**DCIT208 Software Engineering**

**Software Requirements Specification (SRS) Template**

Items intended to stay as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

This document is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Keep the header where you decide to omit a section, but insert a comment saying why you omit the data.

(Project Title)

(Team Name and Number)

(Team Members)

**Software Requirements Specification**

**Document**

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

**Table of Contents**

[1. Introduction 5](#_Toc137810604)

[1.1 Purpose 5](#_Toc137810605)

[1.2 Scope 5](#_Toc137810606)

[1.3 Definitions, Acronyms, and Abbreviations. 5](#_Toc137810607)

[1.4 References 5](#_Toc137810608)

[1.5 Overview 5](#_Toc137810609)

[2. The Overall Description 6](#_Toc137810610)

[2.1 Product Perspective 6](#_Toc137810611)

[2.1.1 System Interfaces 6](#_Toc137810612)

[2.1.2 Interfaces 7](#_Toc137810613)

[2.1.3 Hardware Interfaces 7](#_Toc137810614)

[2.1.4 Software Interfaces 7](#_Toc137810615)

[2.1.5 Communications Interfaces 8](#_Toc137810616)

[2.1.6 Memory Constraints 8](#_Toc137810617)

[2.1.7 Operations 8](#_Toc137810618)

[2.1.8 Site Adaptation Requirements 8](#_Toc137810619)

[2.2 Product Functions 9](#_Toc137810620)

[2.3 User Characteristics 9](#_Toc137810621)

[2.4 Constraints 10](#_Toc137810622)

[2.5 Assumptions and Dependencies 10](#_Toc137810623)

[2.6 Apportioning of Requirements. 11](#_Toc137810624)

[3. Specific Requirements 11](#_Toc137810625)

[3.1 External Interfaces 13](#_Toc137810626)

[3.2 Functions 13](#_Toc137810627)

[3.3 Performance Requirements 14](#_Toc137810628)

[3.4 Design Constraints 14](#_Toc137810629)

[3.6 Software System Attributes 15](#_Toc137810630)

[3.6.1 Reliability 15](#_Toc137810631)

[3.6.2 Availability 15](#_Toc137810632)

[3.6.3 Security 16](#_Toc137810633)

[3.6.4 Maintainability 16](#_Toc137810634)

[3.7 Organizing the Specific Requirements 17](#_Toc137810635)

[3.7.1 System Mode 17](#_Toc137810636)

[3.7.2 User Class 17](#_Toc137810637)

[3.7.3 Objects 17](#_Toc137810638)

[3.7.4 Feature 18](#_Toc137810639)

[3.7.5 Stimulus 18](#_Toc137810640)

[3. 7.6 Response 18](#_Toc137810641)

[3.7.7 Functional Hierarchy 18](#_Toc137810642)

[3.8 Additional Comments 18](#_Toc137810643)

[4. Change Management Process 19](#_Toc137810644)

[5. Document Approvals 19](#_Toc137810645)

[6. Supporting Information 19](#_Toc137810646)

# 1. Introduction

*The following subsections of the Software Requirements Specifications (SRS) document should provide an overview of the entire SRS. The thing to keep in mind as you write this document is that you are telling what the system must do – so that designers can ultimately build it. Do not use this document for design!!!*

## 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 software product(s) to be produced by name*
2. *Explain what the software product(s) will, and, if necessary, will not do*
3. *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
4. *Be consistent with similar statements in higher-level specifications if they exist*

*This should be an executive-level summary. Do not enumerate the whole requirements list here.*

## 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*

1. *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 your application uses specific protocols or RFC’s, 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 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. The Overall Description

*Describe the general factors that affect the product 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 plain English for the consumption of the customer. Section3 will contain a specification written for the developers.*

## 2.1 Product Perspective

*Put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software. If you are building a real system,compare its similarity and differences to other systems in the marketplace. If you are doing a research-oriented project, what related research compares to the system you are planning to build.*

*A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. This is not a design or architecture picture. It is more to provide context, especially if your system will interact with external actors. The system you are building should be shown as a black box. Let the design document present 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 you have to interact with. For instance, if you are building a business application that interfaces with the existing employee payroll system, what is the API to that system that designer’s will need to use?*

### 2.1.2 Interfaces

*Specify:*

1. *The logical characteristics of each interface between the software product and its users.*
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? If you are designing for the general student population for instance, what is the impact of ADA (American with Disabilities Act) on your interface?*

### 2.1.3 Hardware Interfaces

*Specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics. 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 Mac 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 X10 type home 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” If you just delete sections that are not applicable, then readers do not know if: a. this does not apply or b. you forgot to include the section in the first place.*

### 2.1.4 Software Interfaces

*Specify the use of other required software products and interfaces with other application systems. For each required software product, include:*

1. *Name*
2. *Mnemonic*
3. *Specification number*
4. *Version number*
5. *Source*

*For each interface, provide:*

1. *Discussion of the purpose of the interfacing software as related to this software product*
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 customer 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 definintions to be provided to the company DBA for setup.*

*A key point to remember is that you do NOT want to specify software here that you think would be good to use. This is only for* ***customer-specified systems*** *that you* ***have*** *to interact with. Choosing SQL Server 7 as a DB without a customer requirement is a Design choice, not a requirement. This is a subtle but important point to writing good requirements and not over-constraining the design.*

### 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 happen to use web services transparently to your application then do not list it here. 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 or RFC.*

### 2.1.6 Memory Constraints

*Specify any applicable characteristics and limits on primary and secondary memory*. *Don’t just make up something here. If all the customer’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 256-512M 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*
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 stuff earlier, then cover business process type stuff that would impact the design. For instance, if the company 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 your system requires any modifications to the customer’s work area, document that here. For instance, “A 100Kw backup generator and 10000 BTU air conditioning system must be installed at the user site before software installation”.*

*This could also be software-specific, like, “New data tables created for this system must be installed on the company’s existing DB server and populated prior to system activation.” Any equipment the customer would need to buy or any software setup needed so that your system will install and operate correctly should be documented here.*

## 2.2 Product Functions

Leadership Development Resources, the program will provide a rage of resources such as articles, e-books and case studies to help individuals enhance their leadership skills, knowledge and capabilities.

Personal Development tools, the program will offer a range of tools and resources to support individuals with the passion of becoming a leader in their personal development journey. These tools may include goal setting templates and time management techniques, empowering individuals to enhance their overall well-being.

Donation platform, the program will have a donation mechanism that will allow individuals who would like to support the course of the organization.

Platform for joining the organization, the program will have a form where individuals can fill out if they want to join the organization.

## 2.3 User Characteristics

The intended users of the program will mostly be individuals who are high schools, university and those who recently graduated from the university. Based on this, the language that will be used on the program will be clear and concise. Information on the program will be accessible and understandable to all individuals from any educational background. The program will be structured such that individuals who are now entering into leadership will have proper guidelines. Due to the diverse languages in the world, the system may have a multilingual feature which will allow the user who use the program to select the language they want.

## 2.4 Constraints

The development of the program must comply with relevant policies and guidelines like data protection and privacy regulation and copyright rules that govern the handling of sensitive information.

The programs design must consider any hardware limitations such as signal timing requirements, compatibility with older and less powerful devices that may impact its functionality.

The integration of the program with other applications may cause constraints because of compatibility, data exchange format required for seamless integration.

If the program needs to support simultaneous user interactions or handle high volumes of traffic, its design should be able to handle parallel operation which would mean taking into consideration load balancing, scalability, and performance optimization techniques.

The choice of programming languages and frameworks may be constrained by the team’s expertise and with compatibility with already existing systems.

## 2.5 Assumptions and Dependencies

Availability of Required Hardware and Software, there is the need for hardware and software components like servers, hosting services, operating systems, databases and web frameworks to be available and accessible to support the implementation and deployment of the program, therefore any changes in the availability of these resources will impact the requirements and design of the program.

Internet Connectivity, it is assumed that before users can access the program, there should be a reliable internet connection. The requirement and design of the program may need to be adjusted if there are constraints on internet connectivity.

Compatibility with Web Browsers, the SRS assumes that the program will be compatible with highly used web browsers like Chrome and Firefox, so if there is any changes in the browser technologies or versions, updates will have to be made to the requirements and design of the program to ensure user flexibility and system functionality.

Budget Constraints, it is assumed that there is a decided budget for the development of the program. Therefore, if there is a change in budget allocation, it may influence the prioritization of the requirement, the scope of the project and the overall design decisions.

## 2.6 Apportioning of Requirements.

Looking at the time range certain functionality will not be available at the initial version of the program. Functionalities like the multilingual feature may not be the in the first version of the program.

# 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*

1. *Specific requirements should be cross-referenced to earlier documents that relate*
2. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
3. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*

*Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*

*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 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” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*

*Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer 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 customer/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 Functions

|  |  |
| --- | --- |
| ID | REQUIREMENT DESCRIPTION |
| FR.1 | New users and volunteers registration |
| FR.2 | Content management |
| FR.3 | Donations |
| FR.4 | Event registration |
| FR.5 | User Feedback |

FR.1

* Forms will be available for users who would like to join the organization as volunteers or new members.
* Validation checks will be performed.

FR.2

* Users can browse and read published content, which is categorized and tagged for easy navigation.

FR.3

* Users will be able to make donations to the organization through a simple and user friendly platform on the website

FR.4

* Users can RSVP for upcoming events hosted on the website.

FR.5

* There will be section to take user feedback.

## 3.3 Performance Requirements

Page Load Time

The leadership website shall have a maximum page load time of 3 seconds for all pages, measured from the initiation of the request to the complete rendering of the page content in the user's browser.

Image and Media Loading

Images and media files displayed on the website shall load within 2 seconds on a standard broadband internet connection (with a minimum bandwidth of 5 Mbps). The system should employ appropriate techniques such as image compression and content delivery networks (CDNs) to optimize the loading time of media assets.

## 3.4 Design Constraints

*Cross-Platform Compatibility:*

*The leadership website shall be designed to be compatible with major web browsers, including but not limited to Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. The website should be responsive and optimized for various screen sizes, including desktop, tablets, and mobile devices, to ensure a consistent user experience across different platforms.*

*Performance and Scalability:*

*The website shall be designed to deliver optimal performance, with fast loading times and minimal downtime. Performance optimization techniques such as caching, and code optimization should be considered during the development process.*

*5. Integration with Third-Party Services:*

*The website may require integration with third-party services such as payment gateways, or analytics tools. The design should allow for seamless integration and provide appropriate APIs or interfaces to connect with these services, ensuring smooth functionality and data exchange.*

*Branding and Visual Consistency:*

*The design of the leadership website should align with the organization's branding guidelines and maintain visual consistency. This includes the use of appropriate color schemes, fonts, logos, and overall design elements to reflect the organization's identity and enhance brand recognition.*

*Maintenance and Upgradability:*

*The website design should facilitate easy maintenance, updates, and future enhancements. It should use modular and well-documented code, allowing developers to make changes without causing disruptions to the overall system.*

## 3.6 Software System Attributes

* The system will be able to handle a huge number of users within an acceptable responsive time. The system will be able to recover from failures and restore normal operation without data loss.
* The system will have a user-friendly interface that will allow users to easily understand and interact with its features and the system will be compatible with most hardware platform and software dependencies.
* The system will comply with relevant legal and regulatory standards, it will meet specific requirements related to data privacy and ethical standards.

## 3.6.1 Reliability

* Analysis of potential failures and their impact on the systems functionality will be conducted. This will help in identifying the critical components and their potential consequences.
* Error handling mechanisms will be implemented so that the system will be able to detect errors, recover from them and continue operation without data loss.
* Implementing monitoring mechanisms to access the system’s performance and reliability. Regular maintenance and updates will be done to address any identified issues and ensure the system remain reliable over time.

## 3.6.2 Availability

* Redundancy and fault tolerance, Redundancy will be included in the systems architecture to alleviate the impact of failures. This can involve hardware components like servers, and this will ensure that even if one component fails, the system will continue to operate without interaction.
* Fast recovery and restart, the system will be designed to enable fast recovery and restart after a failure.
* Continuous monitoring and maintenance of the systems health performance to identify any issues affecting availability. This will help address the system’s vulnerability and ensure resilience and availability of the system.

## 3.6.3 Security

* Implementation of access control mechanisms to restrict access to the software’s system based on user roles. This maybe involve user authentication to ensure that only authorized persons can interact with the system and its data.
* Keeping comprehensive logs that capture relevant events within the system. This may include logging user action, system events and security events. This will serve as a way of tracking and investigating suspicious activities and provide evidence in case of security incidents.
* Following secure coding practices during software development will help reduce the introduction of security vulnerabilities. This includes the following of coding standards, code reviews and the use of secure coding frameworks.

## 3.6.4 Maintainability

* The system will be designed with a modular structure with well-defined and clear responsibilities of the components.
* A clear and comprehensive documentation will be provided explaining the systems architecture, design, and functionality.
* Codes will be easy to read and understand and follow. Also, software will be as simple as possible to minimize complexity.
* Version control systems will be utilized to keep track of the changes that will be made to the system.
* The system will be designed with testing in mind, this is to enable easier detection and resolution of issues during maintenance and helps ensure modifications do not introduce new defects.

## 3.7 Organizing the Specific Requirements

*For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subclasses.*

### 3.7.1 System Mode

*Some systems behave quite differently depending on the mode of operation. When organizing by mode there are two possible outlines. The choice depends on whether interfaces and performance are dependent on mode.*

### 3.7.2 User Class

*Some systems provide different sets of functions to different classes of users.*

### 3.7.3 Objects

*Objects are real-world entities that have a counterpart within the system. Associated with each object is a set of attributes and functions. These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.*

### 3.7.4 Feature

*A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. Each feature is generally described in as sequence eof stimulus-response pairs.*

### 3.7.5 Stimulus

*Some systems can be best organized by describing their functions in terms of stimuli.*

### 3. 7.6 Response

*Some systems can be best organized by describing their functions in support of the generation of a response.*

### 3.7.7 Functional Hierarchy

*When none of he above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be use dot show the relationships between and among the functions and data.*

## 3.8 Additional Comments

*Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.*

*Three are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; 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, Outputs.*

# 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 customer 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?*

# Document Approvals

*Identify the approvers of the SRS document. Approver name, signature, and date should be used.*

# 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*

*(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*

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

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements. You should pick the best one of these to organize section 3 requirements.

**Outline for SRS Section 3**

**Organized by mode: Version 1**

3. Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. Mode 2

.....

3.2.*m* Mode *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by mode: Version 2**

3. Specific Requirements

3.1 Functional Requirements

1. Mode 1

3.1.1.1 External interfaces

3.1.1.1 User interfaces

3.1.1.2 Hardware interfaces

3.1.1.3 Software interfaces

3.1.1.4 Communications interfaces

3.1.1.2 Functional Requirement

3.1.1.2.1 Functional requirement 1

.....

3.1.1.2.*n* Functional requirement *n*

3.1.1.3 Performance

3.1.2 Mode 2

.....

3.1.*m* Mode *m*

1. Design constraints
2. Software system attributes
3. Other requirements

**Outline for SRS Section 3**

**Organized by user class (i.e. different types of users ->System Adminstrators, Managers, Clerks, etc.)**

3. Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. User class 2

.....

3.2.*m* User class *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by object (Good if you did an object-oriented analysis as part of your requirements)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Classes/Objects

3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)

1. Attribute 1

.....

3.2.1.1.*n* Attribute *n*

1. Functions (services, methods, direct or inherited)

3.2.1.2.1 Functional requirement 1.1

.....

3.2.1.2.*m* Functional requirement 1.*m*

3.2.1.3 Messages (communications received or sent)

3.2.2 Class/Object 2

.....

3.2.*p* Class/Object *p*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by feature (Good when there are clearly delimited feature sets.**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. System features

3.2.1 System Feature 1

3.2.1.1 Introduction/Purpose of feature

3.2.1.2 Stimulus/Response sequence

3.2.1.3 Associated functional requirements

3.2.1.3.1 Functional requirement 1

.....

3.2.1.3.*n* Functional requirement *n*

3.2.2 System Feature 2

.....

3.2.*m* System Feature *m*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by stimulus (Good for event driven systems where the events form logical groupings)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Stimulus 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Stimulus 2

.....

3.2.*m* Stimulus *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by response (Good for event driven systems where the responses form logical groupings)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Response 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Response 2

.....

3.2.*m* Response *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by functional hierarchy (Good if you have done structured analysis as part of your design.)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 1

1. Data entities
2. Pertinent processes
3. Topology

3.2.1.2 Data flow diagram 2

1. Data entities
2. Pertinent processes
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

1. Process 1
2. Input data entities
3. Algorithm or formula of process
4. 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

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

**Outline for SRS Section 3**

**Showing multiple organizations (Can’t decide? Then glob it all together)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Feature 1.1

3.2.1.1.1 Introduction/Purpose of feature

3.2.1.1.2 Stimulus/Response sequence

3.2.1.1.3 Associated functional requirements

3.2.1.2 Feature 1.2

3.2.1.2.1 Introduction/Purpose of feature

3.2.1.2.2 Stimulus/Response sequence

3.2.1.2.3 Associated functional requirements

…..

3.2.1.*m* Feature 1.*m*

3.2.1.*m*.1 Introduction/Purpose of feature

3.2.1.*m*.2 Stimulus/Response sequence

3.2.1.*m*.3 Associated functional requirements

3.2.2 User class 2

.....

3.2.*n* User class *n*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by Use Case (Good when following UML development)**

3. Specific Requirements

3.1 External Actor Descriptions

3.1.1 Human Actors

3.1.2 Hardware Actors

3.1.3 Software System Actors

3.2 Use Case Descriptions

3.2.1 Use Case 1

3.2.2 Use Case 2

3.2.n Use Case n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements