Modern SRS Package Template

This template provides an outline for a Modern Software Requirements Specification (SRS) Package applying both traditional document-based techniques and use-case modeling. For large systems, optional packaging is recommended at the feature (or other appropriate subsystem grouping) level. For example, if feature-level packaging is used, this specification will include or reference *all* software requirements related to the implementation of that feature. In some cases, a Modern SRS Package may be specified as one or more documents, for which this outline serves as an annotated template starting point. In other cases, the package is a logical construct that may consist of only one physical document, with references to other model- or tool-based physical representations (UML models, use cases, requirements tool repositories, or other) of the data described herein.

Company Name
Division Name, if appropriate

Project Name
Software Requirements Specification
Document Number, if appropriate

Revision History

Date	Revision	Description	Author
mm/dd/yyyy	1.0	Initial version	Author name

Table of Contents

1 Introduction

1.1 Purpose

Specify the purpose of this SRS. The SRS should fully describe the external behavior of the application or subsystem identified, as well as nonfunctional requirements, design constraints, and other factors necessary to provide a complete, comprehensive description of the software requirements.

1.2. Scope

This section provides a brief description of the software application that the SRS applies to, the features or other subsystem grouping, what use-case model(s) it is associated with, and anything else that is affected or influenced by this document.

1.3 References

Provide a list of project-related references or applicable documents that bear on this project.

1.4 Assumptions and Dependencies

This section describes any key technical feasibility, subsystem, or component availability or other project-related assumptions on which the viability of the software described by this SRS may be based.

2 Use-Case Model Survey

This section provides an overview of the use-case model. The survey is used by people interested in the behavior of the system, such as the customer, users, architects, use case authors, designers, use case designers, testers, managers, reviewers, and writers. This section lists for each use case

- The use case name.
- A brief description explaining the use case's function and role in the system.
- A list of actors for the use case (Primary and secondary, active and passive). The aggregation of these actors is further defined in the accompanying actor survey.
- Diagram of the use-case model. A diagram of the entire use-case model is included here.

3 Actor Survey

All of the actors mentioned in the use-case model survey are reported here. For each actor, you should list

- The actor's name
- A brief description of the actor

4 Requirements

4.1 Functional Requirements

This section describes the functional requirements of the system for those requirements that are expressed in the natural-language style.
For many applications, this may constitute the bulk of the package, and thought should be given to the organization of this section. This section is typically organized by feature, but alternative organization methods, by user or by subsystem, may also be appropriate.
Where application development tools (requirements tools, modeling tools, and so on) are used to capture the functionality, this section of the document will refer to the availability of that data and will indicate the location and name of the tool used to capture the data.

4.2 Nonfunctional Requirements

Most nonfunctional requirements are typically recorded in natural language in this section of the specification. However, nonfunctional requirements may also be included with a specific use case specification.

4.2.1 Usability

This section should include all of those requirements that affect usability. These often include:

- Specify the required training time for normal users and power users to become productive at particular operations.

- Specify measurable task times for typical tasks; alternatively, base usability requirements of the new system on other systems that the users know and like.

- Specify requirements to conform to common usability standards, such as IBM's CUA standards or the GUI standards published by Microsoft for Windows 98/2000/XP.

4.2.2 Reliability

Requirements for system reliability should be specified here.

- Availability: Specify percent of time available (xx.xx%), hours of use, maintenance access, degraded-mode operations, and so on.

- Mean time between failures (MTBF): This is usually specified in hours but could also be specified in terms of days, months, or years.

- Mean time to repair (MTTR): How long is the system allowed to be out of operation after it has failed?

- Accuracy: Specify precision (resolution) and accuracy (by some known standard) that is required in the system's output.

- Maximum bugs or defect rate: Usually expressed in terms of bugs/KLOC (thousands of lines of code) or bugs per function-point.

- Bugs or defect levels: Categorized in terms of minor, significant, and critical bugs. The requirement(s) must define what is meant by a "critical" bug (such as complete loss of data or complete inability to use certain parts of the functionality of the system).

4.2.3 Performance

The performance characteristics of the system should be outlined in this section. Include specific response times. Where applicable, reference related use cases by name.

- Response time for a transaction (average, maximum)

- Throughput (transactions per second)

- Capacity (the number of customers or transactions the system can accommodate)

- Degradation modes (the acceptable mode of operation when the system has been degraded)

- Resource utilization (memory, disk, communications)

4.2.4 Supportability

This section indicates any requirements that will enhance the supportability or maintainability of the system being built, including coding standards, naming conventions, class libraries, maintenance access, and maintenance utilities.

5 Online User Documentation and Help System Requirements

Describes the requirements, if any, for online user documentation, help systems, help notices, and so on.

6 Design Constraints

This section should indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, and class libraries.

7 Purchased Components

This section describes any purchased components to be used with the system, any applicable licensing or usage restrictions, and any associated compatibility/interoperability or interface standards.

8 Interfaces

This section defines the interfaces that must be supported by the application. This section should contain adequate specificity, protocols, ports, and logical addresses, and so on, so that the software can be developed and verified against the interface requirements.

8.1 User Interfaces

Describe the user interfaces that are to be implemented by the software.

8.2 Hardware Interfaces

Define any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, and expected behavior.

8.3 Software Interfaces

Describe software interfaces to other components of the software system. These may be purchased components, components reused from another application, or components being developed for subsystems outside of the scope of this SRS but with which this software application must interact.

8.4 Communications Interfaces

Describe any communications interfaces to other systems or devices, such as local area networks or remote serial devices.

9 Licensing Requirements

Define any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.

10 Legal, Copyright, and Other Notices

Describe any necessary legal disclaimers, warranties, copyright notices, patent notice, wordmark, trademark, or logo compliance issues for the software.

11 Applicable Standards

Describe by reference any standards (and the specific sections of any such standards) that apply to the system being described. For example, this could include legal, quality, and regulatory standards, as well as industry standards for usability, interoperability, internationalization, operating system compliance, and so on.

Index

The index is provided to assist the reader in locating key concepts and topics that occur throughout the document.

Glossary

Describe any terms that are unique to this application context and any definitions, acronyms, abbreviations, or other project or company-specific shorthand that is necessary for an understanding of this document and the application.

Appendixes

You should insert appendixes here as appropriate. Note that the following template appendix is provided specifically to allow you to record use cases. Feel free to insert as many appendixes as you need.

Appendix: Use Case Specifications

This appendix contains references the elaborated use cases for the system. The following template is provided as a starting point.

Revision History

Date	Issue	Description	Author
dd/mmm/yy	x.x	Details	Author name

Note that the revision history is provided for each use case included in the appendixes. The current revision history block should be on the first page of each use case appendix.

Table of Contents

Normally, a use case specification will not be long enough to warrant a table of contents for the use case. But this element may be required if the use case presents unusual problems in finding portions of the specification.

Use Case Name

Brief Description

The role and purpose of the use case: A single paragraph should suffice for this description.

Participating Actors (Primary, secondary, active and non-activet)

Insert the name only, and refer to the ID number of the actor in this document

Actor ID	Actor Name	Role (Primary, secondary)	Active/Non-Active

Pre-conditions (Numbered List)

Precondition of a use case is the state of the system that must be present prior to a use case being performed.

Flow of Events

Basic Flow

This use case starts when the actor does something. An actor always initiates use cases. The use case should describe what the actor does and what the system does in response. The use case should be phrased in the form of a dialogue between the actor and the system.

Actor A	System	Actor B
1		
	2 2.1 2.2	
		3

The use case should describe what happens inside the system but not how or why.

A glossary is often useful to keep the complexity of the use case manageable; you may want to define customer information there, to keep the use case from drowning in details.

Simple alternatives may be presented within the text of the use case. If it takes only a few sentences to describe what happens when there is an alternative, do it directly within the flow-of-events section. If the alternative flows are more complex, use a separate section. For example, an alternative flow describes how to describe more complex alternatives.

Use notation: A1 A2, to represent Alternative flow 1 or Alternative flow 2 respectively.

A picture is sometimes worth a thousand words, although there is no substitute for clean, clear prose. If doing so improves clarity, feel free to include graphical depictions of user interfaces, process flows, or other figures into the use case.
If a technical method, such as an activity diagram is useful to present a complex decision process, by all means use it!
Similarly for state-dependent behavior, a state-transition diagram often clarifies the behavior of a system better than do pages upon pages of text.
Use the right presentation medium for your problem, but be wary of using terminology, notation, or figures that your audience may not understand.
Remember that your purpose is to clarify, not to obscure.

Alternative Flows
1. **First alternative flow:** More complex alternatives should be described in a separate section, which is referred to in the basic flow-of-events section. Think of the alternative flow sections as *alternative behavior;* each alternative flow represents alternative behavior (many times, because of exceptions that occur in the main flow). They may be as long as necessary to describe the events associated with the alternative behavior. When an alternative flow ends, the events of the main flow of events are resumed unless otherwise stated.
Alternative flows may, in turn, be broken down into subsections.

2. **Second alternative flow:** There may be, and most likely will be, a number of alternative flows in a use case. Keep each alternative separate, to improve clarity. Using alternative flows improves the readability of the use case, as well as prevents use cases from being *decomposed* into hierarchies of use cases. Keep in mind that use cases are just textual descriptions and that their main purpose is to document the behavior of a system in a clear, concise, and understandable way.

Special Requirements (Numbered List)
These are typically nonfunctional requirements that are specific to a use case but are not easily or naturally specified in the text of the use case's event flow. Examples of special requirements include legal and regulatory requirements, application standards, and quality attributes of the system to be built, including *usability reliability, performance, or supportability requirements.* Other requirements, such as operating systems and environments, compatibility requirements, and design constraints, should also be captured in this section.

Post-conditions (Numbered List)
Post-condition of a use case is a list of possible states the system can be in immediately after a use case has finished.

Extension Points (Numbered List)
Extension points of the use case.
Name of extension point: Definition of the location of the extension point in the flow of events.