# **A blue and white logo AI-generated content may be incorrect.**

OmniStock Inventory Tracker

Software Requirements Specification

*Version 1*

Team Number: 7

Project Manager: Brandon Tatum

Mentor: Sandeep Ginna

Team Members: Brandon Tatum, Esteban Spadea,

Bethany Hill, Serina Rodriguez, Dillon Hollis

## Revisions

* **Version:** 1.0
* **Primary Author:** Brandon Tatum
* **Description:** Initial Draft
* **Date Completed:** 03/09/2025

## Review History

* **Reviewer:** Sandeep Ginna
* **Version Reviewed:** 1.0
* **Date:** TBD

**Table of Contents**

1. **Introduction**
   * 1.1 Project Objectives
   * 1.2 Project Scope
   * 1.3 Project Overview
2. **Project Description**
   * 2.1 Project Features / Functions
   * 2.2 User Stories
   * 2.3 Use Cases
   * 2.4 Project Assumptions and Dependencies
3. **Project Collaboration and Documentation**
4. **Project Management**
5. **Requirements Specification**
   * 5.1 Business Requirements
   * 5.2 User Requirements
   * 5.3 Functional Requirements
   * 5.4 Non-Functional Requirements
   * 5.5 Implementation (Performance) Requirements (Optional)

**1. Introduction**

**1.1 Project Objectives**

OmniStock is an inventory management system designed to facilitate seamless inventory tracking, organization, and stock monitoring. It aims to provide an easy-to-use interface for small businesses and individuals to manage inventory efficiently.

**1.2 Project Scope**

The system will include:

* Secure user authentication
* Barcode scanning for inventory tracking
* Stock level alerts
* CRUD operations for inventory management
* Data analytics for inventory usage trends

The system will **not** include:

* Full ERP functionalities
* Role-based access control in the initial release

**1.3 Project Overview**

OmniStock will provide a cloud-based inventory tracking solution that allows users to scan, monitor, and manage stock levels in real time. The project follows Agile and Scrum methodologies.

**2. Project Description**

**2.1 Project Features / Functions**

OmniStock provides the following key features:

* Secure user authentication
* Barcode scanning for quick inventory entry
* Stock level monitoring with real-time alerts
* Data analytics and usage trends
* Import/export functionality for inventory data

**2.2 – 2.3 Use Case and User Stories**

* As a user, I want to easily create an account and securely login using my email and password to access my inventory.
* As a user, I want to scan barcodes quickly so that they can be added to the inventory system.
* As a user, I want to easily view detailed information about a product after scanning its barcode
* As a user, I want to receive alert notifications when an item’s inventory count falls too low so that I can restock in time.
* As a user, I want to easily view a list of recently scanned items and view recent inventory activity.
* As a user, I want to check the current stock levels of my items so that I can know immediately what is available.
* As a user, I want to add, view, update, or delete items in my inventory so I can manage my stock effectively.

**A diagram of a software process

AI-generated content may be incorrect.**

**2.4 Project Assumptions and Dependencies**

* Users must have internet access to use the application.
* The system depends on barcode scanner APIs for barcode recognition.
* Development will follow Agile principles and Scrum methodology.

**3. Project Collaboration and Documentation**

The following tools will be used for project collaboration and documentation:

* **GitHub**: Version control and CI/CD
* **Jira**: Task tracking and sprint planning
* **Teams**: Collaboration

**4. Project Management**

The project will be managed using Agile methodologies, specifically Scrum-based sprints. The following tools will be utilized:

* **Jira** for sprint planning and task tracking
* **GitHub** for version control and collaborative development

1. **Requirements Specification**

**5.1 Business Requirements**

| **Requirement ID** | **Requirement Description** | **MOSCOW** |
| --- | --- | --- |
| BR1 | We must establish better supplier communication by implementing a supplier portal to track real-time stock levels and reduce delivery delays by 20%. | M |
| BR2 | We must implement an AI-driven inventory forecasting tool to analyze demand trends and adjust stock levels with at least 90% accuracy. | M |
| BR3 | We should automate order tracking and implement real-time notifications to ensure at least 95% on-time delivery and increase customer satisfaction | S |

**5.2 User Requirements**

| **Requirement ID** | **Requirement Description** | **MOSCOW** |
| --- | --- | --- |
| UR1 | Users must be able to create an account and log in securely using their email and password. | M |
| UR2 | Users must be able to scan bar codes of grocery items | M |
| UR3 | Users must be able to view information about the product after scanning the bar code | M |
| UR4 | Users must receive and be able to view alert notifications if the stock level of an item decreases to a certain threshold | M |
| UR5 | Users must be able to view what items’ barcodes were recently scanned | M |
| UR6 | Users must be able to look at usage trends of items stored in database when barcode scanned | C |
| UR7 | Users must be able to look at stock levels of items within the database | M |
| UR8 | Users must be able to add, view, update, or delete items within the inventory database | M |

**5.3 Functional Requirements**

| **Requirement ID** | **Requirement Description** | **MOSCOW** |
| --- | --- | --- |
| FR1 | Users must be able to register, log in, and manage access credentials. | M |
| FR2 | CRUD operations must be available for inventory items. | M |
| FR3 | The system should notify users when stock levels fall below a predefined threshold. | S |
| FR4 | Users could import/export inventory data in CSV format. | C |
| FR5 | Different levels of permissions for administrators and standard users. | S |

**5.4 Non-Functional Requirements**

| **Requirement ID** | **Requirement Description** | **MOSCOW** |
| --- | --- | --- |
| NFR1 | The system should handle up to 1000 concurrent users with minimal latency. | M |
| NFR2 | Implements encrypted password storage and follows OWASP security guidelines and 2FA. | M |
| NFR3 | The system should scale to support increased data loads without performance degradation. | S |
| NFR4 | UI could be optimized for accessibility and responsiveness across multiple devices. | S |
| NFR5 | The system will aim for 99.9% uptime but won't guarantee it initially. | S |

**5.5 Implementation (Performance) Requirements**

| **Requirement ID** | **Requirement Description** | **MOSCOW** |
| --- | --- | --- |
| IR1 | The frontend will be built using React.js, optimized for responsive design and accessibility. | M |
| IR2 | The backend will be developed using Node.js with Express ensuring an API response time of ≤200ms for 95% of requests. | M |
| IR3 | The database will be hosted on AWS RDS using PostgreSQL, with automatic backups and read replicas for scalability. | M |
| IR4 | Authentication will be handled via OAuth 2.0 / JWT, with role-based access control (RBAC) for multi-user security. | M |
| IR5 | Unit testing will be conducted using Jest, with integration testing via Cypress. CI/CD pipelines will be used for automated deployments. | M |