SOFTWARE REQUIREMENTS SPECIFICATION

StockFlow: Enterprise Inventory Management System

Version 1.0

Contents

1 1. Introduction

1.1 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to describe the functional and non-functional requirements for "StockFlow," a web-based Inventory Management System. This document is intended for the development team, project stakeholders, and quality assurance testers to ensure a shared understanding of the system's deliverables.

1.2 1.2 Scope

"StockFlow" is a browser-based application designed to digitize inventory tracking for Small-to-Medium Enterprises (SMEs). The system will:

- Facilitate the creation, reading, updating, and deletion (CRUD) of stock records.
- Provide real-time visualization of asset value and stock levels.
- Automate low-stock alerts to prevent supply chain disruptions.
- Maintain data persistence using client-side storage technologies for the prototype phase.

1.3 1.3 Definitions, Acronyms, and Abbreviations

SRS Software Requirements Specification

SKU Stock Keeping Unit (Unique Identifier for items)

CRUD Create, Read, Update, Delete

SPA Single Page Application

ROP Reorder Point (Threshold for low stock)

2 2. Overall Description

2.1 2.1 Product Perspective

StockFlow is a standalone web application. In its current iteration (Prototype/v1.0), it operates as a Single Page Application (SPA) utilizing the user's browser for logic execution and storage (‘LocalStorage‘). It is designed to be forward-compatible with a future RESTful backend.

2.2 2.2 Product Functions

The major functions of the system include:

1. **Dashboard Monitoring:** View high-level KPIs (Total Value, Item Count).
2. **Inventory Management:** Add new items, update quantities, and remove obsolete stock.
3. **Search & Retrieval:** Filter inventory data in real-time.
4. **Alerting:** Visual indicators for items below safety thresholds.

2.3 2.3 User Characteristics

- **Warehouse Administrator:** Technically literate user capable of managing data entry and interpreting dashboard analytics.
- **Inventory Manager:** Focuses on strategic reporting and asset valuation.

2.4 2.4 Constraints

- The system must run within a standard web browser (Chrome, Firefox, Edge) without plugins.
- The prototype relies on ‘LocalStorage‘, limiting storage capacity to approx. 5MB.
- Development must adhere to the RAD model time constraints (48 hours).

3 3. Specific Requirements

3.1 3.1 Functional Requirements

3.1.1 3.1.1 Inventory Dashboard

Description: The system shall present a dashboard summary upon login.

- **Input:** System Load Event.
- **Processing:** Aggregate price $\times$ quantity for all items; count items with $qty < 10$.
- **Output:** Display "Total Inventory Value", "Total Items", and "Low Stock Alerts".

3.1.2 3.1.2 Add Item (Create)

Description: Users shall be able to add new inventory records.

- **Input:** Item Name (String), Category (String), Quantity (Int), Price (Float).
- **Validation:** Name $\neq$ Null; Quantity ≥ 0; Price ≥ 0.
- **Output:** New record added to the data store; Table refreshed.

3.1.3 3.1.3 Update Item (Edit)

Description: Users shall be able to modify existing item details.

- **Input:** Selection of item ID, modified values.
- **Processing:** Locate item by ID, overwrite fields, save to storage.
- **Output:** Success notification ("Item Updated").

3.1.4 3.1.4 Delete Item (Remove)

Description: Users shall be able to permanently remove an item.

- **Input:** Click "Delete" action on specific row.
- **Processing:** Display confirmation modal. If confirmed, remove ID from array.
- **Output:** Item removed from Grid View.

3.1.5 3.1.5 Real-Time Search

Description: Users shall be able to filter the inventory list.

- **Input:** Alphanumeric search string.
- **Processing:** Filter array where 'Item.Name' contains 'InputString'.
- **Output:** Dynamic update of the Inventory Table to show only matches.

3.2 3.2 Non-Functional Requirements

3.2.1 3.2.1 Performance

- **Response Time:** All local interactions (Add/Edit) must complete in < 100ms.
- **Load Time:** Initial dashboard load must occur within 2 seconds on 4G networks.

3.2.2 3.2.2 Reliability

- **Data Persistence:** Data must survive browser refreshes via LocalStorage.
- **Availability:** System uptime target of 99.9% via static cloud hosting (Vercel).

3.2.3 3.2.3 Usability

- **Interface:** Must adhere to Material Design principles for clarity.
- **Accessibility:** Must support keyboard navigation (Tab-indexing) for form entry.

4 4. Interface Requirements

4.1 4.1 User Interfaces

The application will utilize a responsive web interface.

- **Sidebar:** Vertical navigation for module switching.
- **Data Grid:** Sortable columns for inventory listing.
- **Modals:** For data entry and confirmation dialogs to maintain context.

4.2 4.2 Software Interfaces

- **Browser API:** Utilizes the DOM API for rendering and Web Storage API for persistence.
- **Export Format:** Supports CSV generation for data portability.