# Software Requirements Specification (SRS) Document

**Project Name:** [Insert Project Name Here]
**Date:** [Insert Date of Submission]
**Student ID:** [Insert Student ID]
**Prepared By:** [Insert Student Name]

# Contents

# 1 Introduction

## 1.1 Purpose

State the **purpose** of this document. It should specify the requirements for the proposed software system, detailing its functions, capabilities, and constraints. This document is intended for **[e.g., development team, testing team, project manager]**.

## 1.2 Scope (Product Scope)

Identify the **software product** by name and explain what it will and won't do.

- **Included Features:** List the primary features (e.g., "The system will allow users to track inventory, generate monthly reports, and manage user roles.").
- **Excluded Features:** Clearly state what the system will *not* cover (e.g., "The system will not handle payments or integrate with external shipping providers in this phase.").

## 1.3 Definitions, Acronyms, and Abbreviations

Provide a comprehensive list of all technical terms, acronyms, and abbreviations used throughout the document (e.g., **SRS** - Software Requirements Specification, **API** - Application Programming Interface, **DB** - Database, **CRUD** - Create, Read, Update, Delete).

## 1.4 References

List all documents, sources, or literature used or referenced in the creation of this SRS (e.g., Project Proposal, Vision Document, relevant industry standards, or academic papers).

## 1.5 Document Overview

Briefly describe what the rest of the document contains (e.g., Section 2 describes the high-level context, and Section 3 details the specific functional and non-functional requirements).

# 2 Overall Description

## 2.1 Product Perspective

Describe how the proposed system fits into the larger operating environment.

- **System Interface:** Is this a completely new, stand-alone system, or does it integrate with existing systems? (If it integrates, name those systems.)
- **Hardware and Software:** Briefly describe the required hardware (e.g., standard PC/mobile device) and the operating system/browser requirements.

## 2.2 Product Functions

Provide a high-level summary of the major functions the product will perform. This is a general overview; detailed requirements come in Section 3. Use a numbered or bulleted list.

- **Example:** User Authentication (Login/Logout)

- **Example:** Data Management (CRUD operations on inventory records)
- **Example:** Reporting (Generate weekly summary)

### 2.3 User Characteristics

Describe the general characteristics of the intended users.

Table 1: User Characteristics

| User Type | Technical Skill Level | Typical Tasks Performed |
|---|---|---|
| **Admin** | High | User management, system configuration, data backup. |
| **Standard User** | Medium | Data entry, viewing reports, performing core business tasks. |
| **External Client** | Low | Viewing publicly available data, simple interaction. |

### 2.4 General Constraints

Identify any limitations or restrictions placed on the system, process, or development team.

- **Regulatory/Legal:** (e.g., Must comply with GDPR or HIPAA data privacy standards.)
- **Hardware Limitations:** (e.g., Must run on a minimum of 4GB RAM devices.)
- **Software Language:** (e.g., Must be developed using Python and React.)
- **Time:** (e.g., Must be deployed by the end of Q3.)

### 2.5 Assumptions and Dependencies

List all factors that, if changed, would impact the requirements in this document.

- **Assumption:** (e.g., The customer will provide the initial legacy data set in a CSV format.)
- **Dependency:** (e.g., Development of Feature X is dependent on the completion of the external API gateway.)

## 3 Specific Requirements

This section details all software requirements to the extent that a designer can create a final design, and a tester can verify the product. Requirements should be **Verifiable, Unambiguous, and Complete.**

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces (UI)

- Specify required screen elements (e.g., ”The system must present a dashboard view upon login.”).
- Specify required look-and-feel (e.g., ”The interface must be fully responsive and support both dark and light modes.”).

### 3.1.2 Hardware Interfaces

(e.g., The system must be able to interface with a USB barcode scanner.)

### 3.1.3 Software Interfaces

(e.g., The system must use the Google Maps API for location lookups.)

### 3.1.4 Communication Interfaces

(e.g., The system must support HTTPS protocol for all client-server communication.)

## 3.2 Functional Requirements

Functional requirements describe the actions the system **must** perform. Structure these using **Use Cases** or **User Stories**.

Table 2: Functional Requirements

| ID | Function/User Story | Description | Priority | Status |
|---|---|---|---|---|
| FR-001 | Create New User Account | As an Administrator, I want to create new user accounts by providing name, email, and assigning a role, so that new employees can access the system. | High | Approved |
| FR-002 | View Inventory Stock | As a Standard User, I want to view a real-time list of all inventory items, so I can check availability before fulfilling an order. | High | Approved |
| FR-003 | Generate PDF Report | As an Administrator, I want to generate a sales summary report for a specified date range in PDF format. | Medium | Approved |

*(Continue with as many requirements as necessary.)*

## 3.3 Non-Functional Requirements (NFRs)

NFRs define system quality attributes, often dictating *how* the system works.

### 3.3.1 Performance Requirements

- **Load Time:** (e.g., All pages must load within 2 seconds under peak load conditions (100 concurrent users).)
- **Transaction Rate:** (e.g., The system must process 50 inventory update transactions per minute.)

### 3.3.2 Security Requirements

- (e.g., The system must implement multi-factor authentication (MFA) for all administrative accounts.)
- (e.g., All sensitive data fields must be encrypted using AES-256.)

### 3.3.3 Reliability Requirements

- **Availability:** (e.g., The system must be available 99.5% of the time, excluding scheduled maintenance.)
- **Recovery:** (e.g., In the event of a system failure, the system must recover and restore the last saved state within 5 minutes.)

### 3.3.4 Maintainability Requirements

- (e.g., The code must adhere to the **PEP 8** style guide.)
- (e.g., The system must support hot-patching for minor bug fixes without requiring a full system restart.)

### 3.3.5 Portability Requirements

(e.g., The application must be runnable on Linux, Windows, and macOS operating systems.)

# 4 Supporting Information

## 4.1 Appendix A: Data Model and Class Diagram

Include diagrams or descriptions of the high-level data structures or the primary database schema (e.g., list of tables and their key fields). The **UML Class Diagram** should be included here to illustrate the static structure and relationships between key entities.

## 4.2 Appendix B: Use Case and System Diagrams

Include diagrams illustrating the major components of the system (e.g., client, server, database) and how they interact. This appendix is the appropriate location for key UML behavioral diagrams, such as the **Use Case Diagram**, **Sequence Diagrams** (for critical scenarios), and, if applicable, **Activity** or **State Diagrams**.