**Shape, rectangle

Description automatically generated**



**Enterprise Level Data Management Framework – POC**

**Oct 2025**

**Version 1.0**

Table of Contents

[1.1 Objective 3](#_Toc211337548)

[1.2 Team & Timeline 3](#_Toc211337549)

[1.3 AI Companions leveraged 3](#_Toc211337550)

[1.4 Environment Setup 3](#_Toc211337551)

[1.5 Existing Data Hub Understanding 4](#_Toc211337552)

[1.6 Prompt Structure 5](#_Toc211337553)

[1.7 Prompt Content 5](#_Toc211337554)

[1.8 5](#_Toc211337555)

## Objective

To build from scratch an enterprise grade Data Management framework using GitHub Copilot as AI Companion.

## Team & Timeline

|  |  |
| --- | --- |
| **The Team** | |
| **Name** | **Role** |
| Subhash Bhaskaran | AI Engineer |
| Murugan Sundaram | Data Hub SME |
| Viswanatha R | Data Hub SME |

**Start Date: 14th Oct 2025**

**End Date: 30th Oct 2025**

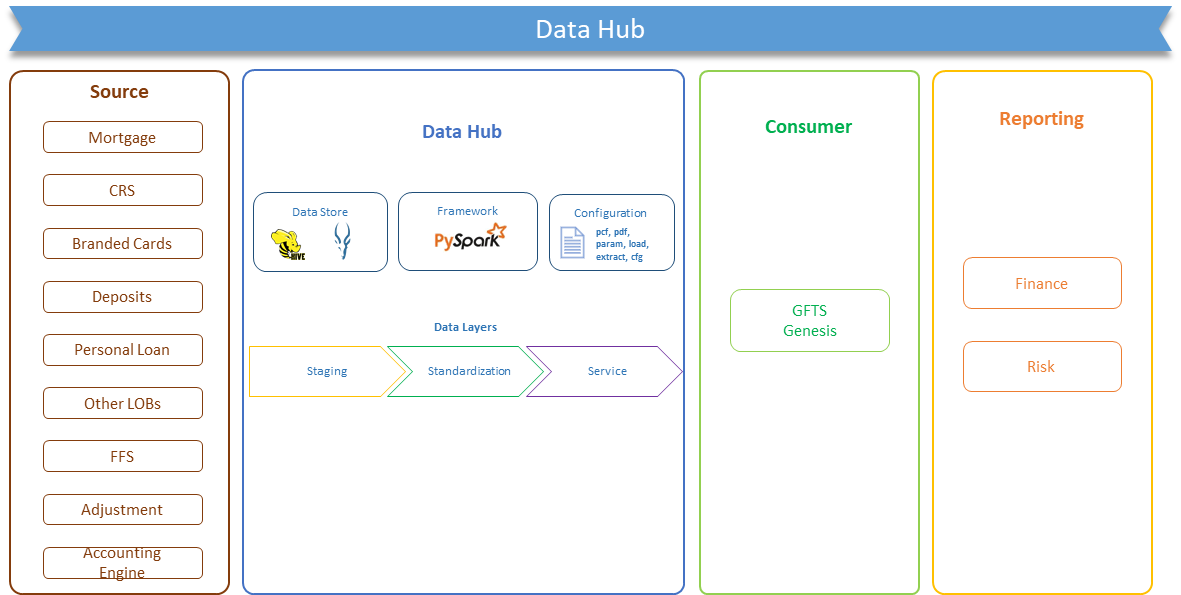
## AI Companions leveraged

1. Github Co Pilot

## Environment Setup

1. Developer Laptop
   * Github
   * Any IDE like IntelliJ or VS Code with Co Pilot plugin enabled

## Existing Data Hub Understanding



* Datahub is a centralized data store for USPB and Wealth sourcing data from Customer Interaction channels, Product Processors, and Operation Data sources to serve as golden copy/authorized data source for all downstream consumption.
* Once fully operationally, Data Hub will source data from more than 15 different PPs and will daily process more the 1500 inbound files and generate 1000+ outbound files
* Migration from Gensis to Datahub for Finance/Retail products (like Mortgage, Branded Cards, Deposits, PLN…) to be completed by Oct’25.
* Sustainability period for PLN is targeted till Feb’25
* Wealth products migration are targeted for Q4’26

## Prompt Structure

Requirement

Features

Transformation Supported

Application Folder Structure

Configuration JSON Structure

Technology Stack

Coding Language

Python Libraries

Database

Repository

Testing

## Prompt Content

Requirement

as a solution architect, provide a high-level design for a data‑processing application.

Features

The application should support all and more data processing components specified in the Transformation Supported section of the prompt

The application should provide a data pipeline component where the data processing steps can be configured. This component should be in xml format.

The application should provide a job pipeline component where multiple data pipeline components execution sequence can be configured in form of dependency

In production the application will handle thousands of input and output files sent or requested by multiple other internal applications. Hence will need to keep all configuration files input, output and data related to same application together. The application should also segregate the configuration into staging layer to load data from source to database table, standardization layer for data transformation on database table and finally service layer to generate output files from database table. Design the folder structure accordingly

Scalability, Performance are key to this application, hence use appropriate design patterns and libraries

The application should record the details on each data pipeline run and should provide a dashboard with operational metrics to identify run per day, success and failure and reason for failure

Transformation Supported

Input from CSV files

Input from database tables

Deduplication and Sorting

Sorting

Join

Lookup

Merging

Rollup or Aggregation

Output to CSV files

Data Masking

Technology Stack

any open source NoSQL database like Hive

Python

Dockers

Python Libraries

PySpark

numpy

pandas

Flask

FastAPI

Repository

create a new repository called smartFlow in GitHub.com/ltibfspoc

## 

