Software Requirements Specification for Quick Stock

Toqa Ezzatly/D-hub

December 12, 2024

Contents

1 Introduction **1**
- 1.1 Purpose . . . 1
- 1.2 Scope . . . 2
- 1.3 Definitions, Acronyms, and Abbreviations . . . 2
- 1.4 References . . . 2
- 1.5 Overview . . . 2

2 Overall Description **2**
- 2.1 Product Perspective . . . 2
- 2.2 Product Functions . . . 3
- 2.3 User Classes and Characteristics . . . 3
- 2.4 Operating Environment . . . 3
- 2.5 Design and Implementation Constraints . . . 3
- 2.6 User Documentation . . . 3
- 2.7 Assumptions and Dependencies . . . 4

3 Specific Requirements **4**
- 3.1 Functional Requirements . . . 4
  - 3.1.1 User Management (Login, Register) . . . 4
  - 3.1.2 Product Management . . . 4
  - 3.1.3 Inventory Display . . . 4
  - 3.1.4 Theme Management . . . 5
- 3.2 Non-Functional Requirements . . . 5
  - 3.2.1 Performance Requirements . . . 5
  - 3.2.2 Security Requirements . . . 5
  - 3.2.3 Usability Requirements . . . 5
  - 3.2.4 Maintainability Requirements . . . 5

4 Diagrams Overview **5**

1 Introduction

1.1 Purpose

This document specifies the software requirements for the "Quick Stock" inventory management system. It outlines the functionality, performance, and interface requirements

that the system must meet. This document is intended for all stakeholders, including developers, testers, and project managers.

1.2 Scope

The Quick Stock system is a web-based application designed for small businesses to track their product inventory. It provides functionalities for adding, viewing, editing, and deleting products, as well as basic user management including login and registration. The system is intended to be used by small store owners and their staff.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **UI:** User Interface
- **DB:** Database
- **HTTP:** Hypertext Transfer Protocol
- **SQL:** Structured Query Language
- **OOP:** Object Oriented Programming

1.4 References

- IEEE Standard 830-1998, Recommended Practice for Software Requirements Specifications
- Flask Documentation: `https://flask.palletsprojects.com/`
- SQLite Documentation: `https://www.sqlite.org/docs.html`

1.5 Overview

The remainder of this document is organized as follows: Section 2 describes the overall description of the product. Section 3 specifies the detailed functional and non-functional requirements.

2 Overall Description

2.1 Product Perspective

Quick Stock is a standalone web application, accessed through a standard web browser. It does not rely on other specific systems for its core functionality other than a standard web server and a database system. It interacts directly with the user through a web-based interface and uses a local SQLite database for data storage.

2.2 Product Functions

The system shall provide the following core functions:

- **Product Management:** Add, view, update, and delete products from the inventory.
- **User Management:** Register, log in, and log out user accounts.
- **Inventory Tracking:** Display all products with their current information (name, price, quantity).
- **Theme Control:** Ability to change theme with a style sheet selector.

2.3 User Classes and Characteristics

- **Store Owners:** Primary users with full access to manage products and the user accounts. They will have basic computer literacy.
- **Staff Members:** Users who may need to access inventory for product management, with access controlled by the store owner. They will have basic computer literacy.

2.4 Operating Environment

- **Platform:** Any modern operating system (Windows, macOS, Linux) with a compatible web browser.
- **Web Browser:** Modern web browsers such as Chrome, Firefox, Safari, or Edge.
- **Database:** SQLite will be used for local data storage.
- **Web Server:** The application uses Flask's built-in web server during development. A production-ready WSGI server like Gunicorn would be used for deployment.
- **Language:** Python 3.7 or higher

2.5 Design and Implementation Constraints

- The application shall be developed using Python and the Flask web framework.
- Data persistence shall be achieved using an SQLite database.
- The UI shall be developed using HTML, CSS, and basic JavaScript.
- Passwords should be stored in a hashed format.
- The application is designed for a small number of concurrent users.
- The application should be deployable on simple hosting solutions

2.6 User Documentation

A basic user manual with instructions on how to use each feature, register, login and perform all the functionalities will be provided as part of the final delivery, but this not this SRS document.

2.7 Assumptions and Dependencies

- Users are assumed to have basic computer literacy and an understanding of web browsing.
- The server is assumed to be running and accessible to the users.
- The database is assumed to be available for storage.

3 Specific Requirements

3.1 Functional Requirements

3.1.1 User Management (Login, Register)

- **REQ-1.1** The system shall allow users to register with a unique username and password.
- **REQ-1.2** The system shall securely store user passwords in a hashed format.
- **REQ-1.3** The system shall allow users to log in with their registered username and password.
- **REQ-1.4** The system shall maintain user session to keep logged user logged in
- **REQ-1.5** The system shall allow logged in users to log out.
- **REQ-1.6** Unlogged users should be redirected to login page.

3.1.2 Product Management

- **REQ-2.1** The system shall allow authorized users to add new products with a name, price, and quantity.
- **REQ-2.2** The system shall display a list of all products with their names, prices, and quantities.
- **REQ-2.3** The system shall allow authorized users to edit the details of an existing product (name, price, quantity).
- **REQ-2.4** The system shall allow authorized users to delete products from the inventory.
- **REQ-2.5** All product updates shall be reflected in the database.
- **REQ-2.6** When a product is deleted, it shall be removed from the database.

3.1.3 Inventory Display

- **REQ-3.1** The system shall display products in a tabular format.
- **REQ-3.2** The system shall display product ID, name, price, and quantity.
- **REQ-3.3** The display should be clear and easy to understand for all users.

3.1.4 Theme Management

- **REQ-4.1** The system shall have a default theme.
- **REQ-4.2** The system shall support multiple style sheets by using classes on body, nav and container.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

- **NFR-1.1** The system shall respond to user actions within 2 seconds on a standard web connection.
- **NFR-1.2** The system shall handle up to 10 concurrent users without significant performance degradation.

3.2.2 Security Requirements

- **NFR-2.1** The system shall protect user passwords by storing them in hashed format.
- **NFR-2.2** The system shall only allow authenticated users to perform product management.
- **NFR-2.3** The system shall prevent unauthorized access to the database.

3.2.3 Usability Requirements

- **NFR-3.1** The system shall have a clean and simple user interface.
- **NFR-3.2** The system shall be easy to navigate and use for non-technical users.
- **NFR-3.3** The system shall provide clear feedback to the user after each action.

3.2.4 Maintainability Requirements

- **NFR-4.1** The system shall be developed with clean and well-documented code.
- **NFR-4.2** The system shall be modular and easily extensible.

4 Diagrams Overview

This section provides an overview of the diagrams used in the system. You can navigate to each diagram by clicking the links below:

- Functional Diagrams
- Sequence Diagram
- Data Flow Diagram

Revision History

Version	Date	Author(s)	Description
1.0	2024-12-1	Toqa Ezzatly	A Very basic and proto-type
1.1	2024-12-1	Toqa Ezzatly	Proto-type with some edit option
1.2	2024-12-1	Toqa Ezzatly	UI design
2.0	2024-12-3	Toqa Ezzatly	First released version