

**Palantir Foundry Data Management Platform Proposal**

|  |
| --- |
|  |
| Take Home Test - Response |
| Status |
| Baselined |
| Version |
| 1.0 |
| Date of the Document |
| 02-25 -2024 |
|  |
| Prepared by Infosys Team |

**Document History**

**Change History**

This section represents the change history of the document. All revisions of the document must be tracked by identifying a new version number, the date it was modified, the person making the change, and the reason for the change. This document must be placed under version control. The following table can be used to track the change history:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date Modified** | **Author(s)** | **Reason for Change** |
| 1.0 | 25-FEB-2024 | Infosys Team | Baselined version |

**Purpose of Document**

The main purpose of the document is to capture the execution methodology to address the provided problem statements.

Table of Contents

[1. PROBLEM STATEMENT 1 : Plotting a Dynamic Cube given input face N 4](#_Toc159793774)

[1.1 Overview 4](#_Toc159793775)

[1.2 Solution Approach 4](#_Toc159793776)

[1.3 Component Details 4](#_Toc159793777)

[1.4 Validation 5](#_Toc159793778)

[1.4.1 Scenario 1 5](#_Toc159793779)

[1.4.2 Scenario 2 6](#_Toc159793780)

[1.4.3 Scenario 3 7](#_Toc159793781)

[1.4.4 Test Strategy recommendations: 7](#_Toc159793782)

[1.5 Reference 7](#_Toc159793783)

[2. PROBLEM STATEMENT 2 : Weekly inventory Silos report generation 8](#_Toc159793784)

[2.1 Overview 8](#_Toc159793785)

[2.2 Solution Approach 8](#_Toc159793786)

[2.3 Component Details 9](#_Toc159793787)

[2.4 Assumptions 9](#_Toc159793788)

[2.5 Validation 10](#_Toc159793789)

[2.5.1 Test Strategy recommendations: 10](#_Toc159793790)

[2.6 Reference 10](#_Toc159793791)

[3. PROBLEM STATEMENT 3: Object Modelling 11](#_Toc159793792)

[3.1 Overview 11](#_Toc159793793)

[3.2 Solution 11](#_Toc159793794)

[3.2.1 PSPS Initiative Workflow 12](#_Toc159793795)

[3.2.2 PSPS Initiative Workflow – Object Modelling 13](#_Toc159793796)

# PROBLEM STATEMENT 1 : Plotting a Dynamic Cube given input face N

## Overview

|  |  |
| --- | --- |
| Brief Description of Problem | Plot a N x N x N cube with N**3** cubes inside |
| Outcome Expected | 3D image of a cube with size relative to N. Must have labels for axes |

## Solution Approach

Chart below describes the thought process and various steps taken to plot the model.

**A diagram of a grid

Description automatically generated**

|  |  |
| --- | --- |
| Process # | Description |
| 1 | Matplotlib was selected for tight integration and minimalistic design |
| 2 | N x N x N grid for coordinate arrays |
| 3 | Using matplotlib functions to create a visual representation of the grid |
| 4 | Example: N=4 would result in 4x4x4 cube |

## Component Details

|  |  |  |
| --- | --- | --- |
| **Component Type** | **Script Name** | **Script Location** |
| Python | infosys\_dynamic\_cube.py |  |

## Validation

**Copilot** was given the task of inspecting and validating image for visual confirmation. It was able to successfully count the number of smaller cubes inside the large cube in the image produced by our plot.

### Scenario 1

We set N parameter to 3 to create a 3x3x3 cube. It plotted a three-dimensional plot of the cube with smaller cubes inside it.

**Result:**

A screenshot of a computer

Description automatically generated

### Scenario 2

We set N parameter to 4 to create a 4x4x4 cube. It plotted a three-dimensional plot of the cube with smaller cubes inside it.

**Result:**

A screenshot of a computer

Description automatically generated

### Scenario 3

On providing unexpected values as input for script to check on exception cases

**Result:**

A screenshot of a computer program

Description automatically generated

### Test Strategy recommendations:

* In the above use-case the output is a visual plot, and this can be validated through manual inspection and AI based tools.

## Reference

* Prompt helped to co-ordinate with LLM(Copilot) to get starting template “***Plotting a Dynamic Cube given input face N.”***
* **Copilot** helped us to perform Unit testing on the final version of the code.
* Matplotlib reference: [cheatsheets-1.png (1754×1240) (matplotlib.org)](https://matplotlib.org/cheatsheets/_images/cheatsheets-1.png)

# PROBLEM STATEMENT 2 : Weekly inventory Silos report generation

## Overview

|  |  |
| --- | --- |
| Brief Description of Problem | Create a weekly inventory report for Silos by using existing data and filling in gaps with historical daily averages. |
| Outcome Expected | Report generated should match the reference specified |

## Solution Approach

Chart below describes the thought process and various steps taken to plot the model.

A diagram of a process flow

Description automatically generated

|  |  |
| --- | --- |
| Process # | Description |
| 1 | Load data from silo\_actuals.csv & historical\_averages.csv |
| 2 | Initialize output dataset with June dates in first column |
| 3 | Add ‘day’ column temporarily to the output dataset to facilitate the join |
| 4 | Join input datasets with output dataset |
| 5 | Provide Window specifications to calculate required running totals and perform the aggregation |
| 6 | Drop columns and format data to match with specified report format |
| 7 | Refactor to chain transforms according to recommendation in requirement |

## Component Details

|  |  |  |
| --- | --- | --- |
| **Component Type** | **Script Name** | **Script Location** |
| Python | infosys\_silo\_weekly\_report.py |  |
| HTML version of notebook file (before refactoring) | infosys\_silo\_weekly\_report.html |  |
| Output generated | infosys\_silo\_weekly\_report\_output.csv |  |
| Test script | infosys\_silo\_weekly\_report\_validation\_script.py |  |

Note: **In typical practice, functions are often encapsulated and organized into a utility file. This utility file is then referenced from the main script. However, for simplicity and convenience, all functions and the actual code logic have been consolidated within a single script file.**

## Assumptions

* **monthly\_grand\_total column in the weekly should have the total tons incoming per month, to facilitate common reporting use-cases total monthly tons has been populated to every entry in the report.**

**example use-case: typically, this will help to while performing analysis to understand how much each day is constituting to the total monthly aggregate and find the usage patterns.**

## Validation

Output report generated has been compared against output reference provided in the problem statement using a python script to compare csv files.

Note: Validation has been performed by considering date, silo\_wt\_in\_tons, weekly\_total\_tons, daily\_total\_tons columns in the output.

A screenshot of a computer program

Description automatically generated

### Test Strategy recommendations:

Following is the testing approach usually followed:

* Test cases to validate around each function will be created considering the code coverage and boundary cases.
* Test cases can be bind and executed as part of CI (Continuous Integration)
* Pytest will be used to facilitate the execution of all the unit test cases once any code changes are committed.

## Reference

* [pyspark - Python Spark Cumulative Sum by Group Using DataFrame - Stack Overflow](https://stackoverflow.com/questions/45946349/python-spark-cumulative-sum-by-group-using-dataframe)

# PROBLEM STATEMENT 3: Object Modelling

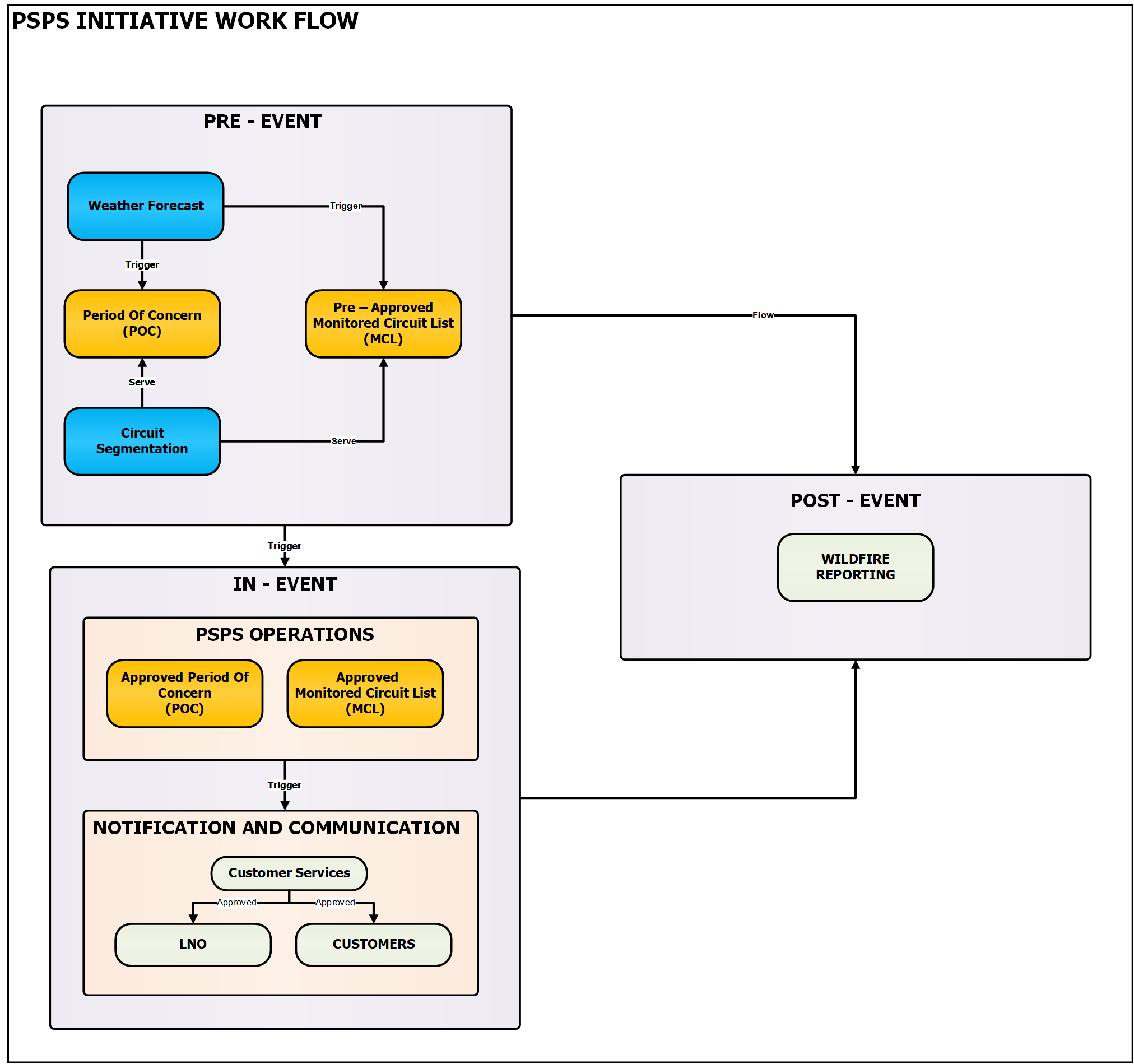
## Overview

|  |  |
| --- | --- |
| Brief Description of Problem | The PSPS initiative’s workflow is divided into three events: 1) Pre event 2) In event 3) Post event. Pre-event: A weather forecast and fire simulation ML engine triggers a “period of concern (POC)” on high risk “monitored circuit list (MCL)” and segments (note: segments are chunks within continuous circuits). This POC is approved through PSPS operations that triggers the notification process to notify, the notification event is approved by customer service (CS) team following which notification email are send two different types of customers  The End user customers and LNO (Liaison Officer) customers.   * The end user customers: as the name suggest, are affected end user customers (residential, commercial, etc) in that circuit. * The LNO customers: Are public service authorities like Sherriff’s office, Mayor’s office, hospitals, fire fighters, ambulances, etc. |
| Outcome Expected | Object model should be in-line with the provided scenario |

## Solution

Object model below describes the thought process during PSPS events.

### PSPS Initiative Workflow

****

### PSPS Initiative Workflow – Object Modelling

A diagram of a computer

Description automatically generated with medium confidence

****

(use <https://app.diagrams.net/> to read file)

**3.2.3 Assumptions:**

* Inputs provided to the Weather forecast/Fire simulator ML model are not represented in the design
* PSPS operations and Customer Service will have Approval work flow activity
* In, Pre & Post are considered as various phases of an event
* One Event can be associated with many “period of concerns”
* Above design has been limited to triggering email notification and did not consider tracking the email acknowlegements
* Event to Circuit are assumed to have many to many relationship i.e., one Event can be assiciated to many Circiuts and vise versa