": "Chose Supplier B",
 "trade_offs": {
  "Supplier A": {"cost": 100, "reliability": 0.7},
  "Supplier B": {"cost": 105, "reliability": 0.9}
 },
 "reasoning": "Supplier B selected for reliability; +5% cost justified by reduced delay risk."
}
```

Memory Persistence:

- Each agent's memory is stored in a local JSON file.
- Example: inventory_management_agent_memory.json
- Stored data includes:
    - Conversations
    - Decisions and reasoning
    - Performance metrics
    - Learning patterns

Example Memory :


```
{
 "decisions": [
  {
   "context": "inventory_analysis",
   "decision": "completed_inventory_analysis",
   "confidence": 0.95
  }
 ],
 "conversations": [],
 "timestamp": "2025-10-05T15:00:00"
}
```


Implementation Details

- Platform: Google Colab
- Libraries: pandas, matplotlib, folium, openrouteservice, pyowm, langchain
- APIs: Gemini (reasoning), OpenRouteService (routing), OpenWeatherMap (risk)
- Input Data: inventory.csv, suppliers.csv
- Output: JSON logs, route maps, scorecards, and explainability reports.

Usage Manual

Prerequisites

- Google Colab environment
- Python 3.10 or higher
- Required libraries installed
- Optional API keys for OpenWeatherMap and OpenRouteService

Installation & Setup

- !pip install pandas matplotlib folium openrouteservice pyowm langchain

Execution Steps

- Upload 'inventory.csv' and 'suppliers.csv' to your Colab environment.
- Run all notebook cells sequentially.
- Interact with the Planner Agent via queries like:
  → 'Analyze current supply chain and suggest improvements.'
- View charts, risk maps, and JSON logs in the output section.
- Check saved outputs: supply_chain_analysis_*.json, session_summary_*.json.

Testing:

The test suite validates:

- Supplier delays
- Demand spikes
- Weather disruptions
- Stockouts

Each test outputs a pass/fail summary and detailed reasoning logs.

Sample Result:

✅ 5/5 Tests Passed

📄 Results saved to test_results.json

Conclusion:

The **Multi-Agent Smart Supply Chain Optimizer** successfully demonstrates how Artificial Intelligence, when combined with modular agent-based design and real-world data integration, can revolutionize supply chain management.
This system achieves **autonomous decision-making**, **collaborative optimization**, and **explainable intelligence**, making it suitable for scalable, real-world industrial deployment.

Through its **Planner/Manager Agent**, the system coordinates specialized agents—Inventory Management, Supplier, Route Optimization, and Risk Management—each focused on a distinct operational domain. These agents communicate dynamically through a shared communication bus and produce structured, transparent logs in JSON format, ensuring both **traceability** and **auditable reasoning**.

The solution integrates multiple data sources, including CSV-based datasets and live APIs such as **OpenRouteService** for route optimization and **OpenWeatherMap** for real-time risk assessment. In static mode, it can operate entirely offline using realistic mock datasets, ensuring flexibility across deployment environments.

Key outcomes include:

- **Improved operational visibility** with real-time stock and supplier insights.
- **Data-driven route and risk management** decisions minimizing costs and disruptions.
- **Explainable AI-driven decision logs**, enhancing trust and accountability.
- **Feedback loops and persistent memory**, allowing agents to learn and improve over time.

The system's modular, extensible architecture adheres to **industry-grade standards**, enabling future expansion—such as integrating predictive demand forecasting, IoT-based tracking, or ERP connectivity.
By combining intelligent agents, transparent decision-making, and adaptive learning, this project lays a strong foundation for the next generation of **autonomous, resilient, and sustainable supply chain systems**.