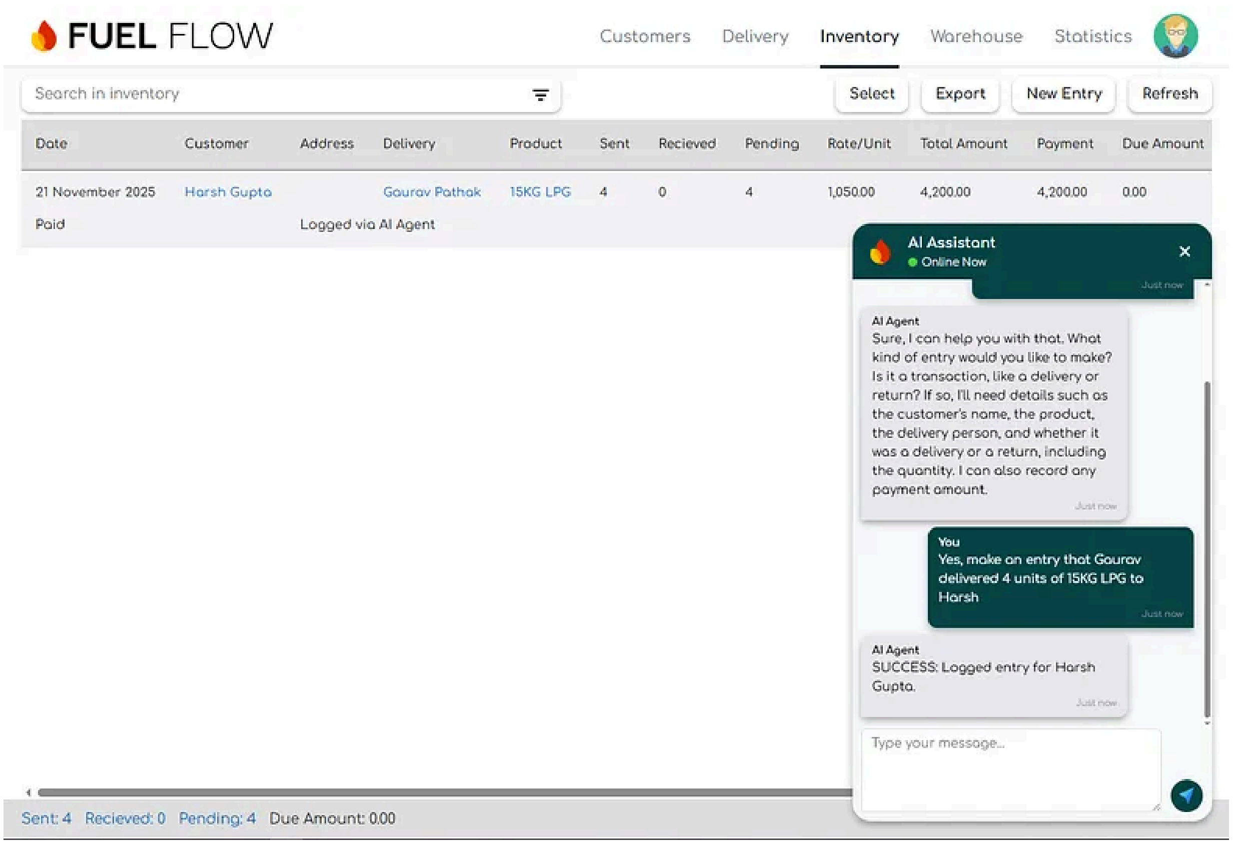
Transforming Logistics with Agentic AI on Google Cloud



Date	Customer	Address	Delivery	Product	Sent	Recieved	Pending	Rate/Unit	Total Amount	Payment	Due Amount
21 November 2025	Harsh Gupta		Gaurav Pathak	15KG LPG	4	0	4	1,050.00	4,200.00	4,200.00	0.00
Paid		Logged via AI Agent									

AI Assistant
Online Now

AI Agent
Sure, I can help you with that. What kind of entry would you like to make? Is it a transaction, like a delivery or return? If so, I'll need details such as the customer's name, the product, the delivery person, and whether it was a delivery or a return, including the quantity. I can also record any payment amount.
Just now

You
Yes, make an entry that Gaurav delivered 4 units of 15KG LPG to Harsh
Just now

AI Agent
SUCCESS: Logged entry for Harsh Gupta.
Just now

Type your message...

Sent: 4 Recieved: 0 Pending: 4 Due Amount: 0.00

AI Data Entry for Logistics Business

GitHub : G-Event_Fuel_Flow

App Link: Fuel-Flow

Video Demo: Google Drive

Transforming Logistics with Agentic AI on Google Cloud

Learn to build a production-ready Agentic AI that eliminates manual data entry. In this guide, we will combine Angular, Python (FastAPI) on Cloud Run, and Gemini 2.5 Flash to create a business manager that understands business context and executes complex database transactions.

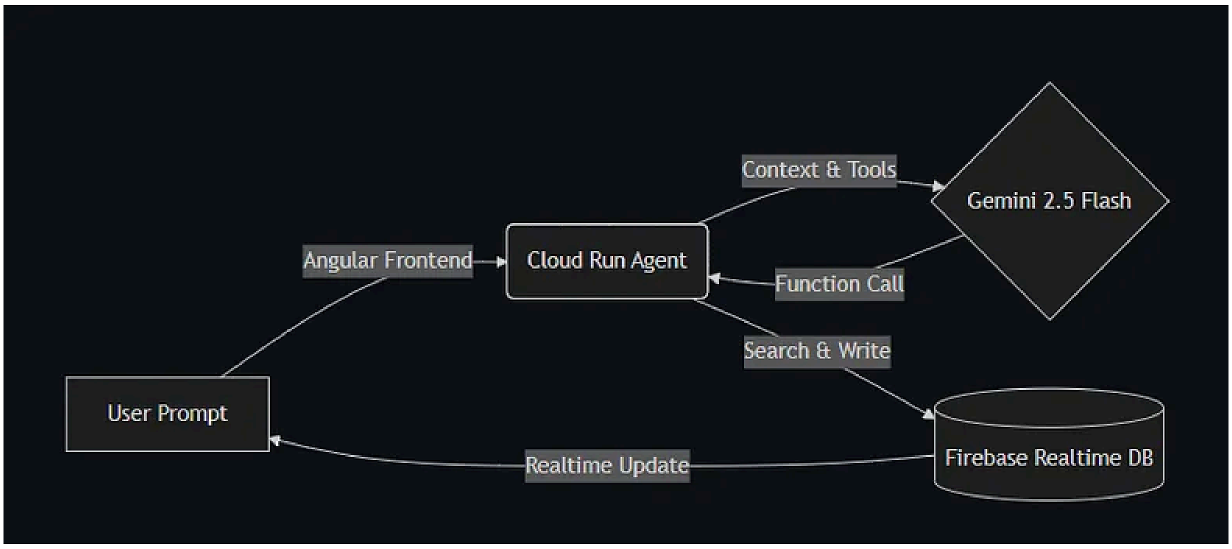
Introduction

In the fast-paced logistics industry, the speed and accuracy of data entry are critical for operational success. This blog explores how we developed FuelFlow AI, an agentic inventory manager that integrates seamlessly with Google Cloud services. By leveraging Angular, Cloud Run, and the advanced reasoning of Gemini 2.5 Flash, we transformed the way wholesale distributors track inventory, replacing complex manual forms with intuitive natural language commands to make logistics management more efficient and error-free.

Design

We chose a Hybrid Serverless Architecture to combine the responsiveness of a Single Page Application (SPA) with the reasoning power of a secure backend agent.

High-Level Architecture



Design Rationale

1. **Frontend (Angular):** We needed a reactive UI for the warehouse dashboard. Using the Web Speech API allows us to capture voice commands natively in the browser.
2. **Backend (Cloud Run):** We chose Google Cloud Run for the "Brain" because it scales to zero (cost-effective) and allows us to run a stateful Python agent. The backend handles the sensitive logic of searching the database and constructing complex JSON objects, keeping the frontend lightweight.
3. **AI Model (Gemini 2.5 Flash):** We switched from Gemini 1.5 Pro to 2.5 Flash to reduce latency. For a real-time voice interface, sub-second response times are critical.
4. **Database (Google Firebase):** The Realtime Database ensures that as soon as a driver speaks, the inventory dashboard updates instantly across all devices.

Prerequisites

Before we build, ensure you have the following tools and knowledge:

1. Google Cloud Project: With Vertex AI API enabled.
2. Firebase Project: A Realtime Database instance created.
3. Python 3.11+: For the backend agent.
4. Node.js & Angular CLI: For the frontend application.
5. Basic Knowledge: REST APIs, JSON structure, and TypeScript.

Step-by-step instructions

We will build this agent in three major stages: Defining the Tools, Building the Backend Logic, and Connecting the Interface.

Step 1: Define the "Agentic" Tools

The core of our system isn't the prompt, but the **Function Definitions.** We need to tell Gemini strictly how to structure a transaction.

We use `vertexai.generative_models.FunctionDeclaration` to define the schema:

```
complex_transaction_func = FunctionDeclaration(
    name="process_transaction",
    description="Log a business transaction where goods are delivered to a custo
    parameters={
        "type": "object",
        "properties": {
            "customer_name": {
                "type": "string",
                "description": "The end client/customer who bought or returned t
            },
            "product_name": {"type": "string", "description": "Name of product (
            "sent_units": {"type": "integer", "description": "Quantity SOLD/DELI
            "received_units": {"type": "integer", "description": "Quantity RETUR
        },
        "required": ["customer_name", "product_name"]
    }
)
```

Step 2: Implement the Backend Search Logic (Grounding)

LLMs are bad at guessing IDs. If we ask Gemini for a User ID, it might hallucinate. Instead, we ask Gemini for the **Name**, and use Python to find the **ID.**

In your `main.py` (FastAPI app):

```
def execute_complex_write(cust_name, prod_name, sent, received):
    # 1. Search Firebase for the name "Ramesh"
    customer_data = find_customer_by_name(cust_name)

    # 2. Construct the complex Transaction Object
    transaction_data = {
        "customer": {
            "fullName": customer_data.get('fullName'),
            "userId": customer_data.get('userId'), # The REAL ID from DB
        },
        "deliveryDone": [
            {
                "productId": find_product_id(prod_name),
                "sentUnits": sent,
                "recievedUnits": received
            }
        ],
        "timestamp": {".sv": "timestamp"}
    }

    # 3. Write to Firebase
    ref = db.reference('transactions')
    ref.push(transaction_data)
    return f"SUCCESS: Logged {sent} units for {cust_name}."
```

Step 3: Handle the Function Call in the Chat Loop

When Gemini responds, we must check if it wants to "call a function" rather than just chat.

```
@app.post("/chat")
async def chat_endpoint(request: Request):
    chat = model.start_chat()
    response = chat.send_message(user_message)

    # Check if Gemini wants to execute logic
    part = response.candidates[0].content.parts[0]

    if part.function_call:
        fc = part.function_call
        # Execute our Python logic from Step 2
        result = execute_complex_write(**dict(fc.args))
        return {"reply": result}

    return {"reply": response.text}
```

Result / Demo

At the end of this implementation, you have a system that turns unstructured speech into structured data.

The Workflow in Action:

1. **User Says:** *"Ramesh returned 5 Oxygen cylinders and took 10 LPG cylinders."*

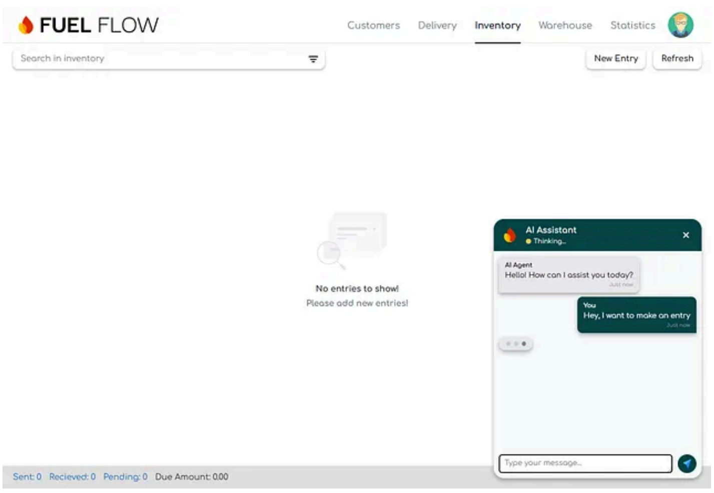
2. **AI Parses:** Identifies “Ramesh”, “Return: 5 Oxygen”, “Sale: 10 LPG”.

3. **System Actions:**

- Searches DB for “Ramesh” -> Found `userId: Xmf...`
- Updates Inventory -> Oxygen +5, LPG -10.
- Logs Transaction.

Visualizing the Impact: By moving from forms to an agentic interface, we achieved:

- **90% Faster Data Entry:** A 2-minute form process is now a 5-second command.
- **Zero “Fat-Finger” Errors:** The AI validates product names against the catalog before logging.
- **Real-time Sync:** The warehouse dashboard updates instantly.

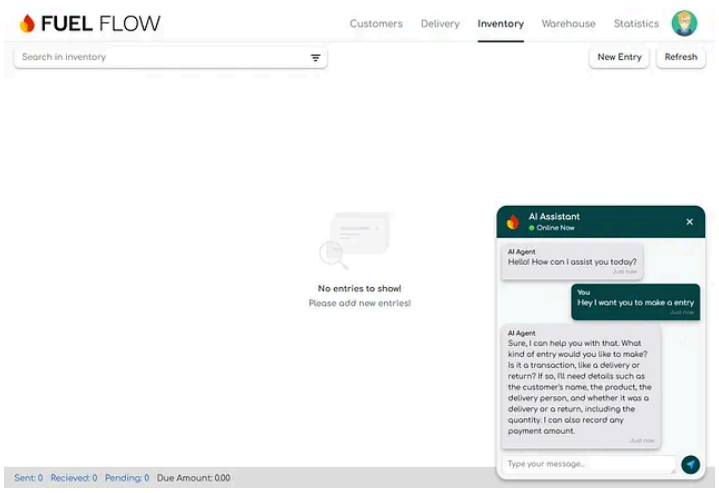


No entries to show!
Please add new entries!

AI Agent
Hello! How can I assist you today?
Just now

You
Hey, I want to make an entry
Just now

Sent: 0 Recieved: 0 Pending: 0 Due Amount: 0.00



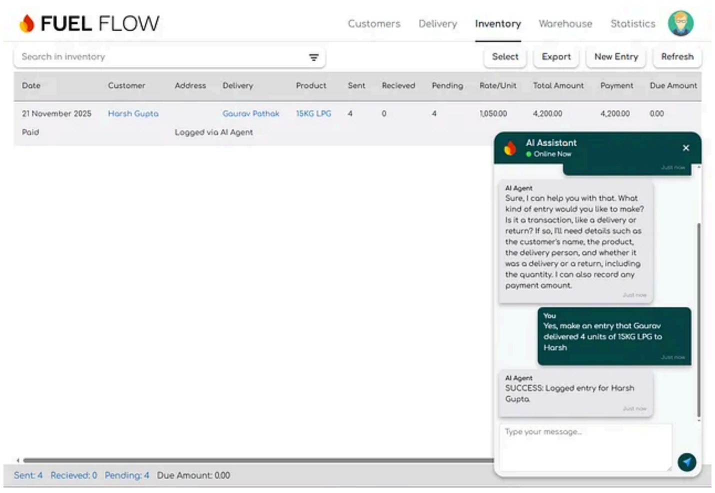
No entries to show!
Please add new entries!

AI Agent
Hello! How can I assist you today?
Just now

You
Hey I want you to make a entry
Just now

AI Agent
Sure, I can help you with that. What kind of entry would you like to make? Is it a transaction, like a delivery or return? If so, I'll need details such as the customer's name, the product, the delivery person, and whether it was a delivery or a return, including the quantity. I can also record any payment amount.
Just now

Sent: 0 Recieved: 0 Pending: 0 Due Amount: 0.00



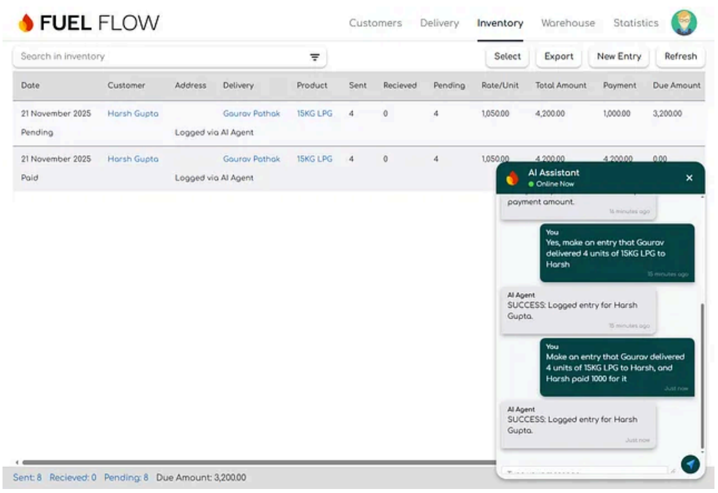
Date	Customer	Address	Delivery	Product	Sent	Recieved	Pending	Rate/Unit	Total Amount	Payment	Due Amount
21 November 2025	Harsh Gupta		Gaurav Pathak	15KG LPG	4	0	4	1,050.00	4,200.00	4,200.00	0.00
Paid		Logged via AI Agent									

AI Agent
Sure, I can help you with that. What kind of entry would you like to make? Is it a transaction, like a delivery or return? If so, I'll need details such as the customer's name, the product, the delivery person, and whether it was a delivery or a return, including the quantity. I can also record any payment amount.
Just now

You
Yes, make an entry that Gaurav delivered 4 units of 15KG LPG to Harsh
Just now

AI Agent
SUCCESS: Logged entry for Harsh Gupta.
Just now

Sent: 4 Recieved: 0 Pending: 4 Due Amount: 0.00



Date	Customer	Address	Delivery	Product	Sent	Recieved	Pending	Rate/Unit	Total Amount	Payment	Due Amount
21 November 2025	Harsh Gupta		Gaurav Pathak	15KG LPG	4	0	4	1,050.00	4,200.00	1,000.00	3,200.00
Pending		Logged via AI Agent									
21 November 2025	Harsh Gupta		Gaurav Pathak	15KG LPG	4	0	4	1,050.00	4,200.00	4,200.00	0.00
Paid		Logged via AI Agent									

payment amount.
16 minutes ago

You
Yes, make an entry that Gaurav delivered 4 units of 15KG LPG to Harsh
15 minutes ago

AI Agent
SUCCESS: Logged entry for Harsh Gupta.
15 minutes ago

You
Make an entry that Gaurav delivered 4 units of 15KG LPG to Harsh, and Harsh paid 1000 for it
Just now

AI Agent
SUCCESS: Logged entry for Harsh Gupta.
Just now

Sent: 8 Recieved: 0 Pending: 8 Due Amount: 3,200.00

What's next?

To take this project further, consider these expansions:

- **Multimodal Input:** Use Gemini 2.5 Flash's vision capabilities to allow drivers to upload a photo of a handwritten delivery slip instead of speaking.
- **Proactive Alerts:** Have the agent analyze the transaction history and warn the user: *"Ramesh has too many pending empty cylinders. Should we collect them?"*
- **Offline Support:** Implement local caching in the Angular app for areas with poor connectivity.

Call to action

To learn more about Google Cloud services and to create impact for the work you do, get around to these steps right away:

- Register for Code Vipassana sessions
- Join the meetup group Datapreneur Social
- Sign up to become Google Cloud Innovator
- Let's Connect on LinkedIn

Agentic Ai Google Cloud Platform

Written by Harsh Gupta

0 followers · 1 following

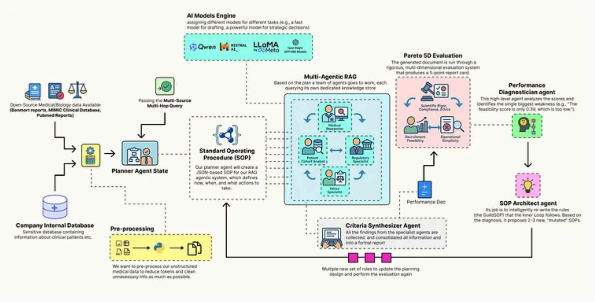
Edit profile

No responses yet

Harsh Gupta

What are your thoughts?

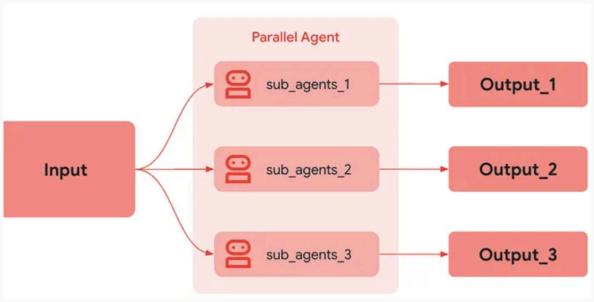
Recommended from Medium


AI Models Engine
Pareto 5D Evaluation
Multi-Agentic RAG
Performance Diagnostician agent
Planner Agent State
Standard Operating Procedure (SOP)
Company Internal Database
Pre-processing
Criteria Synthesizer Agent
SOP Architect agent

In Level Up Coding by Fareed Khan

Building a Self-Improving Agentic RAG System

Specialist agents, multi-dimensional eval, Pareto front and more.



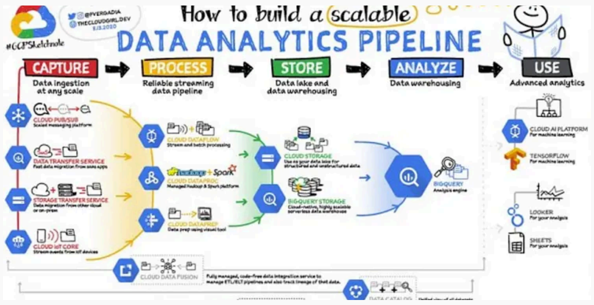
In AI Unboxed by Gopi Donthireddy

The Google ADK Playbook: Part 6 —Efficient Workflows with...

Welcome to the Part 6 of the Google ADK Playbook! We've built an incredible system o...

6d ago 1.1K 5

Aug 5

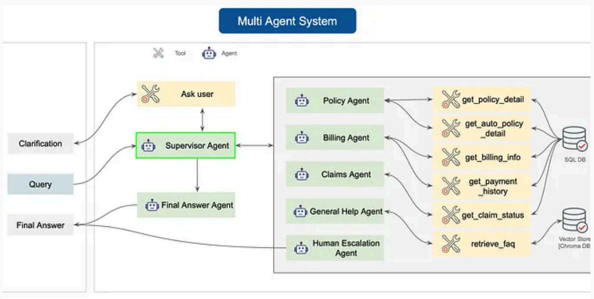


In Google Cloud - Community by Antonella Blasetti

The GCP Data Pipeline: From Raw Ingestion to Business Intelligence…

The whole story....easy way

Nov 10 2

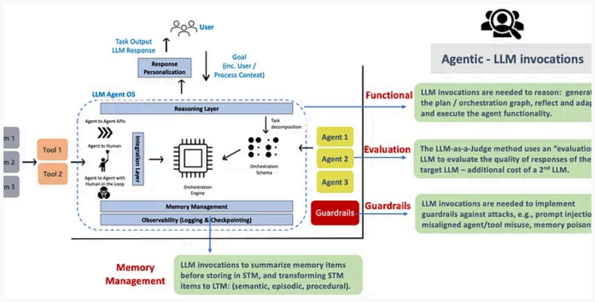


In Towards AI by Alpha Iterations

Agentic AI Project: Build a Multi-Agent system with LangGraph an…

This is an end to end project on building a multi-agent insurance support system using…

Nov 12 299 7



In AI Advances by Debmalya Biswas

Agentic AI FinOps: Cost Optimization of AI Agents

the hidden cost of non-functional LLM calls for Memory, Evals & Guardrails

Nov 12 278 7

Context Engineering: Sessions, Memory

Aakash Gupta

Google Just Dropped 70 Pages on Context Engineering. Here's What…

Imagine an AI that remembers you're vegan. Knows your debugging style. Recalls that…

5d ago 358 7

See more recommendations