**Software Requirements Specification Document**

## **1. Overview**

The Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for the Petstore Catalog web application, which is hosted at <https://petstore.octoperf.com/actions/Catalog.action>. The SRS serves as a comprehensive guide for developers, testers, and stakeholders to understand the functionality, features, and expectations for the system.

### **1.1 Scope**

This SRS document focuses on the features and functionality of the Petstore Catalog, which is a web-based application for browsing pet products and services. The primary features include:

* Viewing product catalogs
* Searching and filtering products by categories and keywords
* Accessing product details such as description, price, and availability
* User authentication and session management (for logged-in users)
* Handling product orders, reviews, and ratings
* Payment gateway integration (if applicable)

This document will not cover aspects such as backend implementation details, security architecture, or specific third-party integrations unless specified.

## **2. References**

1. **IEEE Std 830-1998** - IEEE Recommended Practice for Software Requirements Specifications.
2. **Web Content Accessibility Guidelines (WCAG)** - WCAG 2.1
3. **ISO/IEC 9126** - Software Engineering – Product Quality.
4. **Petstore Catalog URL** - <https://petstore.octoperf.com/actions/Catalog.action>

## **3. Definitions**

* **Petstore Catalog**: The online catalog showcasing various pet products including food, toys, accessories, etc.
* **User**: A customer interacting with the catalog to view products and make purchases.
* **Administrator**: The individual responsible for managing the pet products and overseeing catalog updates.
* **Product**: An item in the catalog, which could be pet food, accessories, or services like grooming.

## **4. Considerations for Producing a Good SRS**

Creating an effective Software Requirements Specification (SRS) requires careful planning, collaboration, and attention to both technical and user needs.

### **4.1 Nature of the SRS**

The SRS is a formal document that describes the system requirements, both functional and non-functional, in a clear, concise, and verifiable manner. It should serve as a contract between stakeholders, developers, and testers.

### **4.2 Environment of the SRS**

The SRS should be produced in a collaborative environment involving stakeholders such as business analysts, developers, testers, and customers. The environment should facilitate easy access to the document, allowing for updates and versioning when necessary.

### **4.3 Characteristics of a Good SRS**

A good SRS document should be:

* **Unambiguous**: Each requirement must have only one interpretation.
* **Complete**: All possible scenarios should be addressed, with no gaps in functionality.
* **Consistent**: Requirements should not contradict each other.
* **Verifiable**: Each requirement must be testable.
* **Feasible**: The requirements must be achievable within the project constraints.
* **Modifiable**: The SRS should allow for easy updates and revisions.
* **Traceable**: Each requirement must have a clear origin and context.

### **4.4 Joint Preparation of the SRS**

The SRS should be a joint effort between the project team and stakeholders. Developers, testers, and business analysts should work together to create the document, ensuring that all perspectives are considered.

### **4.5 SRS Evolution**

The SRS should be treated as a living document. As the project evolves and new requirements emerge, the SRS should be updated to reflect these changes, while maintaining version control.

### **4.6 Prototyping**

A prototype of the system may be created and reviewed with stakeholders before finalizing the SRS. This can help clarify requirements and ensure that the system meets user expectations early in the process.

### **4.7 Embedding Design in the SRS**

While the SRS should mainly focus on requirements, certain high-level design elements can be embedded if they help stakeholders understand the expected architecture or flow of the system. However, the detailed design should be covered in a separate design document.

### **4.8 Embedding Project Requirements in the SRS**

The SRS should not only cover functional requirements but also include any relevant project requirements, such as performance constraints, security measures, and scalability needs. These should be clearly outlined in the non-functional requirements section.

## **5. The Parts of an SRS**

The SRS document for the Petstore Catalog application can be divided into the following sections:

### **5.1 Introduction (Section 1 of the SRS)**

This section provides an overview of the Petstore Catalog system, its scope, objectives, intended users, and a summary of the requirements. It sets the context for the entire document and includes:

* Purpose of the SRS
* Scope of the system
* Definitions, acronyms, and abbreviations
* References to external documents
* Overview of the structure of the SRS

### **5.2 Overall Description (Section 2 of the SRS)**

This section provides a high-level description of the Petstore Catalog, its key features, and the major components involved. It includes:

* **Product Perspective**: How the Petstore Catalog fits within the larger context (e.g., as part of a larger e-commerce platform).
* **User Classes and Characteristics**: Describing the different types of users (e.g., visitors, registered users, administrators).
* **System Features**: Overview of all significant system features and functions.
* **Assumptions and Dependencies**: Any external systems or conditions required for the application to function correctly.

### **5.3 Specific Requirements (Section 3 of the SRS)**

This section details the functional and non-functional requirements of the Petstore Catalog system. The functional requirements should describe what the system must do, such as:

* Product listing and search functionality
* Shopping cart management
* User authentication and order processing
* Payment gateway integration

The non-functional requirements should describe attributes such as:

* **Performance**: The system must handle X number of concurrent users.
* **Security**: User passwords must be encrypted.
* **Usability**: The UI should be responsive and intuitive.

### **5.4 Supporting Information**

This section may include any additional information or appendices that support the development and understanding of the SRS. This could include:

* Use case diagrams or flowcharts
* Glossary of terms
* Data models or entity-relationship diagrams
* Sample UI wireframes or mockups