(Project Title)
(Company Name)
(Participants)

OCP Software Requirements Specification

Document

Version: (n) **Date: (mm/dd/yyyy)**

Table of Contents

Versions Table

No	Name	Date	Comment
1.	Rajeev Sharma	07/12/20201	Created template for the software specifications
2.	TBD	TBD	TBD

1. Introduction

Please provide overview of entire software requirement specifications. Please explain what the system must do, so that the community developers can build it.

1.1 Purpose

Identify the purpose and the intended audience.

1.2 Scope

What is the software product?

What is it called?

What this software will do?

What this software will not do?

What are the objectives and overall goals of this software?

An executive summary is enough. Please do not enumerate the whole feature list here.

1.3 Definitions, Acronyms and Abbreviations

This information is required to properly interpret the software specifications.

1.4 References

Please provide information or link on references like software protocols, RFCs etc. so that developers know from where to find them.

1.5 Overview

Explain how this software specification document is structured. Please do not past table of contents here. Point people to the parts of the document they are most concerned with. Customers/potential users care about section 2, developers care about section 3.

2. Overall Description

Explain requirements from a very high level which is very easy to understand.

2.1 Product Perspective

Is this product self-contained or a part of a larger system? A block diagram showing the major components of the larger system, interconnections, and external interfaces would be helpful.

2.1.1 System interfaces

These are external system interfaces that the product interacts with. An example could be APIs used to get information from an external interface.

2.1.2 Interfaces

This is a description of how the system will interact with its users. Is there a GUI, a command line or some other type of interface? Are there special interface requirements?

2.1.3 Hardware Interfaces

What hardware platforms are supported? Developers should be able to look at this and know what hardware they need to worry about in the design.

2.1.4 Software Interfaces

Please specify the use of other required software products and interfaces with other application systems. Any software dependencies must be listed here.

2.1.5 Communication Interfaces

Please specify various interfaces to communications such as local network protocols etc. These are the protocols that the software needs to directly interact with. If it is a standard protocol, you can reference an existing document or RFC.

2.1.6 Memory constraints

Please mention memory limitations (e.g., 256-512M of RAM) if any.

2.2 Product Functions

Provide a summary of the major functions that the software will perform.

For clarity:

- The functions should be organized in a way that makes the list of functions understandable to the OCP adopter or the end user.

- Textual or graphic methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product but simply shows the logical relationships among variables.

Please remember that this is a description of what the system needs to do, not how you are going to build it. (That comes in the design document).

2.3 User Characteristics

Please describe general characteristics of the intended users of the product including experience and technical expertise.

What is it about your potential user base that will impact the design? Their experience and comfort with technology will drive UI design. Other characteristics might actually influence internal design of the system.

2.4 Constraints

Please provide a general description of any items that limits the developers' options. These can include but not limited to: -

- Regulatory policies
- Hardware limitations
- Interface to other applications
- Reliability requirements
- Security considerations

This section captures the non-functional requirements.

3. Specific Requirements

This section contains all the software requirements at a level of detail sufficient to enable developers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (request) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:

- Specific requirements should be stated with all the characteristics of a good software specification.

    - Correct
    - Unambiguous
    - Complete
    - Consistent
    - Ranked for importance and/or stability
    - Verifiable
    - Modifiable
    - Traceable

- All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)

Remember this is not design. Do not require specific software packages, etc. unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:

The system shall… *A required, must have feature*
The system should… *A desired feature, but may be deferred till later*
The system may… *An optional, nice-to-have feature that may never make it to implementation.*

3.1 External Interfaces

This section contains a detailed description of all the inputs into and outputs from the software system. It complements the interface description in Section-2, but please do not repeat information here.

Please remember that section-2 presents information orientated to the OCP adopters/end-users while Section-3 is oriented to the system developers.

It contains both content and format as follows:

- Name of the item
- Description of purpose
- Source of input or destination of output
- Valid range, accuracy and/or tolerance
- Units of measure

- Timing
- Relationships to other inputs/outputs
- Data formats
- Command formats
- End messages

3.2 Functions

Functional requirements define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as "*shall*" statements starting with "The system shall…

These include:

- Validity checks on the inputs
- Exact sequence of operations
- Responses to abnormal situation, including
    - Overflow
    - Error handling and recovery
- Effect of parameters
- Relationship of outputs to inputs, including
    - Input/Output sequences
    - Formulas for input to output conversion

It may be appropriate to partition the functional requirements into sub-functions or sub-processes.

3.3 Performance Requirements

This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole.

Static numerical requirements may include:

- The number of terminals to be supported
- The number of simultaneous users to be supported
- Amount and type of information to be handled

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

95% of the telemetry information shall be processed in less than 1 second

rather than,

An operator shall not have to wait for the telemetry request to complete.

3.4 Logical Database requirements (Optional)

This optional section specifies the logical requirements for any information that is to be placed into a database. This may include:

- Types of information used by various functions
- Frequency of use
- Accessing capabilities
- Data entities and their relationships
- Integrity constraints
- Data retention requirements

Data models can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships.

3.5 Design Constraints

Specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.6 Software System Attributes

It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples.

These are also known as non-functional requirements or quality attributes.

These requirements have to be testable just like the functional requirements

3.6.1 Reliability

Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF (Mean Time Before Failure) requirements, express them here.

3.6.2 Availability

Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. What are the requirements for system recovery from a failure?

3.6.3 Security

Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:

- Utilize certain cryptographic techniques
- Keep specific log or history data sets
- Assign certain functions to different modules
- Restrict communications between some areas of the program
- Check data integrity for critical variables

3.6.4 Maintainability

Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements, someone else will maintain the system

3.6.5 Portability

Specify attributes of software that relate to the ease of porting the software to other platforms and/or operating systems. This may include:

- Percentage of components with host-dependent code
- Percentage of code that is host dependent
- Use of a proven portable language
- Use of a particular compiler or language subset
- Use of a particular operating system

4. Change Management Process

Identify the change management process to be used to identify, log, evaluate, and update the software requirement specification to reflect changes in project scope and requirements.

How are you going to control changes to the requirements?
Does the collaborating team of companies have to reach consensus?
How do changes to requirements get submitted to the team? Formally in writing, email or phone call? Please list all this process here.

5. Document Approvals

Identify the approvers of the software specification document. Approver name, signature, and date should be used.

6. Supporting Information

The supporting information makes the SRS easier to use. It includes:

- Index
- Appendices

The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:

- Sample I/O formats, descriptions of cost analysis studies, results of user surveys

- Supporting or background information that can help the readers of the software specification

- A description of the problems to be solved by the software.

- Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements

When Appendices are included, the software specification should explicitly state whether or not the Appendices are to be considered part of the requirements.