SRS FRS &BRS FILES

- **SRS Document**
- SRS, or software requirement specification, is a document prepared by a team of system analysts that is used to describe software that will be developed, the main business purpose and functionality of a certain product, and how it performs its core functions. An SRS is a basis for any project as it consists of a framework that will be followed by each team member. An SRS is also a base of a contract with stakeholders (users/clients) that includes all the details about the functionality of the future product and how it is supposed to run. An SRS is used widely by software developers during the process of product or program development.

- An SRS includes both functional and non-functional requirements and use cases as well. A perfect SRS document takes into account, not just the way software will interact with other software or when it’s embedded in hardware, but also potential users and the ways they will interact with the software. It also contains references to tables and diagrams to get a clear understanding of all product-related details.
- An SRS document helps team members from different departments stay on the same and make sure all the requirements are fulfilled. This document also allows minimizing software development expenses and time.

- **BRS Document**
- So what is a BRS in software testing? BRS stands for a business requirement specification which is aimed to show how to meet the business requirements on a broader level. A BRS document is one of the most widely accepted specification documents. It's quite essential, and a BRS is usually created at the very beginning of the product's life cycle and describes the core product goals or needs the client wants to achieve with certain software or product. This one is usually created by a business analyst based on other stakeholders specifications and after a thorough analysis of the client company. Usually, the final version of the document is reviewed by the client to make sure that all business stakeholders' expectations are correct.
- A BRS includes all the requirements requested by a client. Generally, it consists of the product's purpose, users, the overall scope of work, all listed features and functions, usability and performance requirements. In this type of document use cases are not included, as well as diagrams and tables. A BRS is used mainly by upper and middle management, product's investors, business analysts.

- **FRS Document?**
- An FRS, or functional requirement specification is the document that describes all the functions that software or product has to perform. In fact, it’s a step-by-step sequence of all operations required to develop a product from very start to end. An FRS explains the details of how certain software components will behave during user interaction. This document is created by qualified software developers and engineers, and it is considered the result of close collaboration between testers and developers. The main difference, when compared to the SRS document, is that the FRS does not include use cases. It might also contain diagrams and tables, but this is not obligatory.

- This one is the most detailed document as it explains in-depth how the software is expected to function (including business aspects, compliance, security requirements) as it also has to satisfy all the requirements mentioned in both the SRS and BRS documents. An FRS helps developers to understand what product they are supposed to create, and software testers get a better understanding of different test cases and scenarios in which the product is expected to be tested.
- Getting a better understanding of the requirement specifications is paramount for successful job performance. Once you know more about these core testing documentation types, it will be much easier to work with them and increase the performance quality and productivity to achieve better results much faster.

- **Pre-Steps To Software Requirements Specification Review**
- **Step #1)** Documents go through multiple revisions, so make sure we have the right version of the referenced document, the SRS.
- **Step #2)** Establish guidelines on what is expected at the end of the review from each team member. In other words, decide on what deliverables are expected from this step – typically, the output of this step is to identify the test scenarios. Test scenarios are nothing but one line pointers of 'what to test' for certain functionality.
- **Step #3)** Also establish guidelines on how this deliverable is to be presented- I mean, the template.
- **Step #4)** Decide on whether each member of the team is going to work on the entire document or divide it among themselves. It is highly recommended that everyone reads everything because that will prevent knowledge concentration with certain team members.

- But in case of a huge project, with the SRS documents running close to 1000 pages, the approach of breaking up the document module wise and assigning to individual team members is most practical.

- **Step #5)** SRS review also helps in better understanding if there are any specific prerequisites required for the testing of the software.
- **Step #6)** As a by-product, a list of queries where some functionality is difficult to understand or if more information needs to be incorporated into functional requirements or if mistakes are made in SRS are identified.

- Software requirement specifications
- Create an Outline (Or Use an SRS Template) Your first step is to create an outline for your software requirements specification
- Start With a Purpose
- Give an Overview of What You'll Build
- Detail Your Specific Requirements
- Get Approval for the SRS.

- Business software specifications steps
- Project overview (including vision, objectives, and context)
- Success factors
- Project scope
- Stakeholder identification
- Business requirements
- Scope of the solution
- Project constraints (such as schedule and budget)

- Functional requirement specifications steps
- Requirements
- Objectives
- Functional specification
- Design change requests
- Logic specification
- User documentation
- Test plan
- The final product

- **Ideas to improve your Software Testing Process**
- Double confirm the requirements from the client. Make sure there are no ambiguities. If there are any get them sorted out at the onset. The entire quality is based on the user requirements; you cannot afford to go wrong in this area.
- Get involved from the beginning of the development process to get a comprehensive understanding of the AUT which would, in turn, help you identify areas needing more intensive testing
- During the design phase, you can start working on test cases based on design documents and with the help of the developers. This will mean less time for creating and updating test cases during the actual testing phase.
- The test plan should be written by a Manager or QA lead. The good test plan will cover deadlines, scope, schedule, risk identifications among other things. Make sure to have it reviewed by the project manager as well for concurrence.

- Ensure that the quality is maintained right from the beginning irrespective of whether it is development, testing or network support. QA's are like the gatekeepers of quality, they should not let anything pass which is not as per the expected quality.
- Testing as early as possible and as frequently as possible also helps get more detailed testing done. Do not worry about the number of modules delivered, if you have the bandwidth get the testing done for even a single module.
- User training should be provided to the QA team for a better understanding of the way the product is actually used by the clients or end users. This will also help them in testing the product better.
- While creating test cases make sure that all the permutation and combination of different types of data input are covered. Use the well-known principles like boundary value analysis (BVA), Equivalence Partitioning (EP), Decision Table This ensures that the code is tested thoroughly.

- Always ensure that testing goes in parallel with development. Make sure to test individual modules as and when they are ready for testing instead of waiting for the completion of the entire module.
- Make use of stubs and drivers to testing smaller modules which requiring input from other modules or are pushing data to other subsystems.
- Make a distinction between testable and non-testable items in the requirement document and include it as part of the test plan. This will be helpful in having confusion later on.
- Always ensure that one test case covers only 1 validation point. Though in some cases it may lead to duplication and extra effort for writing the test cases, this is needed for better traceability of the bugs.
- Test Coverage is an important aspect of ensuring a quality product. Use a traceability matrix to make sure each requirement is mapped to at least one or more test cases. This ensures complete coverage of the system

- Ensure that the test case documents are completed, reviewed and signed off by the stakeholders before the actual testing starts. Also, make sure that the test cases document is made available to the rest of the team for understanding and doubt clearance if any.
- Identify and group test cases based on their importance or priority. This is helpful at the time when only limited test cases can be run during a particular testing cycle. This grouping is also needed when you aim to run or re-run only a particular set based on their priority.
- You can also have test cases grouped based on functionality. This is useful at the time of regression testing done after a bug fix in any particular module. For example, if there has been a bug fix related to functionality in module A, then you can pick all test cases grouped as module A for the next regression test.
- Reviews are important to ensure correctness, following of guidelines etc. Ensure that all test cases are reviewed internally within the team (peer review) and once externally also to ensure correctness of the expected results and any other discrepancies.

- Involve tools to make your life simpler. Tools like Quality Centre, Test Complete, Rally etc. make your life much easier by taking care of a lot of metrics generation and data collection. These tools help you manage your testing projects in a much better and more professional manner.
- Automation is very important. It reduces manual effort on repetitive tasks. So it is always a good choice to go in for automation of modules or features which are stable and repetitive.
- Encourage team members to come up with innovative thoughts to make the testing process less time consuming but at the same time more fruitful. Many minds at work is always better than a single one.
- Always have clear precise and achievable deliverables. Keep a 10% buffer for retesting and other risks. Keeps you and your plan safe and viable.
- Version control all your artefacts. This is especially important if you are using automation. Make sure the code is version controlled so you are able to create multiple branches and each one should be easy to revert back to in the worst cases if needed.

- If feasible implement CI/CD. This ensures that each time a build is created the automation suite is run automatically to check if the basic build and its functionalities are in place or not. This helps save a lot of time as well as human error in forgetting to test a particular build.
- Delegate responsibility to team members for modules or smaller sections. This makes them work harder for ensuring the quality
- Always look at the software from the eyes of the user as a product and not as a project. The product is most important, think like a consumer when you use the product.
- **Test Environment and Teams**
- Has the environment set up in such a way that it replicates the production environment as closely as possible?
- Ensure that all API's, backend systems, user access, admin roles, test data and any other input needed are set up well ahead of the testing cycle. This is to ensure a loss of time in setting up when the testing window in open.
- Always have a healthy relationship with the development team. This will help in a quick turnaround in case of clarifications.

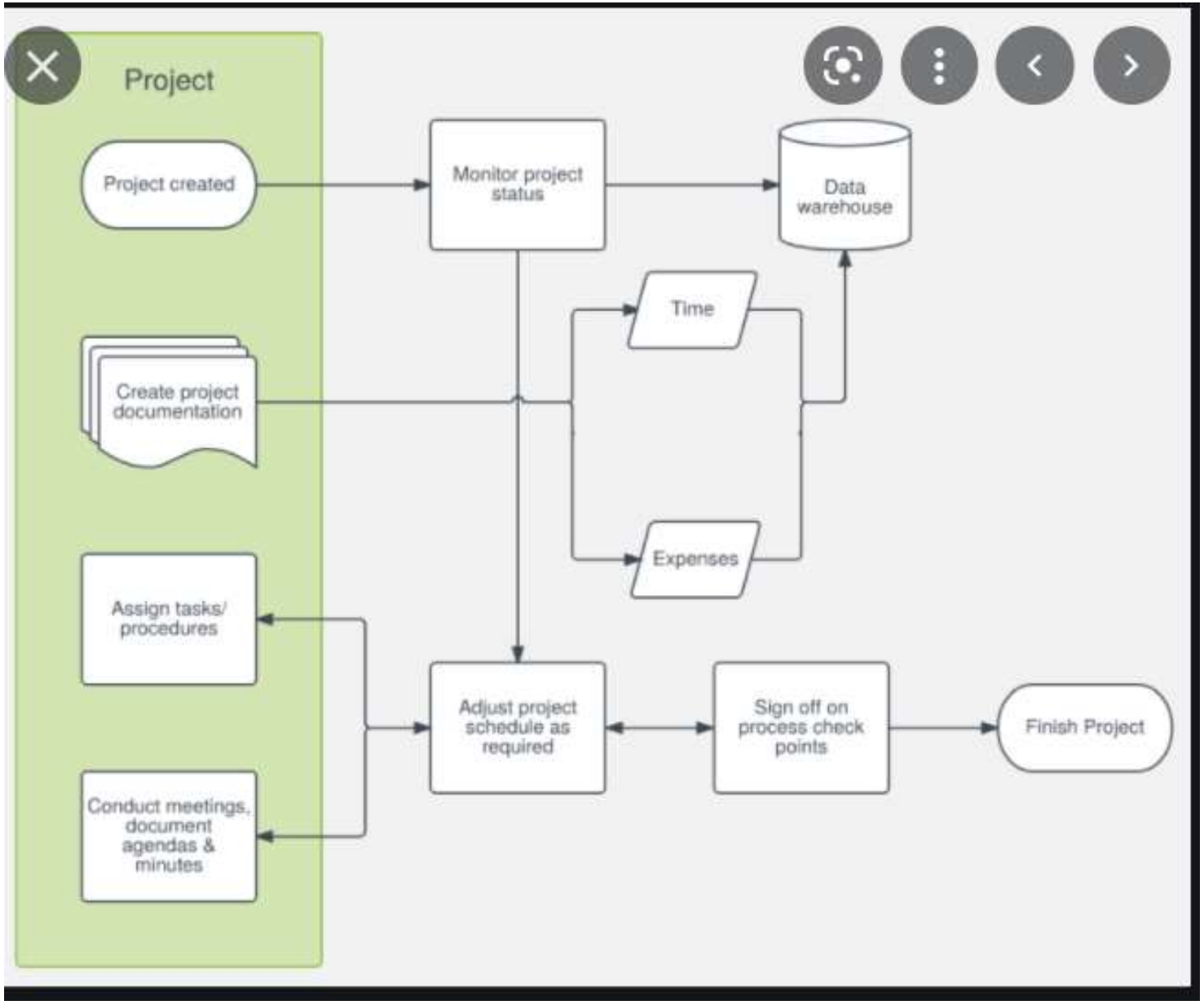
- Always ensure to have constant communication with the developer at every stage of the testing process so that everyone is aware of the testing activity in the process.
- Have at least one person from the testing team join the daily project meetings, so that the testing is aware of the modules coming up for testing in the next release and hence can start preparing for it.
- During the testing phase, have a daily meeting within the team, to clear any doubts and get additional information where ever needed. This is also helpful in tracking the testing progress.
- Each team has a role to play in the delivery of a good quality software product. Ensure that each team is assigned clear cut guidelines and areas, avoiding a conflict of interest.
- Establish a clear and precise communication plan along with SPOC (Single point of contact) list and escalation matrix which will contain a list people or contacts to be used in case of any escalations or in cases where information is needed immediately.
- It is always advisable to have the team members test a different module each time. The reason is twofold. For one the tester get a better understanding of the complete project instead of a single module and also new bugs can be found if the module is tested by fresh testers each time.

- Always be prepared for changes and contingencies this also includes the risks. Planning such situations help you avoid panic and face the scenario more gracefully too.
- Always maintain a risk register and review it frequently. You can also have a time slot set aside with all the stakeholders to review this risk register say weekly. Also, make sure that all risks and mitigation plans are captured accurately.
- Maintain and update simple metrics as defect found, defects fixed and percentage completion, quality level etc. Send out these metrics to all stakeholders and people involved preferably in a pictorial format, along with the comparison with previous week's data. This will have a much greater impact and visibility than you can imagine. Give it a try.
- Monitor your production app. And yes for the unfortunate bugs that do find their way to production, make sure they are included as part of the regression test suite with a high priority.

- **Overall Issues and Reporting**
- All the information regarding bugs and product development should be maintained in a common Share Point or repository for everyone to access and derive reports when needed.
- Always have a fresh and open mind when testing software. Have no assumptions. And always refer to the requirements document for doubts and clarifications.
- The entire product module should be divided into smaller parts to ensure each and every part is tested thoroughly
- Always ensure to have included the maximum details in the bug report which should include the user credentials, navigations or step to reproduce, expected and actual results, environment, version, logs and screenshots of the error.
- Always give clear instructions to reproduce an issue. Do not assume that the reader will know the application. Mention each object (button, link, edit box etc.) and action to be performed (click, enter, double click etc.) along with the pages navigated.

- Discuss and reach a consensus before the starting of the testing on the number of high priority and medium priority bugs can remain open or deferred at the time of movement to production
- Metrics are important criteria proving the effectiveness of any testing process. Choose appropriate metrics based on the project you are working on. Defect Density, Defect Removal Rate, Testing Efficiency, Test Case Execution Rate etc. are some commonly used software testing metrics.
- Verbal communications should be topped up by documenting the same in email and shared with relevant stakeholders. This not only creates better visibility but is also helpful in having a chronological record of communications.
- TDD – Test Driven Development is a newer form of development which takes the base from the test cases. This can also be employed for a better quality of code development.
- Have a UAT testing at least once before the product is released into production. This will be useful in identifying issues which are related to actual client usage. UAT testing is best performed by the users themselves.

- Always keep a check on the time taken by the development to fix the bugs. This data is part of an important metrics which is used to calculate the overall efficiency of the development team.
- Last but not least, trust your instincts. Yes, the QA people sense issues. So just follow your instincts, if you feel a certain area can have issues, mark my words there more chances that issues will be found in that part of the code.
- As important as testing and quality assurance is, we should give it the time and resources it needs. This pays off in the long run. These tips will be helpful to you for implementing a good test strategy as well as a quality product.


Project
Project created
Monitor project status
Data warehouse
Time
Create project documentation
Expenses
Assign tasks/ procedures
Adjust project schedule as required
Sign off on process check points
Finish Project
Conduct meetings, document agendas & minutes