Software Requirements Specification (SRS) Template

Items that are intended to stay in 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.

The document in this file 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. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.

<u>Stay clear and short, yet complete.</u>

Encryptor
The Only Member Team
Omer Edut

Software Requirements Specification

Document

Version: 1.0 **Date: 01/21/2020**

Table of Contents

1. Introduction

1.1 Purpose

This SRS document defines the requirements for the product, and is intended for customers (users) to approve the business requirements, as well as the development personnel to define the required functions and technological guidelines. Also, to the examiners as the basis for designing the test scripts.

1.2 Scope

This software product is called Encryptor.
Ecryptor is a mobile app designed to encrypt or decode an image with sensitive information / content within another image so as not to arouse suspicion or interest among other stakeholders.

1.3 Definitions, Acronyms, and Abbreviations.

This section is not relevant in this SRS document.

1.4 References

This section is not relevant in this SRS document.

1.5 Overview

This SRS document provides a general explanation of the Encryptor product, its purpose, and who it is intended for. This explanation can be found in section 1 of this document.
In addition, this SRS document includes the customer's requirements for this product which the customer can find in section 2 of this document clearly and specified.
In addition, this SRS document contains the relevant product requirements intended for the development and testing department of this product, these requirements can be found in Section 3 of this document.

2. The Overall Description

Encryptor is a mobile application that allows you to create an encrypted image within another image, or decrypt an encrypted image within another image.
When encrypting an image, the app will require two images (cover image and encryption image), in addition to requiring encryption parameters.
After entering the data the app will encrypt the image and display the result.
When decoding an image, the application will require two images (a cover image and the encrypted image) and return the decoding result.

The app will allow the result to be exported to an image file, or transferred to another messaging app.

2.1 Product Perspective

Encryptor is a totally self-contained system.
But it does allow input from other image applications, as well as exporting the result to other messaging apps.

2.1.1 System Interfaces

Encryption allows you to receive images in an external image app for a cover image or encryption image.
Encryptor can export the encryption result or the decryption results to the messaging app.

2.1.2 Interfaces

When starting the app the user can choose to switch from encryption mode to decoding mode or vice versa.
In encryption mode, there is a user interface that expects the user to receive a cover image and encryption image, and other encryption parameters.
After entering this data the user will run the encryption algorithm and wait for the encryption result.
Similarly, in the image decoding mode the user is required to enter a standard image and an encrypted image. The user will then run the decoding algorithm and wait for the result.
The result can be saved to the device or exported to another app.
The user can also completely cancel the operation and select new parameters.

2.1.3 Hardware Interfaces

As part of the app options, it will be possible to connect to the camera of the device to upload an image directly from the camera.
In addition, the encryption / decryption result can be saved on the device by accessing the device memory.

2.1.4 Software Interfaces

As part of the app options, images can be uploaded directly from the device's photo gallery.

When selecting an image, a user interface will be opened that allows you to choose how to upload the image to the app, if the selected image succeeds, the selected image will be displayed to the user, unless a suitable message is displayed to the user and another option is chosen to select an image.

2.1.5 Communications Interfaces

Encryptor does not use communication interfaces.

2.1.6 Memory Constraints

Mobile devices are limited in their amount of RAM, however, nowadays almost all devices have 16GB RAM or more.
Therefore the use of the system should not exceed 16GB RAM at any given time.
For further reading: https://www.androidpit.com/memory-needed-on-smartphone.
For external memory, the system does not require external memory, so no external memory is required.

2.1.7 Operations

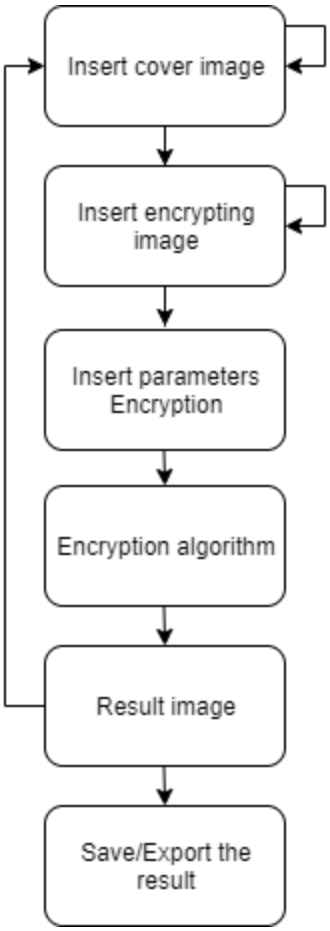
There are no relevant actions for this section.

2.1.8 Site Adaptation Requirements

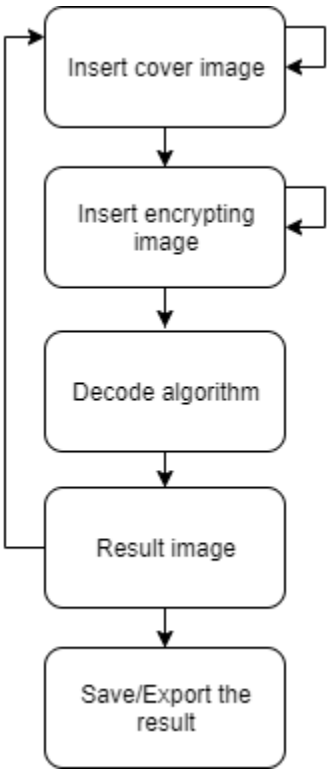
There are no relevant site adaptation requirements for this section.

2.2 Product Functions

Encryption process



Decode process



2.3 User Characteristics

Encryptor is simple software to operate and requires no prior knowledge or additional skill besides the basic use of a mobile device.

2.4 Constraints

Encryptor has no general (non-functional) requirements that limit key options.

2.5 Assumptions and Dependencies

Encryptor has no relevant requirements for this section.

2.6 Apportioning of Requirements.

Requirement	priority
Insert cover image	1
Cancel cover image	1
Insert encryption image	1
Cancel encryption image	1
Insert parameters ecryption	1
Encryption algorithm	1
Insert decode image	2
Cancel decode image	2
Decode algorithm	2
Show result image	1
Export/Save result image	1
Send result to some messaging app	3

3. Specific Requirements

[Here put most of effort]
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 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

[SKIP THIS PART]

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

3.2.1 Encryption mode

3.2.1.1 The software will require entering a cover image.
3.2.1.2 The software will enable the cover image to be cancel.
3.2.1.3 The software will require an image encryption.
3.2.1.4 The software will enable the encryption image to be cancel.
3.2.1.5 The software will require setting grayscale for encryption.
3.2.1.6 The software will run the encryption operation.
3.2.1.7 The software will display the result.
3.2.1.8 The software will enable save the result on the device.
3.2.1.9 The software will allow the result to be exported to another app.

3.2.2 Decode mode

3.2.2.1 The software will require entering a cover image.
3.2.2.2 The software will enable the cover image to be cancel.
3.2.2.3 The software will require an image encryption.
3.2.2.4 The software will enable the encryption image to be cancel.
3.2.2.5 The software will run the decoding operation.
3.2.2.6 The software will display the result.
3.2.2.7 The software will enable save the result on the device.
3.2.2.8 The software will allow the result to be exported to another app.

3.3 Performance Requirements

[SKIP THIS PART]

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 terminals to be supported
(b) The number of simultaneous 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 1 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

Encryptor has no database requirements.

3.5 Design Constraints

Encryptor has no design constraints requirements.

3.5.1 Standards Compliance

[SKIP THIS PART]

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

[SKIP THIS PART – FILL ONLY SECTION 3.6.3 ON Security]

There are a number of attributes of software that can serve as requirements. 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 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. Its easy to start philosophizing here, but keep it specific.

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 requirements, express them here. This doesn't refer to just having a program that does not crash. This has a specific engineering meaning.

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

This section is not relevant to the app.

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 should not be placed here just because they are thought to be good design practices. If 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 host machines 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*

Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating tradeoff analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.

ID	Characteristic	H/M/L	1	2	3	4	5	6	7	8	9	10	11	12
1	Correctness													
2	Efficiency													
3	Flexibility													
4	Integrity/Security													
5	Interoperability													
6	Maintainability													
7	Portability													
8	Reliability													
9	Reusability													
10	Testability													
11	Usability													
12	Availability													

Definitions of the quality characteristics not defined in the paragraphs above 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 really a section, it is talking about how to organize requirements you write in section 3.2. At the end of this template there are a bunch of alternative organizations for section 3.2. Choose the ONE best for the system you are writing the requirements for.

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

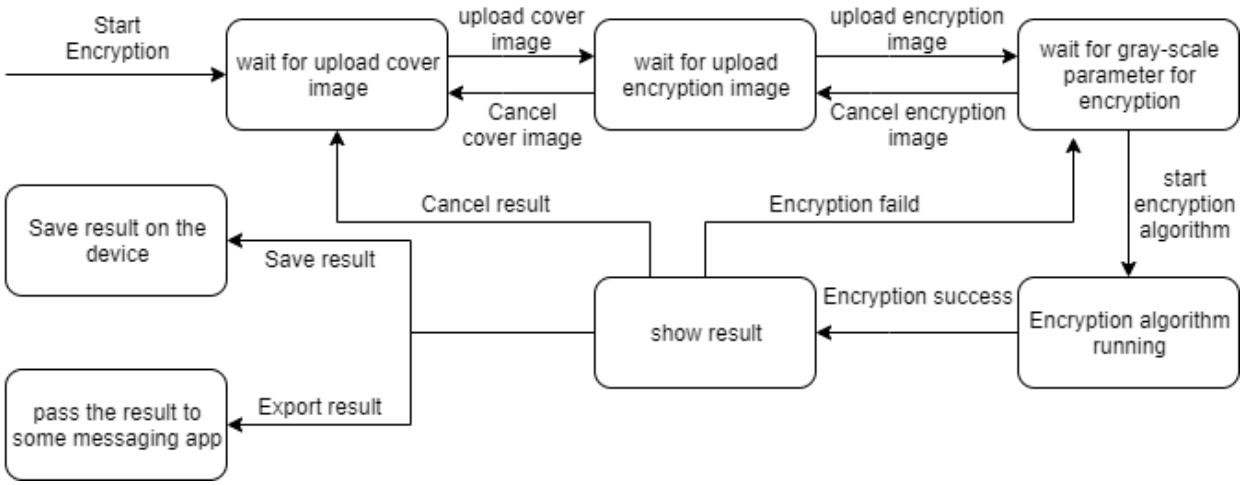
Some systems can be best organized by describing their functions in support of the generation of a response.

3.7.7 Functional Hierarchy

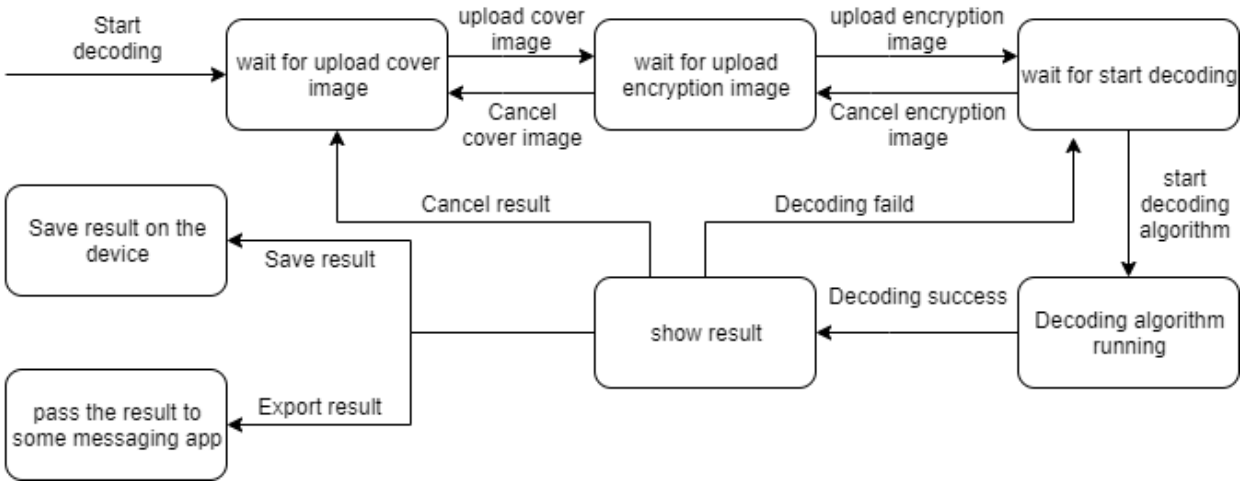
When none of he above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be use dot show the relationships between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

3.8 Additional Comments

Encryption state:



Decoding state:



4. Change Management Process

[SKIP THIS PART]

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?

5. Document Approvals
[YOUR SUPERVISOR]

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

.
---------- [END OF RELEVANT PARTS]----------

6. Supporting Information
[SKIP THIS PART]

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
      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 Mode 1
         3.2.1.1 Functional requirement 1.1
         .....
         3.2.1.*n* Functional requirement 1.*n*
      3.2.2 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
    3.1.1 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*
  3.2 Design constraints
  3.3 Software system attributes
  3.4 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
      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 User class 1
         3.2.1.1 Functional requirement 1.1
         .....
         3.2.1.*n* Functional requirement 1.*n*
      3.2.2 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
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Classes/Objects
  - 3.2.1 Class/Object 1
    - 3.2.1.1 Attributes (direct or inherited)
      - 3.2.1.1.1 Attribute 1
      - .....
      - 3.2.1.1.*n* Attribute *n*

    - 3.2.1.2 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
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 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
  - 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 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
  - 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 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
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

Outline for SRS Section 3
Showing multiple organizations (Can't decide? Then glob it all together)

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 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