

# **Software Requirements Specification**

## **Document**

(Version  
**No.2) for**

**<Project Title>**

**Version No.**

**Date:**

**Prepared by:(Name & Address of the Organisation/Department)**

**For Department of:(User Department)**

**Submitted to:**

## Document Release Notice

**Notice No. Client**

**Project Document**

**Details**

Document ID	Version No.	Description

**APPROVED BY:**(Approval Authority)  
(Name, Signature & Designation)

**DATE:**



NOTE: The table below is not a part of the standard document. It is an indicative of the mandatory fields to be filled in case of different project levels.

Sr. No.	Description.	Page No.	Mandatory Fields for	
			Large Scale Project	Small Scale Project
<b>1</b>	<b>Introduction</b>	<b>7</b>	<b>X</b>	<b>X</b>
1.1	Purpose	<b>7</b>	<b>X</b>	<b>X</b>
1.2	Scope	<b>7</b>	<b>X</b>	<b>X</b>
1.3	Definitions, Acronyms, and Abbreviations	<b>7</b>	<b>X</b>	<b>X</b>
1.4	References	<b>7</b>	<b>X</b>	<b>X</b>
1.5	Overview	<b>7</b>	<b>X</b>	<b>X</b>
<b>2</b>	<b>The Overall Description</b>	<b>8</b>	<b>X</b>	<b>X</b>
2.1	Application Software Perspective	<b>8</b>	<b>X</b>	<b>X</b>
2.1.1	System Interfaces	<b>8</b>	<b>X</b>	<b>X</b>
2.1.2	User Interfaces	<b>8</b>	<b>X</b>	<b>X</b>
2.1.3	Hardware Interfaces	<b>9</b>	<b>X</b>	
2.1.4	Software Interfaces	<b>9</b>	<b>X</b>	<b>X</b>
2.1.5	Communications Interfaces	<b>9</b>	<b>X</b>	
2.1.6	Memory Constraints	<b>10</b>	<b>X</b>	<b>X</b>
2.1.7	Operations	<b>10</b>	<b>X</b>	<b>X</b>
2.1.8	Site Adaptation Requirements	<b>10</b>	<b>X</b>	<b>X</b>
2.2	Application Software Functions	<b>11</b>	<b>X</b>	<b>X</b>
2.3	User Characteristics	<b>11</b>	<b>X</b>	
2.4	Constraints	<b>11</b>	<b>X</b>	
2.5	Assumptions and Dependencies	<b>12</b>	<b>X</b>	
2.6	Apportioning of Requirements	<b>12</b>	<b>X</b>	
<b>3</b>	<b>Specific Requirements</b>	<b>12</b>	<b>X</b>	<b>X</b>
3.1	External Interfaces	<b>13</b>	<b>X</b>	<b>X</b>
3.2	Functional requirements	<b>14</b>	<b>X</b>	<b>X</b>
3.3	Performance Requirements	<b>14</b>	<b>X</b>	
3.4	Logical Database Requirements	<b>15</b>	<b>X</b>	
3.5	Design Constraints	<b>15</b>	<b>X</b>	<b>X</b>
3.5.1	Standards Compliance	<b>16</b>	<b>X</b>	<b>X</b>
3.6	System Attributes	<b>16</b>	<b>X</b>	
3.6.1	Reliability	<b>16</b>	<b>X</b>	
3.6.2	Availability	<b>16</b>	<b>X</b>	
3.6.3	Security	<b>16</b>	<b>X</b>	<b>X</b>
3.6.4	Maintability	<b>17</b>	<b>X</b>	

*Software Requirements Specifications Document*

3.6.5	Portability	<b>17</b>	<b>X</b>	<b>X</b>
4	<b>Change Management Process</b>	<b>18</b>	<b>X</b>	<b>X</b>
5	<b>Supporting Information</b>	<b>19</b>	<b>X</b>	<b>X</b>

## Table of Contents

<b>1.</b>	<b>7</b>
<b>Introduction</b>	
1.1 Purpose	7
1.2 Scope	7
1.3 Definitions, Acronyms, and Abbreviations	7
1.4 References	7
1.5 Overview	7
<b>2. The Overall Description</b>	<b>8</b>
2.1 Application Software Perspective	8
2.1.1 System Interfaces	8
2.1.2 User Interfaces	8
2.1.3 Hardware Interfaces	9
2.1.4 Software Interfaces	9
2.1.5 Communications Interfaces	9
2.1.6 Memory Constraints	10
2.1.7 Operations	10
2.1.8 Site Adaptation Requirements	10
2.2 Application Software Functions	11
2.3 User Characteristics	11
2.4 Constraints	11
2.5 Assumptions and Dependencies	12
2.6 Apportioning of Requirements	12
<b>3. Specific Requirements</b>	<b>12</b>
3.1 External interfaces	13
3.2 Functional Requirements	14
3.3 Performance Requirements	14
3.4 Logical Database Requirements	15
3.5 Design Constraints	15
3.5.1 Standards Compliance	16
3.6 Software System Attributes	16
3.6.1 Reliability	16
3.6.2 Availability	16
3.6.3 Security	16
3.6.4 Maintainability	17
3.6.5 Portability	17
3.7 Organizing the Specific Requirements by Functional Hierarchy (Not a section of standard document)	18

3.8 Additional Comments (Not a section of Standard document)	18
<b>4. Change Management Process</b>	<b>18</b>
<b>5. Supporting Information</b>	<b>19</b>

## **1. Introduction**

### **1.1 Purpose**

Identify the purpose of this SRS and its intended audience. In this subsection, describe the purpose of the particular SRS and specify the intended audience for the SRS.

### **1.2 Scope**

In this subsection:

- (1) Identify the Application Software(s) to be produced by name (eg. Host DBMS, Report Generator, web Application)
- (2) Explain what the Application Software(s) will, and, if necessary, will not do
- (3) Describe the application of the software being specified, including relevant benefits, objectives, and goals

This should be an executive-level summary.

### **1.3 Definitions, Acronyms, and Abbreviations.**

Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendices in the SRS or by reference to documents. This information may be provided by reference to an Appendix.

### **1.4 References**

In this subsection:

- (1) Provide a complete list of all documents referenced elsewhere in the SRS
- (2) Identify each document by title, report number (if applicable), date, and publishing organization
- (3) Specify the sources from which the references can be obtained.

This information can be provided by reference to an appendix or to another document. If the application uses specific protocols, then reference them here so designers know where to find them.

### **1.5 Overview**

In this subsection:

- (1) Describe what the rest of the SRS contains
- (2) Explain how the SRS is organized

Don't rehash the table of contents here. Point out to the main parts of the document. (Clients/potential users care about section 2, developers care about section 3.)

## **2. The Overall Description**

Describe the general factors that affect the Application Software and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in section 3, and makes them easier to understand. In a sense, this section tells the requirements in non-technical language for the consumption of the client. Section 3 will contain a specification written for the developers.

### **2.1 Application Software Perspective**

Put the Application Software into perspective with other related Application Softwares. If the Application Software is independent and totally self-contained, it should be so stated here. If the SRS defines a Application Software that is a component of a larger system, then this subsection relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software. Compare the Application Software's similarities and differences to other systems in the marketplace.

The system you are building should be shown as a black box. The design document presents the internals.

The following subsections describe how the software operates inside various constraints.

#### **2.1.1 System Interfaces**

List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system. These are external systems that the software interacts with. For instance, if you are building a business application that interfaces with the existing employee payroll system, what is the API (Application Program Interface) to that system that designer's will need to use?

#### **2.1.2 User Interfaces**

Specify:

- (1) The logical characteristics of each interface between the Application Software and its users. This includes those configuration characteristics (eg. Required screen formats, page or window layouts, content of any reports or menus, or availability of any function keys) necessary to accomplish the software requirements.

- (2) All the aspects of optimizing the interface with the person who must use the system. 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**

Specify the logical characteristics of each interface between the Application Software and the hardware components of the system. This includes configuration characteristics. (eg. No. of ports, instruction sets etc.) It also covers such matters as what devices are to be supported, how they are to be supported and protocols. This is not a description of hardware requirements in the sense that "This program must run on a Windows platform with 64M of RAM". This section is for detailing the actual hardware devices your application will interact with and control. For instance, if you are controlling biometric devices, what is the interface to those devices? Designers should be able to look at this and know what hardware they need to worry about in the design. Many business type applications will have no hardware interfaces. If none, just state "The system has no hardware interface requirements"

### **2.1.4 Software Interfaces**

Specify the use of other required Application Softwares and interfaces with other application systems. For each required Application Software, include:

- (1) Name
- (2) Mnemonic
- (3) Specification number
- (4) Version number
- (5) Source

For each interface, provide the following:

- (1) Discussion of the purpose of the interfacing software as related to this Application Software
- (2) Definition of the interface in terms of message content and format

Here we document the APIs, versions of software that we do not have to write, but that our system has to use. For instance if your client uses SQL Server 7 and you are required to use that, then you need to specify i.e. 2.1.4.1 Microsoft SQL Server 7. The system must use SQL Server as its database component. Communication with the DB is through ODBC connections. The system must provide SQL data table definitions to be provided to the Organisation DBA for setup.

### **2.1.5 Communications Interfaces**

Specify the various interfaces to communications such as local network protocols, etc. These are protocols you will need to directly interact with.  
If

you are using a custom protocol to communicate between systems, then document that protocol here so designers know what to design. If it is a standard protocol, you can reference an existing document.

### **2.1.6 Memory Constraints**

Specify any applicable characteristics and limits on primary and secondary memory. If all the client's machines have only 128K of RAM, then your target design has got to come in under 128K so there is an actual requirement. You could also cite market research here for shrink-wrap type applications "Focus groups have determined that our target market has between 256-512MB of RAM, therefore the design footprint should not exceed 256M." If there are no memory constraints, so state.

### **2.1.7 Operations**

Specify the normal and special operations required by the user such as:

- (1) The various modes of operations in the user organization (eg. User initiated operations)
- (2) Periods of interactive operations and periods of unattended operations
- (3) Data processing support functions
- (4) Backup and recovery operations

(Note: This is sometimes specified as part of the User Interfaces section.) If you separate this from the UI, then cover business process types that would impact the design. For instance, if the organisation brings all their systems down at midnight for data backup that might impact the design. These are all the work tasks that impact the design of an application, but which might not be located in software.

### **2.1.8 Site Adaptation Requirements**

In this section:

- (1) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode
- (2) Specify the site or mission-related features that should be modified to adapt the software to a particular installation

If any modifications to the client's work area would be required by your system, then document that here. For instance, "A 100Kw backup generator and 10000 BTU air conditioning system must be installed at the user site prior to software installation".

This could also be software-specific like, "New data tables created for this system must be installed on the organisation's existing DB server and populated prior to system activation." Any equipment the client would need to buy or any software setup that needs to be done so that your system will install and operate correctly should be documented here.

## **2.2 Application Software Functions**

Provide a summary of the major functions that the software will perform. (eg. For an Accounting Software, the Application Software functions can be client account maintenance, client statement and invoice preparation)

For clarity:

- (1) The functions should be organized in a way that makes the list of functions understandable to the client or to anyone else reading the document for the first time.
- (2) 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 Application Software but simply shows the logical relationships among variables.

This describes the functionality of the system in the language of the client. What specifically does the system that will be designed have to do? This is a description of what the system needs to do, not how it is to be built.

## **2.3 User Characteristics**

Describe general characteristics of the intended users of the Application Software including educational level, experience, and technical expertise. Do not state specific requirements but rather provide the reasons why certain specific requirements are later specified in section 3.

It is about the potential user base that will impact the design? Their experience and comfort with technology will drive UI(User Interface) design. Other characteristics might actually influence internal design of the system.

## **2.4 Constraints**

Provide a general description of any other items that will limit the developer's options. These can include:

- (1) Regulatory policies
- (2) Hardware limitations (for example, signal timing requirements)
- (3) Interface to other applications
- (4) Parallel operation
- (5) Audit functions
- (6) Control functions
- (7) Higher-order language requirements
- (8) Reliability requirements
- (10) Criticality of the application
- (11) Safety and security considerations

This section captures non-functional requirements in the clients language.

## **2.5 Assumptions and Dependencies**

List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system would be available on the hardware designated for the Application Software. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

This section is catch-all for everything else that might influence the design of the system and that did not fit in any of the categories above.

## **2.6 Apportioning of Requirements.**

Identify requirements that may be delayed until future versions of the system. After you look at the project plan and hours available, you may realize that you just cannot get everything done. This section divides the requirements into different sections for development and delivery. Check with the client - they should prioritize the requirements and decide what does and does not get done. This can also be useful if you are using an iterative life cycle model to specify which requirements will map to which iteration.

## **3. Specific Requirements**

This section contains all the software requirements at a level of detail sufficient to enable designers 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 (stimulus) 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:

- (1) Specific requirements should be stated with all the characteristics of a good SRS
  - correct
  - unambiguous
  - complete
  - consistent
  - ranked for importance and/or stability
  - verifiable
  - modifiable
  - traceable
- (2) Specific requirements should be cross-referenced to earlier documents that relate
- (3) All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)
- (4) Careful attention should be given to organising the requirements to maximize readability

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

The system shall... A required, must have feature

The system should... A desired feature, but may be deferred til later

The system may... An optional, nice-to-have feature that may never make it to implementation.

Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, "The system shall be easy to use".

Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the client again. Don't say things like, "The system shall accept configuration information such as name and address." The designer doesn't know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.

### **3.1 External Interfaces**

This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there. Remember section 2 presents information oriented to the client/user while section 3 is oriented to the developer.

It contains both content and format as follows:

- Name of 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
- Screen formats/organization
- Window formats/organization
- Data formats
- Command formats
- End messages

### **3.2 Functional Requirements**

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
  - Communication facilities
  - 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. This does not imply that the software design will also be partitioned that way. For Functional requirement, the following information may be listed:

- Information flows
  - Data flow diagram
    - o Data entities
    - o Pertinent processes o Topology
- Process descriptions
  - Process
    - o Input data entities
    - o Algorithm or formula of process
    - o Outout data entities
- Data construct specifications
  - Construct
    - o Record type
    - o Constituent fields
- Data dictionary (MetaData)
  - Data element
    - o Name
    - o Representation
    - o Units/Format
    - o Precision/Accuracy
    - o Range
- Interface with other activities in the system
- Interface with other systems.

### **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:

- (a) The number of users to be supported
- (b) The number of simultaneous Average and Maximum users to be supported
- (c) Amount and type of information to be handled

Static numerical requirements are sometimes identified under a separate section entitled capacity.

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 transactions shall be processed in less than 3 second

rather than,

An operator shall not have to wait for the transaction to complete.

(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)

### **3.4 Logical Database Requirements**

This 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

If the client provided you with data models, those 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.

### **3.5.1 Standards Compliance**

Specify the requirements derived from existing standards or regulations. They might include:

- (1) Report format
- (2) Data naming
- (3) Accounting procedures
- (4) Audit Tracing

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

### **3.6 Software System Attributes**

There are a number of attributes of software that can serve as requirements. It is important that required attributes be 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 are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements.

#### **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. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? "The system shall allow users to restart the application after failure with the loss of at most 12 characters of input".

#### **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



Definitions of the quality characteristics follow.

- Correctness - extent to which program satisfies specifications, fulfills user's mission objectives
- Efficiency - amount of computing resources and code required to perform function
- Flexibility - effort needed to modify operational program
- Interoperability - effort needed to couple one system with another
- Reliability - extent to which program performs with required precision
- Reusability - extent to which it can be reused in another application
- Testability - effort needed to test to ensure performs as intended
- Usability - effort required to learn, operate, prepare input, and interpret output

THE FOLLOWING (3.7) is not a section of the SRS, it is talking about how to organize requirements you write in section 3.2.

### **3.7 Organizing the Specific Requirements**

Detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a proper functional hierarchy.

#### **3.7.1 Functional Hierarchy**

The overall functionality can be organized into a hierarchy of functions organized by common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data.

### **3.8 Additional Comments** (Not a section of Standard document)

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For example, when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.

In any of the outlines below, those sections called "Functional Requirement i" may be described in native language, in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, and Outputs.

## **4. Change Management Process**

Identify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements. How are you going to control changes to the requirements. Can the client just call up and ask for something new? Does your team have to reach consensus? How do changes to requirements get submitted to the team? Formally in writing, email or phone call?

## **5. Supporting Information**

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

- Table of Contents
- Index
- Appendices

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

- (a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys
- (b) Supporting or background information that can help the readers of the SRS like E-R diagrams, Data Flow diagrams, Data dictionary, etc.
- (c) A description of the problems to be solved by the software
- (d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements

If Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.

Table on the following pages provides way to structure section 3 on the specific requirements.

**Outline for SRS Section 3  
Organised by functional  
hierarchy**

- 3 Specific Requirements
  - 3.1 External interface requirements
    - 3.1.1 User interfaces
    - 3.1.2 Hardware interfaces
    - 3.1.3 Software interfaces
    - 3.1.4 Communications interfaces
  - 3.2 Functional requirements
    - 3.2.1 Information flows
      - 3.2.1.1 Data flow diagram 1
        - 3.2.1.1.1 Data entities
        - 3.2.1.1.2 Pertinent processes
        - 3.2.1.1.3 Topology
      - 3.2.1.2 Data flow diagram 2
        - 3.2.1.2.1 Data entities
        - 3.2.1.2.2 Pertinent processes
        - 3.2.1.2.3 Topology
      - 3.2.1.*n* Data flow diagram *n*
        - 3.2.1.*n*.1 Data entities
        - 3.2.1.*n*.2 Pertinent processes
        - 3.2.1.*n*.3 Topology
    - 3.2.2 Process descriptions
      - 3.2.2.1 Process 1
        - 3.2.2.1.1 Input data entities
        - 3.2.2.1.2 Algorithm or formula of process
        - 3.2.2.1.3 Affected data entities
      - 3.2.2.2 Process 2
        - 3.2.2.2.1 Input data entities
        - 3.2.2.2.2 Algorithm or formula of process
        - 3.2.2.2.3 Affected data entities
      - 3.2.2.*m* Process *m*
        - 3.2.2.*m*.1 Input data entities
        - 3.2.2.*m*.2 Algorithm or formula of process
        - 3.2.2.*m*.3 Affected data entities
    - 3.2.3 Data construct specifications
      - 3.2.3.1 Construct 1
        - 3.2.3.1.1 Record type
        - 3.2.3.1.2 Constituent fields
      - 3.2.3.2 Construct 2
        - 3.2.3.2.1 Record type
        - 3.2.3.2.2 Constituent fields
      - 3.2.3.*p* Construct *p*
        - 3.2.3.*p*.1 Record type
        - 3.2.3.*p*.2 Constituent fields

## *Software Requirements Specifications Document*

- 3.2.4 Data dictionary
  - 3.2.4.1 Data element 1
    - 3.2.4.1.1 Name
    - 3.2.4.1.2 Representation
    - 3.2.4.1.3 Units/Format
    - 3.2.4.1.4 Precision/Accuracy
    - 3.2.4.1.5 Range
  - 3.2.4.2 Data element 2
    - 3.2.4.2.1 Name
    - 3.2.4.2.2 Representation
    - 3.2.4.2.3 Units/Format
    - 3.2.4.2.4 Precision/Accuracy
    - 3.2.4.2.5 Range
  
  - 3.2.4.q Data element *q*
    - 3.2.4.q.1 Name
    - 3.2.4.q.2 Representation
    - 3.2.4.q.3 Units/Format
    - 3.2.4.q.4 Precision/Accuracy
    - 3.2.4.q.5 Range
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements