Building a Product Table with Inventory Sync in Pega Constellation: A POC Walkthrough

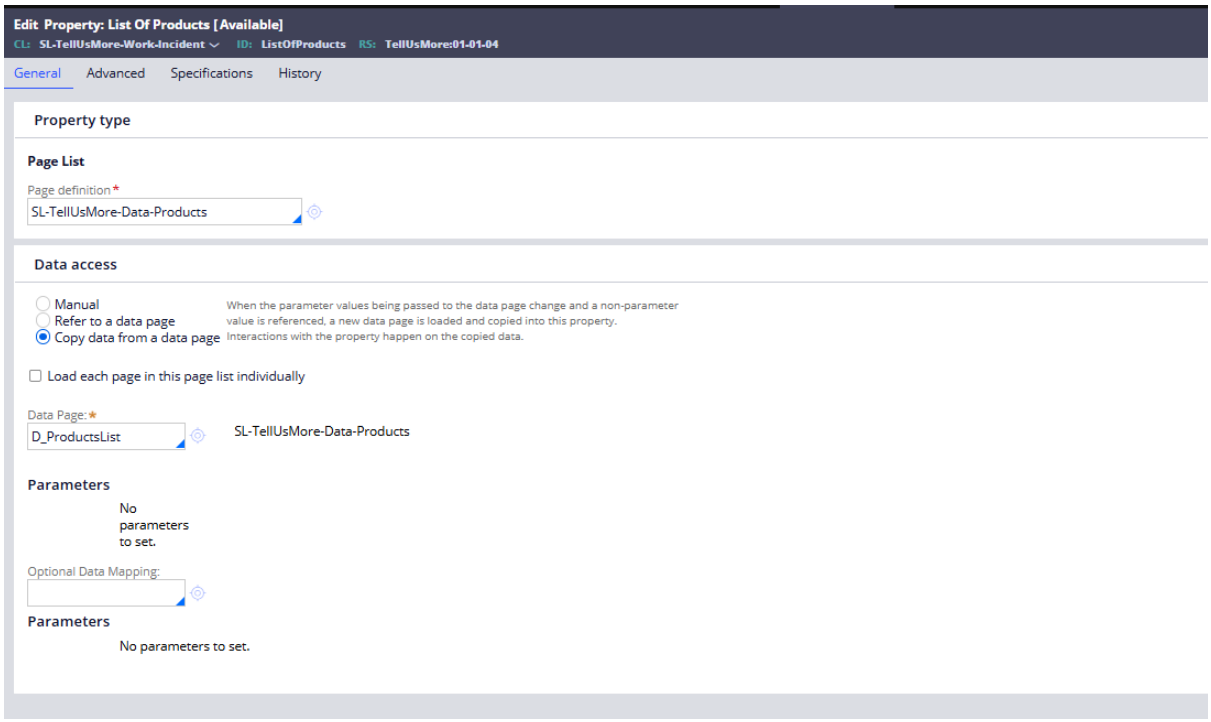
As Constellation becomes the new standard for building modern Pega applications, many of us are exploring how to implement common business use cases using its new design system and capabilities. In this article, We'll Walk you through a **proof of concept (POC)** We built using Constellation that demonstrates how to create a **dynamic product table** with **real-time calculations** and **inventory management**.

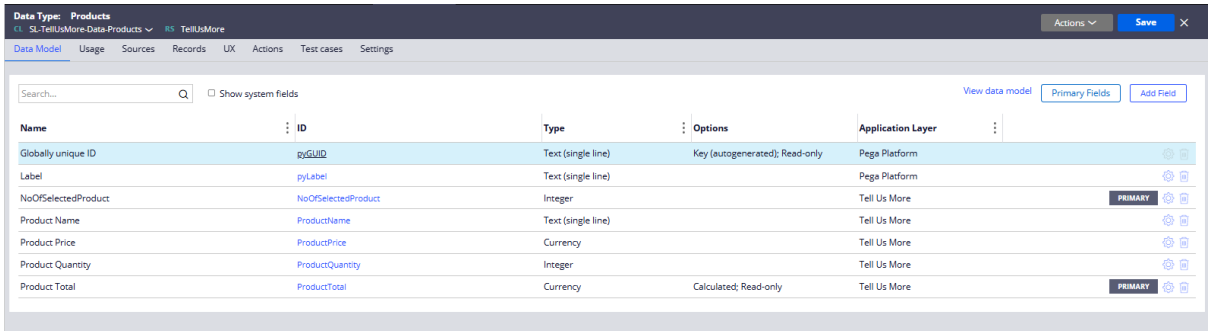
Use Case Overview

The goal was to build a **product selection interface** where users can:

- Select products from a predefined list (sourced from a data type).
- Enter quantities for each product.
- Automatically calculate the **product total** (price × quantity).
- Display a **Total Amount** (sum of all product totals).
- Update the **inventory** by subtracting selected quantities.

Data Model





I created a **Product** data type with the following fields:

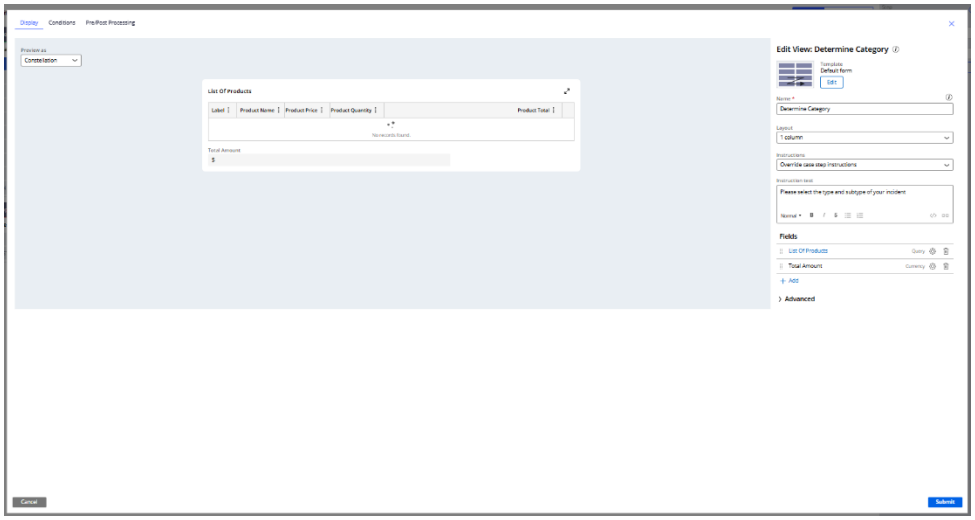
- ProductName (Text)
- ProductPrice (Decimal)
- ProductQuantity (Integer)
- ProductTotal (Read-only, calculated: ProductPrice * ProductQuantity)

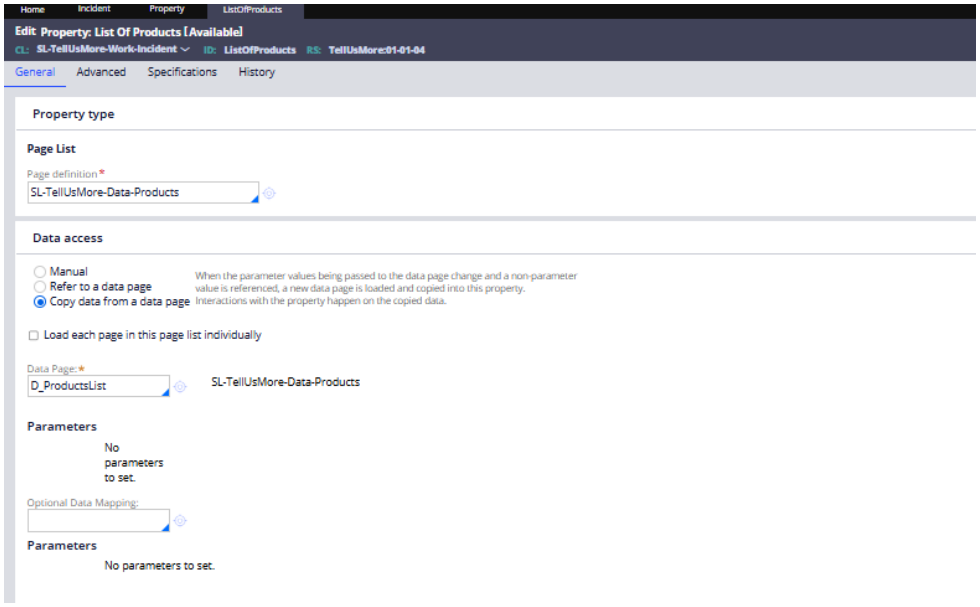
The **Total Amount** is a read-only field calculated as the **sum of all ProductTotal values**.

UX Configuration in Constellation

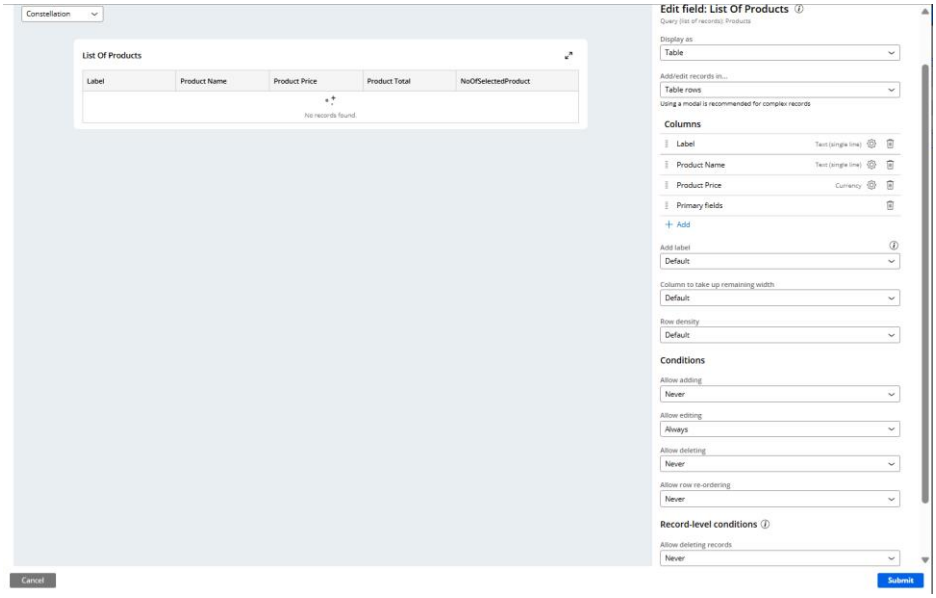
On the step in the case type:

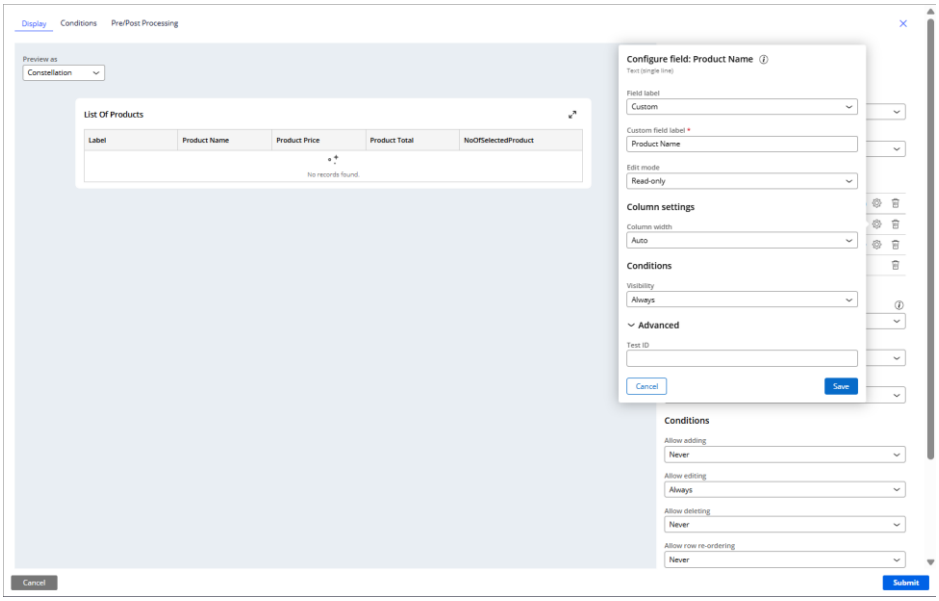
- I used an **embedded list** of the Product data type.



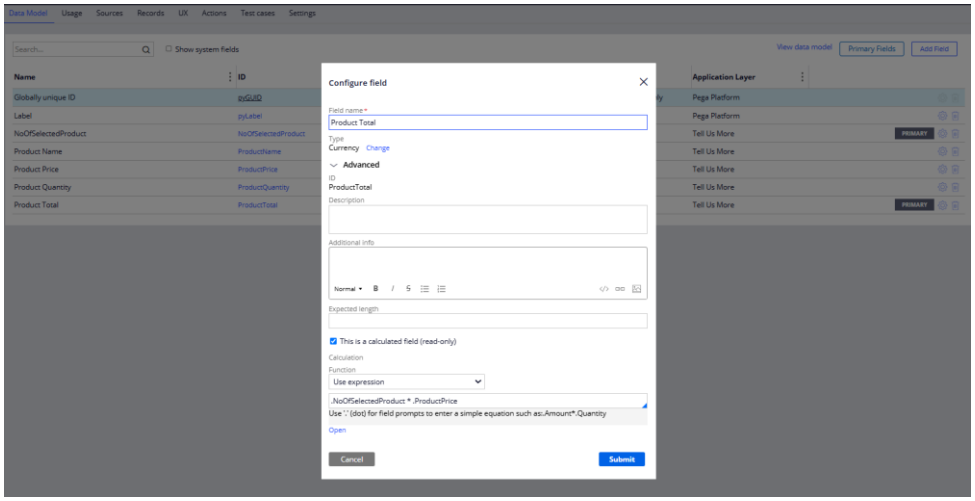


- The table is **editable**, allowing users to input quantities.

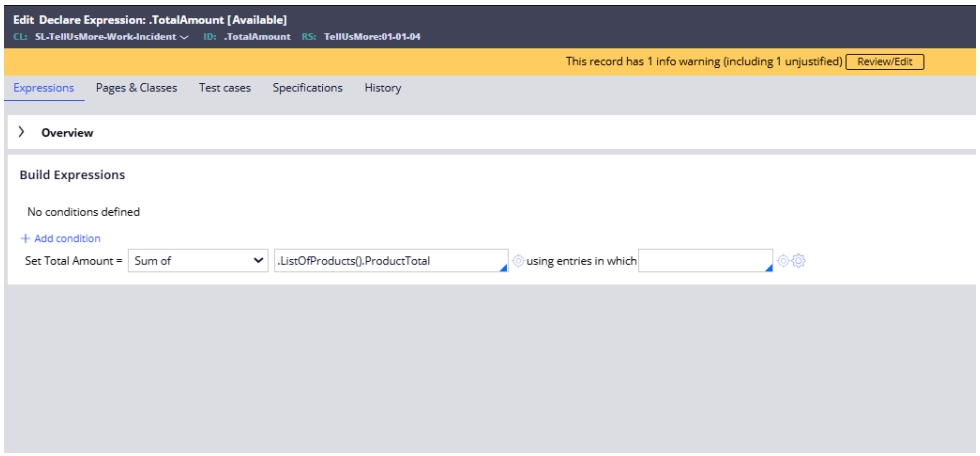




- The ProductTotal field is **auto-calculated** and read-only.

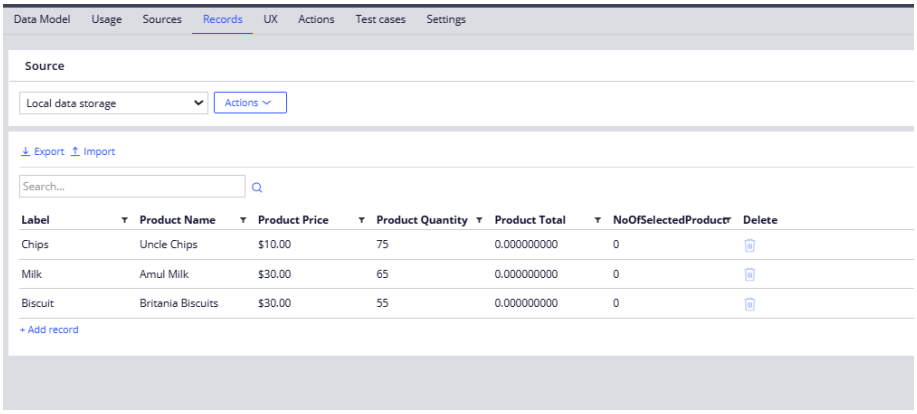


- The TotalAmount field is displayed below the table and updates in real time.
Declare Expression Field

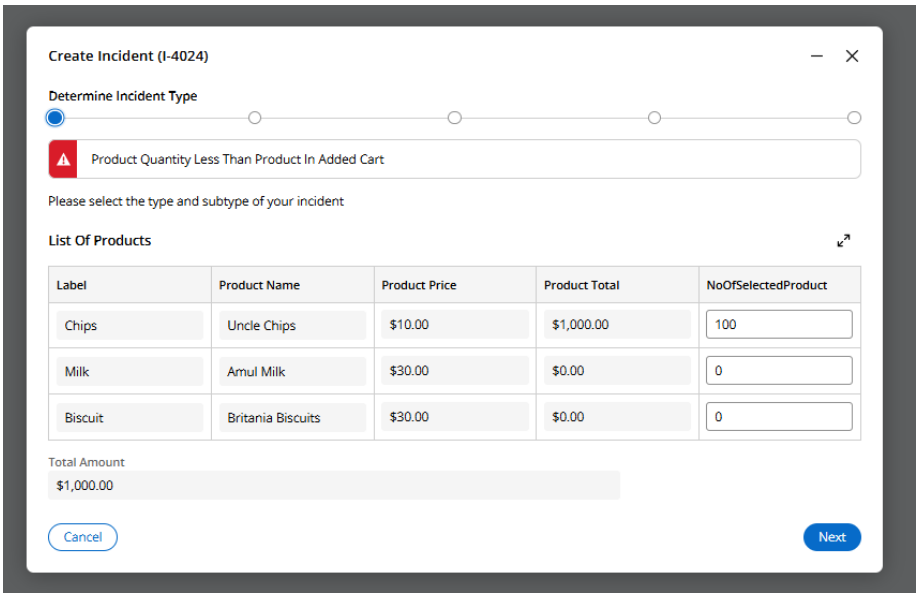


Inventory Management Logic

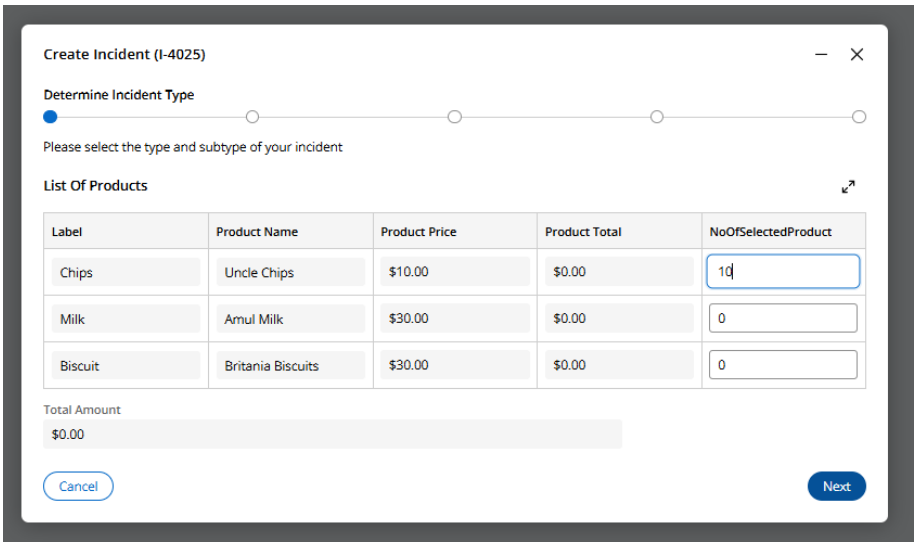
- To manage inventory:
- When a product is added or its quantity is updated, the system **subtracts the quantity** from the available inventory.

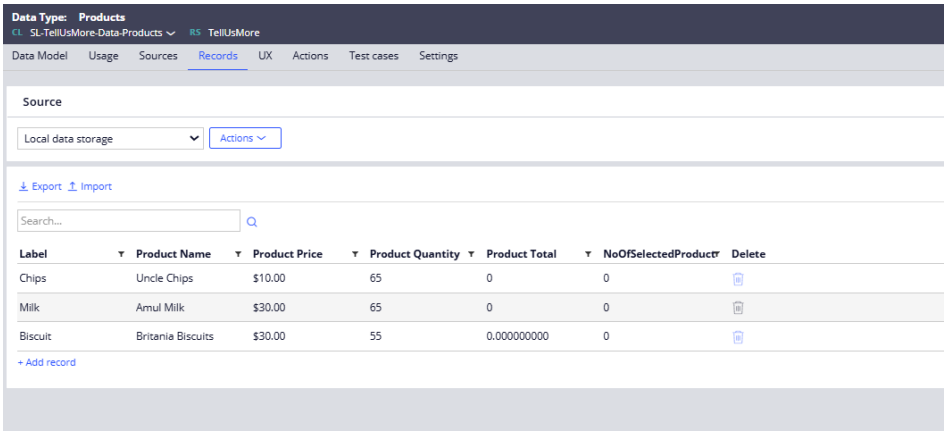


I added **validation** to prevent users from entering quantities greater than what's available.

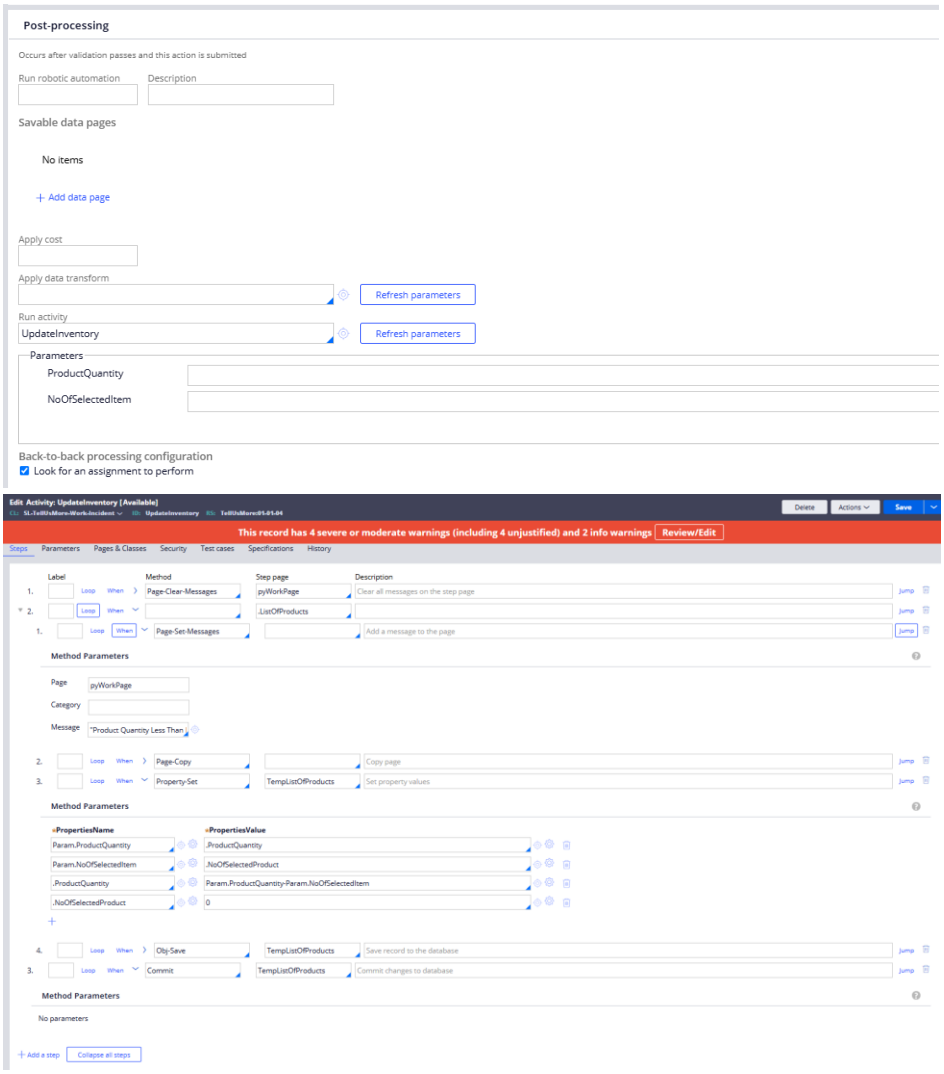


Added valid Number of items





- If a product is removed or quantity is reduced, the inventory is **restored accordingly**.
- **UpdateInventory** : Activity is called at post processing of the flow action



Key Learnings

- **Constellation's editable tables** are powerful for building interactive, real-time interfaces.
- **Declarative expressions** make it easy to calculate totals without writing custom code.
- **Inventory sync** can be handled using data transforms or decision rules triggered on change.
- **Validation rules** are essential to ensure data integrity and a smooth user experience.

This POC helped me understand how to use Constellation's new capabilities to build a real-world use case with dynamic data, calculations, and backend logic. I hope this helps others who are just getting started with Constellation!

Feel free to reach out or comment if you'd like to see a demo or need help implementing something similar.