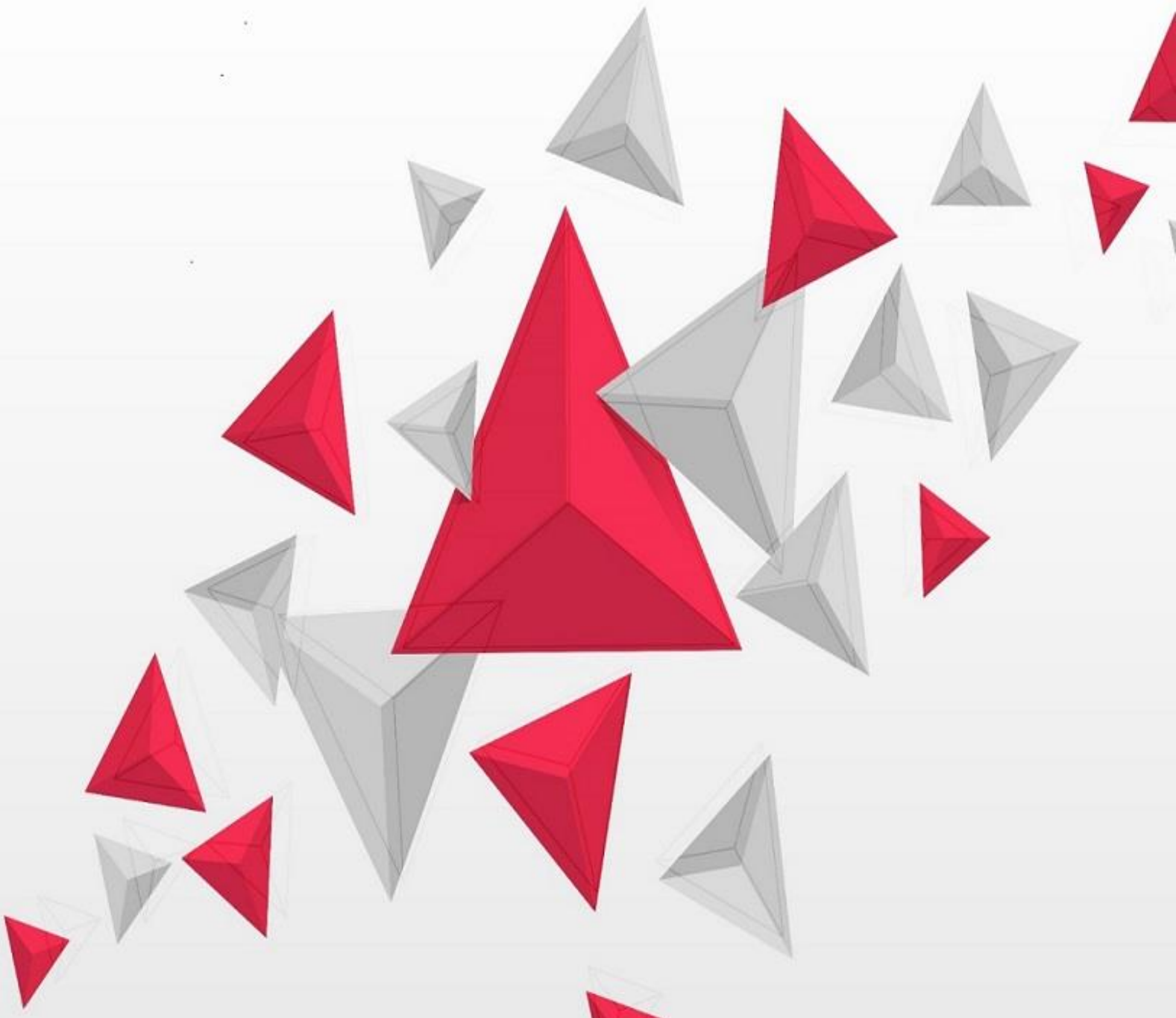


Software Requirements Specification (SRS) Document

Guide Book





SRS DOCUMENT – INTRODUCTION

In the previous lesson, we learned a lot about the distinctions between requirements and specifications documents.

Software Requirement Specifications or SRS is a specifications document that elaborates 'What' requirements must be fulfilled to satisfy the business's needs.

A Software Requirement Specifications (SRS) document is a detailed and structured requirements document that contains the functional requirements (depicts behavior), non-functional requirements (illustrates characteristics), and lists all the use cases that the software must fulfill.

SRS is one of the vital documents in any project. It bridges the gap between what business wants and what they will get by documenting the software's overall layout, characteristics, and workflows.

Since a software's major requirements are clearly organized and elaborated in an SRS, it's commonly used for estimating the cost and effort required for developing that software. At times, it's even used as a contractually binding document between two parties since it contains enough details to reach an agreement.

You can view an SRS as a document that 'elaborates' the BRD's high-level statements into a module, sub-module, and features without elaborating them in-depth.

The systems analyst of the project prepares the SRS; however, in case a systems analyst is not available on the project, it has to be authored by the business analyst.

To prepare an SRS, the analyst has to conduct requirement elaboration sessions with the relevant stakeholders, meticulously analyze all the aspects of the software being developed and then list down the requirements against each one of them. It's ensured that every listed requirement in an SRS should fulfill the business objectives listed in the BRD



ASPECTS OF AN SRS DOCUMENT

- *Not all organizations create the SRS document*

Don't be surprised if you don't find the SRS document within some organizations. Some organizations or small projects do not create an SRS and instead elaborate their BRD to accommodate functional and non-functional requirements and use case details.

- *SRS document is sometimes referred to with some other names*

SRS may sometimes be called Product Requirements Document (PRD) and System Requirements Specification document; however, the documents' essence and contents remain the same.

- *SRS document is prepared before the FRS document, but after the BRD*

Since the SRS document elaborates the information contained within the Business Requirement Document (BRD), it's prepared after the BRD. However, the Functional Requirement Specification (FRS) document is prepared after and in line with the SRS document.

- *The SRS document is prepared in the planning phase of the project*

Unlike the BRD document that defines the high-level requirements and the business case, the SRS document is composed during the 'planning' stage of the project with other requirement documents like the FRS, Use case documents, and Functionality Matrix.

- *It's a functional document and not a technical one*

Like the BRD and FRS document, no technical or infrastructure-related details are ever included in an SRS.



AUDIENCE OF AN SRS DOCUMENT

Let's take a quick look at the primary audience of the SRS document

- Project managers oversee the preparation of the SRS document and ensure the alignment of its content with the BRD
- SMEs (subject matter experts) evaluates the correctness of the information presented in the SRS
- The project's technical architects and Implementation Lead are responsible for designing and developing the listed functionalities.
- Business stakeholders carefully review the SRS in order to ensure it lists all the system features and functionalities
- Development/Technical team
- Testing Team



HOW TO CREATE AN SRS DOCUMENT

If the business analyst has carefully and meticulously created the BRD document for the project, then, believe me, half of the battle is already won!

The SRS document being a functional extension of the BRD document borrows quite a lot of sections from it, and we will now concentrate on the unique areas of the SRS document.

User/Actor Roles and Characteristics

Every application or software product being created will have some users interacting with the system. Based on their level of interaction with the system, these actors will have different roles, and the same should be specified under this section, in the below format:

- Role Name: The name of the role as per the application nomenclature
- Role Designation/Title: The designation or title of the users to whom this role can be assigned
- Role Description: A brief description of the role itself, its underlined rights & privileges on the application, and the characteristics of the user to whom this role is assigned
- Frequency of use: Specify how often the user with this role uses the system

System Features

Functionality is defined as a desired output or result expected from the system after a (series of) inputs. For any system with a moderate level of complexity, such functionalities are grouped to form a 'feature,' and similar features constitute a 'module'.

Each of the requirements should be backward traceable by explicitly referencing its source (business requirement ID) in earlier documents and should be organized in the below format:

Business Req. ID	Module Name	Feature name

Business Process Flow

Software applications follow a hierarchical/sequential flow of events that are initiated after a specific sequence of inputs is carried out.

Any details and/or diagrams related to the solution process flow, data flow, and information flow should come here. Also, if there are multiple processes within the system, the details of where those processes fit, how they are interconnected (if they are), and to what effect they are used should be included.

Wireframes/Prototype

A prototype or mockup is an initial version of a product and visually depicts the end product. A prototype or wireframe of the software being created should be included under this section to show any data or process-based navigation, represent different scenarios, and project the general look and feel of the application being created.

The aim of the prototypes should be to validate the understanding of the requirements and get initial feedback against the user interface and design elements.

User Interface Requirements

This section contains the requirements about the look and feel of the interface using which the user will interact with the software. Following details shall be included under this section:

- The general layout of the application
- The way content, as well as data, is presented to a user
- Different kinds of navigation available in the software
- Representation of dynamic design elements (widgets & menus)
- Does the user interface differ based on the user's role/user group?

Use Case Listing

Use cases are 'requirement specification document' that accurately describes how a user (actor) will interact with the system being developed through a flow of events.

This section should contain a listing of all the use cases that are supposed to be prepared during the analysis phase of the project. Each use case should be supplemented with a brief description of the functionalities, scenarios, and flows contained within each one of them.



SRS DOCUMENT - BEST PRACTICES

In this lesson, I have summarized some of the efficient tips for SRS documentation, which I have noticed are very easy to implement but have a massive impact on the documents you prepare as a BA.

1. **Don't Ramble! Make your sentences short and crisp**

Traditional Research says that a human being can hold 7 objects in his working memory at a time, and the latest studies point that this number has come down to 4. So, if you are using excessively long sentences, you are increasing the chances of your stakeholders missing or misunderstanding your SRS document. So, make your sentences short and chunk the information into multiple sentences to better understand the business users. Also, developers take one requirement at a time, and they will be more focused and clear with crisp and straightforward sentences.

2. **Avoid ambiguity around your SRS document**

It would be best to strive to be as specific as possible in your requirements. Avoid using words like approx., etc., some, sometimes, ordinarily, most, mostly, usually, and might.

Also, make sure you avoid duplications wherever possible and should also steer away from contradictory and debatable statements.

Crisp documents portray clear thinking and a methodical outlook, and both these skills are so very integral to the role of a BA.

Let's see a small example around **Specific and Precise** requirements:

The user can change their password by clicking on the change password link on the login page and writing their registered email ID on the following screen.

Then, the system will send an email on their registered email id, which, when clicked, will bring the user to a new screen. On this new screen, the user can now change his password.

The steps to be followed while changing the password are:

1. Click on the 'Change Password' link on the 'Login' page
2. The flow goes to a 'Confirmation' screen where the user's registered email ID should be entered
3. The system sends a 'Reset Password' link on the registered email ID (entered in the previous step)
4. Clicking on this link will bring the user to the 'Change Password' screen
5. The new password could be set on this screen

Now, let's see an example of a **consistent** requirement:

Inconsistent

Consistent

The application shall	The User management tool shall
The User management tool shall	The User management tool shall
The system shall	The User management tool shall
The user management system shall	The User management tool shall

Now, let's see how the requirements are **complete**:

Incomplete	Complete
While resetting a password, it should not be the same as the previous passwords. Also, the password guidelines should be followed	While resetting a password, it should not be the same as the last 5 passwords. Also, the password should be a minimum of 8 characters long with at least one numeric and one special character

Let's see an **unattainable** requirement:

With one application server, the User management tool shall serve 2 million concurrent users, and the response time should not be more than 1 second.

An example of how the requirements should be **testable**:

The web application should have a multilingual support	The web application should support English, French and German languages
The user interface should work in all the major browsers	The user interface should work in IE, chrome, and safari browser
In case of wrong credentials system should give a 'proper' alert message	In case of incorrect credentials, the system should show an alert message reading 'Invalid Credentials. Please check the email ID/password entered.

3. Know how to use: Shall, Will, and Should

Due to our lack of knowledge of words, we often use them interchangeably, confusing for the end-user. The most common is the use of Shall, Will, and Should.

- "Shall "should be used where requirements are being stated, - The system shall have a response time of < 200ms
- "Will" should be used to represent statements of facts – The system will have a separate login for administrative users
- "Should" represents a goal that needs to be achieved. – The system should concurrently cater to 10,000 users at any given point in time

4. WHAT and not HOW

First thing first – As a business analyst, you have to describe 'What' the system will do. *That's it.* Now, although you might think that this tip is self-explanatory, you forget that we often inadvertently document 'How' the system does it.

E.g., You, in your use case, might write something like, When the user clicks on the 'save' button on the 'Create User' page, the system will create the user in the application portal by saving all the attributes defined in the page in the MySQL database.

Do you think it's correct? No, it isn't.

In the first half, you describe what the system will do, and in the next half, you describe how. So, catch yourself when you're mentioning programming language and software objects in your 'requirement' documents.

5. Define one and only one requirement at a time

Your requirements should be atomic, meaning it should comprise only one component. Don't be tempted to combine multiple requirements by using conjunctions like *and*, *or*, *also*, *with* and the like. This is particularly important because words like these can lead to developers and testers missing out on some of the requirements. One way to achieve this is by breaking the requirement down until it can be considered a discrete test case.

One other great piece of advice is to express any equation or formula being used in the requirements in words - This will be a massive aid to someone reading your requirements. Go ahead and give at least 2 examples to corroborate your equation or formula.

6. Document reviews are as critical as the document itself

There is a lot of difference between a document and an approved document. An approved requirement document is a baseline document and an artifact that is verified and validated. But an unverified use case is just a piece of a client's wish list.

So, always have a dual review of your documents, first by an internal peer/superior and then by the client. The document should also mention the review date, reviewer name, and reviewer's comments.

7. Check before you channel

There are a few things that you should check before sending your SRS document for review. Let's quickly see what they are:

- Formatting of your SRS document – Never send an ill-formatted copy as they reflect your work ethics and professionalism
- Spelling and grammar check – Spelling and grammatical mistakes reflect a careless attitude. Something that nobody wants to be associated to
- Accuracy of the attached diagrams – attachments should be adequately defined. Also, make sure you are attaching the suitable document as I see people attaching the wrong documents in a hurry
- Names and terminology used – Be careful about the names of persons and vocabulary of terms in your documents because if such a mistake is made, it causes a lot of confusion
- The authenticity of Links and references to other documents – if you are referring to any external source or link in the SRS document, make sure it's from a credible source. Otherwise, you will be considered a gullible

analyst.

8. 'Out' of scope but 'In' the document

In the initial stages of my career, I was working with an aviation client on a 'Workflow Management System'.

At the time of our UAT, the client asked for a feature that was discussed briefly and was thought of by the team as an item for the next phase of the project.

However, the client was adamant and demanded us to show where it is documented that the feature is out of scope for that phase!

We never had that documented, which caused us 10 days of development effort to have that feature ready for them. That day I learned, 'If a feature was discussed even in passing, your client might assume it is "in-scope" until and unless you clearly document it as "out of scope".'

9. A final tip - Be flexible in developing the SRS document iteratively

During the planning stages of a project, a lightweight SRS document is often developed, which is then extended during the elaboration or clarifications sessions. You should understand that your SRS document will work as a central mechanism for reducing risks and requirement ambiguity. To achieve so, you will have to create different versions of the SRS document reflecting the change or updating the requirements. I am speaking from experience that not many BAs maintain versions of their SRS documents, and if you are doing so on your project, you are sure to gain attention and recognition for that.

And when you are updating versions, make sure to update the associated diagrams and matrices as well.