# **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
* **Google Drive**: Document collaboration
* **Discord**: Team communication

**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

**5. Requirements Specification**

**5.1 Business Requirements *(Provided by Esteban Spadea)***

* The system must improve how we work with suppliers to avoid delays and keep stock levels consistent. *(Must Have)*
* The system must use data to predict demand and adjust inventory accordingly. *(Must Have)*
* The system should ensure that orders are delivered on time to maintain customer satisfaction. *(Must Have)*

**5.2 User Requirements *(Provided by Bethany Hill)***

* Users must be able to create an account and log in securely. *(Must Have)*
* Users must be able to scan barcodes of grocery items. *(Must Have)*
* Users must be able to view product details after scanning. *(Must Have)*
* Users must receive stock level alerts. *(Must Have)*
* Users must see recently scanned barcodes. *(Must Have)*
* Users should be able to view item usage trends. *(Should Have)*
* Users must be able to check stock levels. *(Must Have)*
* Users must be able to add, update, or delete inventory items. *(Must Have)*

**5.3 Functional Requirements *(Provided by Dillon Hollis)***

* Users must register, log in, and manage credentials. *(Must Have)*
* CRUD operations must be available for inventory items. *(Must Have)*
* The system should notify users when stock levels fall below a predefined threshold. *(Should Have)*
* Users could import/export inventory data in CSV format. *(Could Have)*
* Role-based access control will not be included in the initial release. *(Wont Have)*

**5.4 Non-Functional Requirements *(Provided by Brandon Tatum)***

* The system must handle up to 1000 concurrent users with minimal latency. *(Must Have)*
* Password storage must follow OWASP security guidelines. *(Must Have)*
* The system should scale to support increased data loads without performance degradation. *(Should Have)*
* The UI should be optimized for accessibility and responsiveness across multiple devices. *(Could Have)*
* The system will aim for 99.9% uptime but will not guarantee it initially. *(Wont Have)*

**5.5 Implementation (Performance) Requirements (Optional) *(Provided by Brandon Tatum)***

* The frontend will be built using React.js. *(Must Have)*
* The backend will be developed using Node.js with Express. *(Must Have)*
* The database will be hosted on AWS RDS using PostgreSQL. *(Must Have)*
* Authentication will be handled via OAuth 2.0 / JWT. *(Must Have)*
* Unit testing will be conducted using Jest, with integration testing via Cypress. *(Must Have)*