Software Requirements Specification (SRS) For <Project Name>

Contents

1 Preface

1.1 Document Purpose

This document lists the requirements for the <Client Name> <Project Name> project. The purpose of this document is to identify the system requirements and obtain sign-off on all requirements before moving to the design phase. The Development team will use this document as the basis for the system design.

1.2 Revision History

This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.

Revision	Author	Date	Description

2 Introduction

2.1 Problem Statement

A problem statement is a concise and clear description of an issue or challenge that needs to be addressed. It is typically used at the beginning of a project or research effort to define the context and purpose of the work. A well-crafted problem statement helps to focus on the key issues, sets the direction for the project, and provides a basis for developing solutions.
Key components of a problem statement include:

2.1.1 The Problem:

Clearly define the problem or challenge that needs to be addressed. Be specific and avoid vague language.

2.1.2 Goals and Objectives:

Based on the problem description, describe the main goal(s) as the main of the system. It should be describe precisely using a set of Objectives which will be able to achieve the given goals.

2.1.3 The Scope:

Identify the boundaries of the problem. Specify what is included and what is excluded from the scope of the project.

2.1.4 The Impact:

Describe the consequences or implications of the problem. Explain why it is important to address this issue.

2.1.5 The Stakeholders:

Identify the individuals, groups, or entities that are affected by the problem. This helps in understanding the perspectives and interests of various stakeholders.

2.1.6 The Vision for a Solution:

Provide a high-level vision or goal for what the solution should achieve. This helps in aligning efforts towards a common objective.

2.1.7 Summary (Problem Statement)

Please see the example below,
"The current manual inventory tracking system in our organization is inefficient and error-prone. This leads to delays in order fulfillment, increased operational costs, and

customer dissatisfaction. The scope includes all inventory-related processes from receiving to shipping. Stakeholders affected by this problem include warehouse staff, customer service, and customers. The goal is to implement an automated inventory management system that improves accuracy, reduces processing time, and enhances overall operational efficiency."

<<*In software development, the problem statement often serves as a starting point for creating the Software Requirements Specification (SRS) document, which outlines the detailed requirements for the software solution*.>>

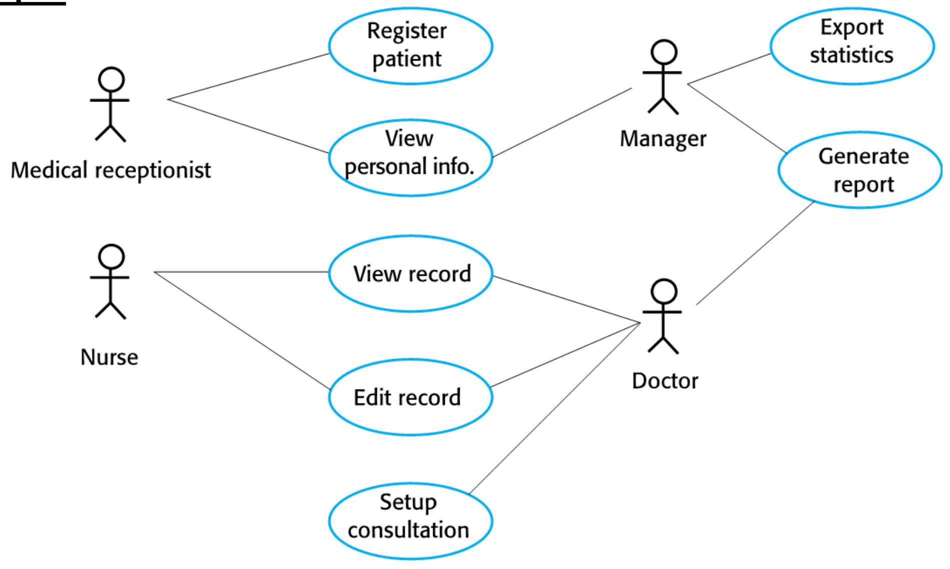
3 User Requirement Definition

Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.

Use Cases of the system

- Use-cases are a kind of scenario that are included in the UML.
- Use cases identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.

Example:



4 System Requirements Definition

The system requirements definition, provides a comprehensive overview of what the software system is expected to accomplish. It serves as a foundation for the entire development process. Here are key elements that should be described in the system requirements definition:

4.1 Functional Requirements:

Provide a detailed description of the system's functionalities, including both primary and secondary features.
Each functional requirement should be described using relevant use cases and its use case descriptions.

4.1.1 Use Case Description

A use case template is a standardized format or structure used to document information about a specific use case within a software system. Use cases describe interactions between the system and its users or other systems to achieve a specific goal. The template helps in organizing and presenting essential information about each use case in a consistent manner. While the exact structure may vary, a typical use case template includes the following elements:

A Template to describe use cases

1. **Use Case Name and ID:**
    - A clear and descriptive name for the use case that reflects its purpose.
2. **Scope:**
    - Defines the boundaries and context of the use case. It clarifies what the use case includes and what it does not.
3. **Primary Actor:**
    - Identifies the primary actor, which is typically the user or system initiating the use case.
4. **Stakeholders and Interests:**
    - Lists other stakeholders involved in or affected by the use case and their interests.
5. **Preconditions:**
    - Describes any conditions that must be true before the use case is initiated.
6. **Trigger:**
    - Specifies the event or condition that triggers the initiation of the use case.
7. **Main Flow:**
    - A step-by-step description of the normal or most common flow of events within the use case. This provides a narrative of the interactions between the actor and the system.
8. **Alternate Flows:**
    - Describes alternative paths or deviations from the main flow, such as error scenarios or exceptional cases.
9. **Exception Flows:**

   - Covers exceptional scenarios that may occur and how the system should handle them.

10. **Postconditions:**
    - Describes the state of the system after the successful completion of the use case.
11. **Success Endings:**
    - Describes the successful outcomes of the use case.
12. **Failure Endings:**
    - Describes the possible failure outcomes of the use case and how they should be handled.
13. **Special Requirements:**
    - Notes any special conditions, technology constraints, or other requirements specific to the use case

4.2 Non-Functional Requirements:

- Performance Requirements: Specify criteria related to response time, throughput, and resource usage.
- Security Requirements: Detail security measures, access controls, and data protection mechanisms.
- Reliability and Availability: Define the system's reliability expectations and any requirements for downtime.
- Usability: Describe user interface and user experience expectations.
- Scalability: Outline the system's ability to handle increased loads or users.
- Maintainability: Specify requirements related to code maintainability, updates, and modifications.
- System Constraints:
- Detail any constraints, such as hardware, software, or regulatory limitations.
- Specify compatibility requirements with existing systems or platforms.
- Assumptions and Dependencies:

Clearly state any assumptions made during the requirements definition.

4.3 Data Requirements:

Describe the types of data the system will handle.
Specify data formats, storage, retrieval, and data processing requirements.
You can use ER diagram to present the data requirement.

4.4 User Characteristics:

Define the intended users of the system and their characteristics.
Describe any specific user roles and their respective permissions.

4.5 Testing Requirements:

Outline the testing strategy, including test cases, scenarios, and acceptance criteria.
Specify any tools or testing environments needed.

4.6 Documentation Requirements:

Define documentation standards and specify the types of documentation required, such as user manuals, technical documentation, and training materials.
Legal and Compliance Requirements: