## 

## 

## **SQA Intern Training | Devsinc**

***Assalam-o-Alaikum!*** We hope that you’re doing great! Super excited to have you join our **Internship Programme** and we really hope that you are excited to be a part of our ***Devsinc*** family as well. Looking forward to a great journey with you!!!

On behalf of ***Devsinc*** family and ***Devsinc’s QA Department***, we would like to extend our warmest welcome and best wishes to you and hope that this internship period would bring fruitful contributions to your career ahead.

The main focus and purpose of this training document is to ensure that you get the main idea as well as its implementation along with hands-on experience of the ***Software Quality Assurance*** essentials for your professional and personal development and growth in your career as an ***SQA Engineer***.

You are advised to read and implement the concepts in this document properly and ask your work buddy in case of any queries/confusions.

Also, please don’t forget to watch the relevant video tutorials, the links to which are given wherever necessary.

Watching tutorials other than the mentioned ones for self-learning are highly encouraged as they could prove to be the differentiating factor in developing/enhancing your skill set and professional aspects right from the very early stages.

**Main Objectives for an SQA Engineer**

* How to bridge the gap of testing from a developer’s perspective vs. from a tester’s perspective
* An introduction to unit testing methods - What developers do?
* Knowledge of Manual and Automated Testing - Where to use What?
* Unit, System, Integration, and User Acceptance Testing
* When to perform Smoke Testing and when to perform Sanity Testing?
* Bugs that unit tests can expose and ones that they will miss
* How to leverage (as a tester, not a developer) unit tests to find more bugs earlier
* How to plan testing on Agile Development Projects
* How to plan testing on Waterfall Development Projects
* Working knowledge of the Testing Tools but what are Testing Tools?
* Developing and Designing Test Suites
* Developing and Designing Test Scenarios and Test Cases
* New and sometimes forgotten strategies and test design methods including attack-based testing, model-based testing, and keyword-driven testing
* What is Test Driven Development?
* What is Behavior Driven Development?
* More effective test-case design with a black-box testing approach
* How to select the most effective practices to find bugs, optimize test planning and execution time?
* Understanding of BRD (Business Requirement Document)
* Understanding of SRS (Software Requirement Specification)
* Types of Testing Models
* How to effectively log bugs? Efficient Bug Life Cycle
* Components/Attributes of effective bugs logging?
* Bringing Quality to Bugs Logging
* Handling Repeating Test Cases
* Extending functionality in depth and finding out the edge/corner cases for effective testing
* Detailed Regression Testing Approaches
* Understanding of Test Coverage both Modular and Integration
* How to be the users’ advocate on a product/project?
* What is Smart Testing?
* When to stop testing? Know when testing is done?
* Acceptance Criteria | User Acceptance Testing
* Why is documentation important for an SQA Engineer?
* Effective Client Communication and Norms for Different Cultures
* How to achieve Client Satisfaction?
* Improving Quality Assurance Practices
* Increasing knowledge and skill set for best Quality Assurance Practices and Implementation

**Topics/Concepts to Cover and Implement**

**SDLC vs STLC:**

**SDLC:**

**SDLC (Software Development Life Cycle)** defines all the standard phases which are involved during the software development process. SDLC life cycle is a process of developing software through a phased manner in the following order

1. Requirements Gathering
2. Design the software
3. Build the Software
4. Test
5. Deployment
6. Maintenance**.**

Each stage has a definite entry and exit criteria along with deliverables.

**STLC:**

**Software Testing Life Cycle (STLC)** is the testing process that is executed in a well-planned manner. In the STLC process, various activities are carried out to improve the quality of the product. However, STLC phases only deal with testing and detecting errors but not development itself.

Different companies define different phases in STLC. However, the generic Software Test Life Cycle has the following stages.

1. Requirement Analysis
2. Test Planning
3. Test Development
4. Test Environment Setup
5. Test Execution & Closure

**Useful links:** <https://www.guru99.com/sdlc-vs-stlc.html>

<https://www.softwaretestingmaterial.com/sdlc-vs-stlc>

<https://www.youtube.com/watch?v=PhzYlopDCX0>

**Task: Prepare a document listing the in parallel comparison of the steps in SDLC and STLC.**

**BRD vs SRS vs FSD:**

The role of formulating a document is to understand fundamentals that will be compelled to develop robust software. Type of record expectation depends upon business type, their criteria, how company processes, and what kind of software is to be developed.

Let us understand common documents which are used by everyone and what we got to discuss ahead.

**1. Business Requirement Document (BRD):**

**Business Requirement Document** is termed as BRD. This document serves to find out practical requirements which may occur while developing any software. It elaborates on the interest of user requirements and developer requirements.

Following are some features of **BRD**:

* It is one of few documents created in any software development life-cycle.
* Documents featuring a company can provide a workable solution.
* It is used to find out what is expected from the system.
* It helps in finding out project sponsors.

**2. Software Requirement Specification (SRS):**

**Software Requirement Specification** is termed as an SRS document. This document serves as a detailed illustration of functional and nonfunctional requirements needed that the software should fulfill.

Following are some features of **SRS**:

* This document bridges the gap between user and developer.
* Documents board imaginations into a structural layout.
* Used for measuring initial costs and efforts.
* Works as an agreement between communicating parties.

**3. Functional Specifications Document (FSD):**

A **functional specification** is a formal document used to describe a product's intended capabilities, appearance, and interactions with users in detail for software developers.

The **functional specification** is a kind of guideline and continuing reference point as the developers write the programming code.

The method of preparing the specifications before the product is known as the "write the manual first" approach, serving as an outline of the finished program.

Typically, the **functional specification** for an application program with a series of interactive windows and dialogs with a user would show the visual appearance of the user interface (UI) and describe each of the possible user input actions and the program response actions.

**Useful links:** <https://www.guru99.com/brs-vs-srs-the-myth-busted.html>

<https://searchsoftwarequality.techtarget.com/definition/functional-specification>

<https://www.guru99.com/functional-requirement-specification-example.html>

**Task: Prepare a document and enlist 3 major differences between BRD, SRS, FSD.**

**Quality Assurance (QA) vs Quality Control (QC):**

### **What is Assurance?**

**Assurance** is provided by organization management, it means giving a positive declaration on a product which obtains confidence for the outcome. It gives a security that the product will work without any glitches as per the expectations or requests.

### **What is Quality Assurance (QA)?**



**Quality Assurance** is known as QA and focuses on preventing defects. Quality Assurance ensures that the approaches, techniques, methods and processes designed for the projects are implemented correctly.

**Quality Assurance** activities monitor and verify that the processes used to manage and create the deliverables have been followed and are operative.

**Quality Assurance** is a proactive process and is Prevention in nature. It recognizes flaws in the process. Quality Assurance has to be completed before Quality Control.

**What is Control?**

****

**Control** is to test or verify actual results by comparing it with the defined standards.

### **What is Quality Control (QC)?**

**Quality Control** is known as QC and focuses on identifying a defect. QC ensures that the approaches, techniques, methods and processes designed in the project are following correctly. QC activities monitor and verify that the project deliverables meet the defined quality standards.

**Quality Control** is a reactive process and is detection in nature. It recognizes the defects. Quality Control has to complete after Quality Assurance.



**What is The Difference in QA/QC?**

Many people think QA and QC are the same and interchangeable but this is not true. Both are tightly linked and sometimes it is very difficult to identify the differences. Fact is both are related to each other but they are different in origins. QA and QC both are part of Quality Management however QA is focusing on preventing defect while QC is focusing on identifying the defect.

**Useful links:** <https://www.guru99.com/quality-assurance-vs-quality-control.html>

<https://www.qualio.com/blog/quality-assurance-vs-quality-control>

<https://www.youtube.com/watch?v=j_KdgdLscU4>

**Task: Statement Coverage and Path Finding is performed in QA or QC? Explain with justifications according to your understanding.**

**Test Suite - Creation and Maintenance:**

A **TEST SUITE** is a collection of test cases. In automated testing, it can mean a collection of test scripts. In a test suite, the test cases / scripts are organized in a logical order. For example, the test case for registration will precede the test case for login.

**ISTQB Definition:**

* **test suite:** A set of test scripts or test procedures to be executed in a specific test run.

When you have hundreds / thousands of test cases, a test suite allows you to categorize them in a way that matches your planning or analysis needs.

For example, you could have a **test suite** for each of the core features of the software or you could have a separate test suite for a particular type of testing (for example, smoke test suite or security test suite).

An example of a test suite for purchasing a product could comprise of the following test cases:

* Test Case 1: Login
* Test Case 2: Add Products
* Test Case 3: Checkout
* Test Case 4: Logout

Note that each of the test cases above are dependent on the success of the previous test cases. For instance, it’s no use checking out if one cannot add products. Hence, if you are running a test suite in sequential mode, you can choose to stop the test suite execution if a single test case does not pass.

**Useful links:** <https://sqa.stackexchange.com/questions/9119/test-suite-vs-test-plan>

<https://www.testmonitor.com/blog/test-case-test-suite-test-run-whats-the-difference>

<https://help.testlodge.com/hc/en-us/articles/115007101447-Using-Test-Suites-and-Test-Cases>

**Test Scenarios vs Test Cases:**

**Test Case:**

A **test case** is a set of conditions or variables under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do.

**Test Scenario**:

The exhaustive testing is not possible due to the large number of data combinations and large number of possible paths in the software.

**Scenario testing** is to make sure that end to end functionality of the application under test is working as expected. Also check if all business flows are working as expected.

In scenario testing the tester needs to put his/her foot in the end user's shoes to check and perform the action as to how they are using the application under test.

In **scenario testing** the preparation of scenarios would be the most important part, to prepare the scenario tester needs to consult or take help from the client, stakeholder or developers.

**Useful links:** <https://www.guru99.com/test-case-vs-test-scenario.html>

<https://www.softwaretestingclass.com/what-is-difference-between-test-cases-vs-test-scenarios>

<https://www.youtube.com/watch?v=ePGAEJURzqU>

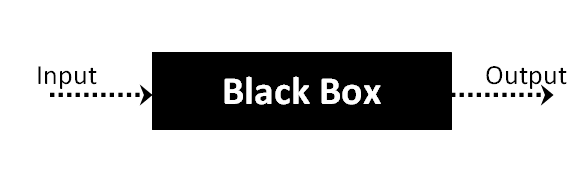
**Task: Create a Test Suite (A folder is your google drive and sheets within that folder) containing the test scenarios, test cases and relevant stuff.**

**Take www.facebook.com as a sample project. The use of screenshots and relevant attributes is encouraged.**

**Furthermore, you can also use any free Online QA Management Tools to achieve this task.**

**Manual Testing (Black-box Testing):**

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.



The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under **Black Box Testing**, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

**Useful links:** <https://www.guru99.com/black-box-testing.html>

<https://softwaretestingfundamentals.com/black-box-testing>

**Black-box Testing Types:**

Some of the most important **Black-box testing** types include:

* Smoke Testing
* Sanity Testing
* Regression Testing
* Functional Testing
* Non-functional Testing
* Acceptance Testing
* Positive Testing
* Negative Testing
* Boundary Value Analysis
* Exploratory Testing
* Integration Testing
* Equivalence Partitioning
* System Testing
* Ad-hoc Testing
* Component Testing
* Alpha/Beta Testing
* Cross-browser Testing
* Cross-platform Testing

**Useful links:** <https://www.guru99.com/types-of-software-testing.html>

**Automated Testing (White-box Testing):**

**White Box Testing** is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security.

In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective.

On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing.

The term **"WhiteBox"** was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings.

**Useful links:** <https://softwaretestingfundamentals.com/white-box-testing>

<https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example>

**White-box Testing Types:**

Some of the most important **White-box testing** types include:

* Control Flow Testing
* Data Flow Testing
* Branch Testing
* Statement Coverage
* Decision Coverage
* Modified Condition/Decision Coverage
* Prime Path Testing
* Path Testing
* Load Testing
* Stress Testing
* Volume Testing
* Mutation Testing

**Useful links:** <https://testinggenez.com/types-of-white-box-testing-techniques>

<https://www.testbytes.net/blog/white-box-testing>

**Automation Tools:**

Some of the most widely used tools/frameworks for **Automation Testing** are:

* Selenium
* Cypress
* Capybara
* JMeter
* JUnit
* LoadRunner
* RSpec

**API Testing:**

**API testing** is a type of software **testing** that involves **testing** application programming interfaces (**APIs**) directly and as part of integration **testing** to determine if they meet expectations for functionality, reliability, performance, and security.

Since **APIs** lack a GUI, **API testing** is performed at the message layer.

**Useful links:** <https://www.guru99.com/api-testing.html>

<https://smartbear.com/solutions/api-testing>

<https://www.katalon.com/resources-center/blog/api-testing-tips>

**REST vs SOAP APIs:**

**REST** **(Representational State Transfer)** is another standard, made in response to SOAP’s shortcomings. It seeks to fix the problems with SOAP and provide a simpler method of accessing web services.

**SOAP** **(Simple Object Access Protocol)** is a standards-based web services access protocol that has been around for a long time. Originally developed by Microsoft, SOAP isn’t as simple as the acronym would suggest.

**Useful links:** <https://smartbear.com/blog/test-and-monitor/soap-vs-rest-whats-the-difference>

<https://www.soapui.org/learn/api/soap-vs-rest-api>

**API Testing Tools:**

The most commonly used tool for **API Testing** is:

* Postman

**Postman is an application for testing APIs, by sending request to the web server and getting the response back.**

* It allows users to set up all the headers and cookies the **API** expects, and checks the response.
* Productivity can be increased using some of the **Postman** features, which are listed below.

**Useful links:** <https://medium.com/aubergine-solutions/api-testing-using-postman-323670c89f6d>

<https://www.postman.com/use-cases/api-testing-automation>

**Task: Setup Postman on your system/web browser and take 5 sample APIs. Perform API testing with the help of Get, Post and Put methods in Postman and prepare a report on your findings.**

**API Documentation:**

**API documentation** is a technical content deliverable, containing instructions about how to effectively use and integrate with an **API**.

**API** description formats like the OpenAPI/Swagger Specification have automated the **documentation** process, making it easier for teams to generate and maintain them.

**Useful links:**

<https://swagger.io/blog/api-documentation/what-is-api-documentation-and-why-it-matters>

<https://www.postman.com/api-documentation-tool>

<https://stoplight.io/api-documentation-guide/basics>

<https://medium.com/technical-writing-is-easy/api-documentation-solutions-d3719af2780f>

**Task: While performing the API testing, prepare the API documentation in parallel by using Postman’s documentation feature or Swagger Documentation.**

**Performance Testing:**

In software quality assurance, performance testing is in general a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload.

**Performance Testing** is a software testing process used for testing the speed, response time, stability, reliability, scalability and resource usage of a software application under particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and also known as “Perf Testing”.

The focus of **Performance Testing** is checking a software program's

* **Speed** - Determines whether the application responds quickly
* **Scalability** - Determines maximum user load the software application can handle.
* **Stability** - Determines if the application is stable under varying loads

**Useful links:** <https://www.guru99.com/performance-testing.html>

<https://www.tutorialspoint.com/software_testing_dictionary/performance_testing.htm>

<https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1>

**Load Testing vs Stress Testing:**

### **Load Testing**

**Load testing** is meant to test the system by constantly and steadily increasing the load on the system until it reaches the threshold limit. It is a subset of performance testing.

Load testing can be easily done by employing any of the suitable automation tools available in the market. WAPT and LoadRunner are two such famous tools that aid in load testing. Load testing is also famous for names like **Volume testing** and **Endurance testing**.

### **Stress Testing**

Under stress testing, various activities to overload the existing resources with excess jobs are carried out in an attempt to break the system down. **Negative testing**, which includes removal of the components from the system is also done as a part of stress testing.

Also known as **fatigue testing**, this testing should capture the stability of an application by testing it beyond its bandwidth capacity.

Thus, basically, stress testing evaluates the behavior of an application beyond peak load and normal conditions.



The purpose of **stress testing** is to ascertain the failure of the system and to monitor how the system recovers back gracefully. The challenge here is to set up a controlled environment before launching the test so that you can precisely capture the behavior of the system repeatedly under the most unpredictable scenarios.

**Useful links:** <https://www.softwaretestinghelp.com/what-is-performance-testing-load-testing-stress-testing>

<https://www.loadview-testing.com/load-testing-vs-stress-testing>

<https://www.youtube.com/watch?v=a8zV6GjExV8>

**Load/Stress Testing Tools:**

Following are some of the most commonly used **Load/Stress Testing** Tools:

* JMeter
* JUnit
* LoadRunner
* WebLoad
* NeoLoad
* GTMetrix

**Useful links:** <https://www.edureka.co/blog/performance-testing-tools>

<https://www.guru99.com/stress-testing-tutorial.html>

<https://www.tutorialspoint.com/jmeter/index.htm>

**Task: Setup JMeter on your system. Perform Load and Stress Testing of** [www.hotmail.com](http://www.hotmail.com) **and prepare a report on your findings.**

**The core focus should be on the response time against a certain applied load over the sample website.**

**Continuous Integration/Continuous Delivery (CI/CD):**

**Continuous Integration** **(CI)** is a development practice where developers **integrate** code into a shared repository frequently, preferably several times a day. Each **integration** can then be verified by an automated build and automated tests. While automated testing is not strictly part of **CI** it is typically implied.

**Continuous Delivery (CD)** is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, doing so manually. It aims at building, testing, and releasing software with greater speed and frequency.

**Useful links:** <https://continuousdelivery.com>

<https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>

<https://codeship.com/continuous-integration-essentials>

**CI/CD Tools and Frameworks:**

Following are the most commonly used **CI/CD tools** and **frameworks**:

* CircleCI
* Travis CI
* Gitlab CI
* Jenkins

**Useful links:** <https://www.guru99.com/top-20-continuous-integration-tools.html>

<https://travis-ci.com>

<https://circleci.com>

**Database Testing:**

**Database Testing** is a type of software **testing** that checks the schema, tables, triggers etc. of the **database** under **test**. It involves creating complex queries for performing the load or stress **test** on the **database** and checking its responsiveness. It checks integrity and consistency of data.

**Useful links:** <https://www.geeksforgeeks.org/software-testing-database-testing>

<https://www.guru99.com/data-testing.html>

**SQL Injection:**

**SQL injection** is a code **injection** technique that might destroy your database. **SQL injection** is one of the most common web hacking techniques.

**SQL injection** is the placement of malicious code in **SQL** statements, via web page input.

**Useful links:** <https://www.w3schools.com/sql/sql_injection.asp>

<https://portswigger.net/web-security/sql-injection>

<https://www.acunetix.com/websitesecurity/sql-injection>

**Penetration Testing:**

**Penetration testing**, also called **pen testing** or **ethical hacking**, is the practice of testing a computer system, network or web application to find security vulnerabilities that an attacker could exploit. **Penetration testing** can be automated with software applications or performed manually.

It is an authorized simulated cyberattack on a computer system, performed to evaluate the security of the system. Not to be confused with a vulnerability assessment.

**Useful links:** <https://searchsecurity.techtarget.com/definition/penetration-testing>

<https://www.imperva.com/learn/application-security/penetration-testing>

<https://www.guru99.com/learn-penetration-testing.html>

<https://www.cloudflare.com/en-gb/learning/security/glossary/what-is-penetration-testing>

**JS Attacks/Injection:**

**Javascript** is one of the most popular technologies and is most widely used for web pages and web applications.

It can be used for realizing different website functionalities. However, this technology can bring some security issues, which the developer and tester should be conscious about.

**Javascript** can be used not only for good purposes but for some malicious attacks too. One among that is **Javascript Injection**. The essence of **JS Injection** is to inject the Javascript code that will be run from the client-side.

### **Why is it Important to Test JS Injection?**

Many would ask if testing for **JS Injection** is really necessary.

Checking for **JS Injection** vulnerabilities is a part of security testing. Security testing is usually performed only if it was included in the project planning, as it requires time, a lot of attention and checking multiple details.

I have noticed that during the project's realization it is quite common to skip testing against any possible attacks – including **JS Injection**. This way the teams try to save the project’s time. However, this practice very often ends with customer’s complaints.

It should be known that security testing is highly recommended even if it is not included in the project plans. Checking for main possible attacks should be performed – at the same time must check for possible **JS Injection** vulnerabilities.

Leaving simple **Javascript Injection** vulnerabilities in the product may cost the product's quality and company’s reputation. Whenever I have learned to test against possible attacks and in general security testing, I never skip this part of testing. This way I am just more sure about the product’s quality.

**Useful links:** <https://www.softwaretestinghelp.com/javascript-injection-tutorial>

<https://www.checkmarx.com/2017/12/07/javascript-attacks-webviews>

**QA Management Tools/Plugins:**

Following are the most popularly used **QA/Test Management Tools**:

* TestRail
* Zephyr
* TestPad
* TestMoniter
* QMetry by JIRA

**Useful links:** <https://www.tutorialspoint.com/testrail/testrail_introduction.htm>

<https://www.youtube.com/watch?v=kBWwMUCYvMk>

<https://www.youtube.com/watch?v=bIt5jawrF8c>

<https://www.qmetry.com/resources/videos/qmetry-test-management-for-jira-test-execution>

**Task: Watch the tutorials for TestRail in detail. You will be asked questions related to it during your evaluation meeting.**

**Project Management Tools:**

Following are the most popularly used **Project Management Tools**:

* JIRA
* Trello
* Asana
* Clickup
* Pivotal Tracker

**Useful links:** <https://www.guru99.com/jira-tutorial-a-complete-guide-for-beginners.html>

<https://www.youtube.com/watch?v=NrHpXvDXVrw>

<https://www.youtube.com/watch?v=JCfYSorYNuA>

**Task: Research which Project Management Tool is free or use the free trial version of JIRA and create a sample test project there. Report some sample bugs and create sample user stories. You can use any sample site of your choice.**

**Software Development and Version Controlling:**

**Software Version Control** **(SVC)**, also called revision control, source control management, and **Versioning Control**, is a management strategy to track and store changes to a **Software Development** document or set of files that follow the development project from beginning to end-of-life.

**Useful links:** <https://www.smartsheet.com/software-version-control>

<https://www.atlassian.com/git/tutorials/what-is-version-control>

**Code Repositories - Github/Bitbucket:**

**Github:**

**Useful links:** <https://www.youtube.com/watch?v=RGOj5yH7evk>

**Bitbucket:**

**Useful links:** <https://www.youtube.com/watch?v=L0XVDTx-cA8>

**Task: Create your Github or Bitbucket accounts and setup a demo project on your system.**

**Now, create a branch named: test-branch-1 and take screenshots of each step. Place all of the screenshots in a document to show.**

**TDD vs BDD:**

**Test Driven Development (TDD)** is a development practice while **Behavioral Driven Development (BDD)** is a team methodology. In **TDD**, the developers write the tests while in **BDD** the automated specifications are created by users or testers (with developers wiring them to the code under test). For small, co-located, developer-centric teams, **TDD** and **BDD** are effectively the same.

**Useful links:** <https://www.pluralsight.com/blog/software-development/tdd-vs-bdd>

<https://www.softwaretestinghelp.com/tdd-vs-bdd>

<https://www.browserstack.com/guide/tdd-vs-bdd-vs-atdd>

**Task: Prepare a document describing the advantages and disadvantages of TDD and BDD.**

**Agile vs Waterfall:**

The two main development methodologies are **Agile** and **Waterfall**. They are commonly applied to software development, and thus, project management as well. The main difference between **Agile** and **Waterfall** is that **Waterfall** projects are completed sequentially whereas **Agile** projects are completed iteratively in a cycle.

**Useful links:** <https://www.guru99.com/waterfall-vs-agile.html>

<https://www.seguetech.com/waterfall-vs-agile-methodology>

**Scrum vs Kanban:**

Basically, **Kanban** can be applied to visualize and improve the flow of work, regardless of the methodology being used to do the work. **Scrum** is an iterative, incremental work method that provides a highly prescriptive way in which work gets completed. **Scrum** teams have defined processes, roles, ceremonies and artifacts.

**Useful links:** <https://www.planview.com/resources/guide/introduction-to-kanban/kanban-vs-scrum>

<https://www.guru99.com/scrum-vs-kanban.html>

<https://www.atlassian.com/agile/kanban/kanban-vs-scrum>

**Task: Prepare a document in which you are going to enlist that in which kind of projects, it is best to use Agile and in which it is a better approach to use Waterfall.**

**Also, enlist the appropriate usage of Scrum or Kanban in different projects according to their needs.**

**Feature/Functional Documentation:**

A **functional specification** is a formal document used to describe a product's intended capabilities, appearance, and interactions with users in detail for software developers. The **functional specification** is a kind of guideline and continuing reference point as the developers write the programming code.

A **functional specification** in systems engineering and software development is a document that specifies the functions that a system or component must perform. The documentation typically describes what is needed by the system user as well as requested properties of inputs and outputs.

**Useful links:** <https://searchsoftwarequality.techtarget.com/definition/functional-specification>

<https://www.justinmind.com/blog/functional-specification-documentation-quick-guide-to-making-your-own/>

<https://medium.com/@essentialdesign/what-is-a-functional-specification-document-fe97dc2084f5>

**Documentation Tools/Platforms - Confluence by JIRA/Google Docs:**

**Confluence by JIRA:**

**Useful links:** <https://www.youtube.com/watch?v=sQD9HUvFjSo>

**Task: Watch Confluence’s tutorials in detail and try to achieve the same functionality somehow, by using Google Docs in your drive.**

**Creating/Maintaining Test Sheets:**

**Task:** Based on your current learning, create the following **test cases sheets**:

1. Smoke Test Sheet
2. Sanity Test Sheet
3. Regression Test Sheet

**Practice Sites:**

[http://phptravels.com/demo](http://phptravels.com/demo/)

[http://thedemosite.co.uk](http://thedemosite.co.uk/)

[http://newtours.demoaut.com](http://newtours.demoaut.com/)

<http://www.way2automation.com/demo.html>

<http://automationpractice.com/index.php>

[http://demoqa.com](http://demoqa.com/)

**Task: Explore the above given sample sites and prepare Smoke, Sanity and Regression test sheets in your Google drive accordingly. Don’t forget to PRIORITIZE the modules.**

**Take any one of the above given sample sites under observation while creating the test sheets. You can also use multiple sites to create separate test sheets.**

**Effective Client Communication:**

**Effective Client Communication** is the way to win the client satisfaction that is one of the most important aspects of an **SQA Engineer’s** day to day life. Smart communication skills can make your day and win the client’s trust on your work giving your morale a real boost. This can really help and motivate you to be involved more passionately in your daily testing tasks.

Explore the links given below or read some of the blogs online and note down your findings in a document.

You will then be questioned during the evaluation meeting what were your findings on **Effective Client Communication**.

**Task: Prepare yourself for a client meeting considering all of the Effective Client Communication tips and tricks. You will be asked questions related to your findings on a few dummy modules that you have tested.**

**Useful links:** <https://www.successagency.com/di/effective-client-communication>

<https://www.teamgantt.com/blog/client-communication-skills>

<https://www.stptax.com/five-strategies-client-communication>

<https://www.stptax.com/five-strategies-client-communication>

**Client Communication Tools:**

**Slack:**

**Slack** is basically a messaging app on steroids. It's meant for teams and workplaces and can be used across multiple devices and platforms, and is equipped with robust features that allow you to not only chat one-on-one with associates but also in groups.