DaaS UI Project: User Interface Functional Requirements

1.0 Introduction

1.1. The current Data as a Service (DaaS) system operates exclusively as a back-end data provider, lacking a dedicated user interface (UI). All user interactions are conducted through a manual file drop process into designated Azure Blob Storage containers. This document outlines the functional requirements for a new DaaS UI designed to replace this manual workflow. The new interface will streamline the data submission process, enhance usability, and provide critical, actionable feedback to users managing stock configurations.1.2. The primary limitation of the existing system is its inability to provide detailed, line-level error reporting. When a user submits a file containing hundreds or thousands of data rows, a single formatting error or data inconsistency can cause the entire file to be rejected. The system moves the file to an "error" folder without specifying which line item caused the failure or the nature of the error. This forces users into a time-consuming and inefficient troubleshooting cycle of manual inspection, guesswork, and repeated file submissions.1.3. The development of the DaaS UI is driven by the following core objectives:

- **Centralized Interface:** Provide a single, user-friendly web interface for uploading all stock configuration files, eliminating the need for direct interaction with cloud storage containers.
- **Robust Validation and Reporting:** Implement a comprehensive validation and error reporting system based on the modern "CMG" model, making it the standard for all business units.
- **Granular Feedback:** Ensure users can clearly identify which individual data rows have been processed successfully and which have failed, along with precise reasons for each failure.
- **Improved Efficiency:** Significantly reduce the manual effort and time required for stock management tasks, error correction, and data resubmission.1.4. This document details the system's intended workflow and the specific functional requirements that the UI must fulfill to meet these objectives.

2.0 System Overview and Scope

2.1. The new DaaS UI will serve as the primary user interaction point for submitting stock configuration data into the DaaS-MAO-PMP ecosystem. It will act as a front-end layer, responsible for file ingestion, preliminary validation, and post-processing reporting. The core business logic for stock calculations and data transmission to downstream systems like MAO will remain within the DaaS back-end system.2.2. The existing manual process follows these steps:

1. Users manually place configuration files into specific Azure Blob Storage containers segregated by Business Unit: CDS, Super Sport, CFR, CFM, and CMG.
2. The DaaS back-end system periodically polls these containers to retrieve and process the files in their entirety.

3. Successfully processed files are moved to an "ACE" folder. Files containing one or more errors are moved in their entirety to an "error" folder with no detailed feedback.2.3. The new, streamlined workflow will be as follows:
4. A user logs into the DaaS UI and uploads a configuration file through a dedicated interface.
5. The UI performs a series of initial validation checks on the file's structure and data formats, providing immediate feedback to the user.
6. Upon successful initial validation, the user submits the file, which is then sent to the DaaS back-end for full processing.
7. The DaaS system processes the file on a line-by-line basis, accepting valid rows and flagging invalid ones.
8. The UI presents a detailed post-processing report to the user, clearly indicating the success or failure status of each line item from the original file.2.4. This high-level workflow establishes the foundation for the specific functional capabilities the UI must provide to the end-user.

3.0 Core Functional Requirements

3.1. This section outlines the universal functionalities the DaaS UI must provide. These foundational features are essential for supporting all specific stock management tasks, focusing on the user's core journey of file submission, validation, and status monitoring.3.2. **File Upload Interface**The UI must provide a clear and intuitive mechanism for users to upload data files (e.g., CSV, Excel). The system design must accommodate files containing hundreds or even thousands of line items, making a file-based upload approach a primary requirement over manual single-entry forms.3.3. **Pre-Submission Validation**Upon file upload and before final submission to the DaaS back-end, the UI must perform a set of preliminary validation checks to provide immediate feedback. This immediate feedback loop directly contrasts with the current "submit and wait blindly" process, saving significant user time and reducing frustration before the file enters the back-end processing queue. These checks must include, at a minimum:

- Verification that all mandatory fields are present and populated.
- Validation of key data formats (e.g., ensuring date fields conform to the required format).The system must alert the user instantly if these preliminary checks fail, highlighting the specific issues and allowing for immediate correction and re-upload before final processing is initiated.3.4. **Post-Processing Reporting and Error Handling**This feature is the cornerstone of the new UI and is designed to resolve the primary pain point of the current system.
- **Unified Reporting Model:** The UI must adopt and standardize the advanced, line-by-line reporting model currently utilized by the CMG business unit. This reporting standard must be applied to *all* business units and stock management functions.
- **Detailed Post-Processing Report:** After the DaaS back-end completes processing, the UI must present a comprehensive report to the user. This report must include:
- A clear status for each individual row from the uploaded file (e.g., 'Success', 'Error').
- For any row with an 'Error' status, a corresponding and descriptive error message explaining the reason for failure (e.g., "Item not found in DaaS," "Invalid date format").

- A mechanism for the user to download this detailed report for offline analysis and correction.
- **Row-Level Processing Logic:** The back-end processing logic, surfaced through the UI, must not reject an entire file due to one or more bad rows. The system must process all valid rows successfully and only report errors for the specific rows that are invalid.
- **Error Correction Workflow:** The user must be able to use the downloaded error report to easily identify, correct, and create a new file containing *only the failed rows* for re-submission.3.5. These core functionalities will support the specific business processes detailed in the following sections.

4.0 Specific Stock Management Features

4.1. This section details the specific business operations that users will perform through the UI. These features represent the primary use cases for manually configuring stock levels for items that are not managed automatically through primary source systems.4.2. **Feature: Config Stock (Overline Stock)**This feature, technically referred to as Overline Stock, allows users to manually configure stock (Config Stock). It is used to define stock quantities for items where the primary merchandise system holds no inventory data, such as consignment goods. It supports two primary modes: Pre-order (one time) and On-hand Available (daily). The behavior is determined by the Frequency field, which can be set to either 'one time' for pre-order scenarios or 'daily' for recurring on-hand stock.| Stock Type | Description & Logic || ------ | ------ || **Pre-order (one time)** | Represents a one-time stock injection for a specific item during a defined period. DaaS sends the specified quantity to MAO a single time at the start date. This stock level then depletes as sales occur and is not automatically replenished by DaaS. || **On-hand Available (daily)** | Represents a recurring, consistent stock level. DaaS sends the specified quantity to MAO *every day* for the duration of the configured start and end dates. This is used for consignment items where availability is expected to be consistent on a daily basis. |

- **Required Data Fields:** The user's upload file for this feature must contain the following columns: Item, Location, Supply Type, Frequency, Quantity, Start Date, End Date.4.3. **Feature: Safety Stock**This feature allows users to set a buffer quantity for an item at a specific location. This quantity is reserved and cannot be sold online. Its primary purpose is to account for potential stock discrepancies arising from in-store purchases, shrinkage (damage/theft), or shipping delays that are not yet reflected in the system. The safety stock value is subtracted from the system's on-hand quantity to calculate the final available-to-sell stock that is published to online channels.A common "Zero-Out" use case exists where users intentionally set an exceptionally high safety stock value (e.g., 999999) to ensure the calculated available-to-sell quantity becomes zero, effectively halting online sales for that item.
- **Required Data Fields:** The user's upload file for this feature must contain the following columns: Item, Location, Safety Stock Quantity, Start Date, End Date. If the End Date field is left blank, the safety stock setting is considered permanent until it is manually changed.4.4. These features represent the standard stock management tools, which will be complemented by more advanced functionalities specific to the CMG business unit.

5.0 CMG-Specific Features

5.1. This section outlines unique functionalities required by the CMG business unit, which operates on a more advanced version of the DaaS system. The UI must provide full support for these features, which may serve as a blueprint for future rollouts to other business units.5.2. **Feature: Stock Reservation**This feature enables the business to reserve specific quantities of stock for dedicated marketing campaigns on particular sales platforms (e.g., reserve 50 units for a Shopee campaign and 30 units for a Lazada campaign). In the current system architecture, the DaaS UI and back-end function as a pass-through mechanism for this feature. The UI accepts the user's file, and DaaS transmits the reservation data directly to MAO, which contains the core logic for managing the stock reservation.

- **Required Data Fields:** The user's upload file must contain the following columns: Campaign, Platform, Item, Location, Reserve Quantity, Start Date, End Date.5.3. **Feature: Stock Resync**This is a manual trigger function used to correct data discrepancies by forcing a synchronization of stock levels between MAO and the final sales endpoint (PMP). When a user uploads a file specifying an item and location, DaaS instructs MAO to re-transmit its current stock count for that item to PMP. This provides an on-demand mechanism to force an update, bypassing the standard, scheduled data synchronization cycles.
- **Required Data Fields:** The user's upload file must contain the following columns: Item, Location ID, Platform. If the Location ID field is left blank, the resync request will apply to the specified item across *all* of its associated locations for the given platform.