1. **Learning Objectives**
   1. **Primary Objectives**
2. To contribute to quality assurance of projects.
3. To manage work to meet requirements.
4. To work effectively with colleagues
5. To established interest in industrial/commercial activities.
6. To gain work experience enabling the student to apply what he learnt in college and acquire new skills.
   1. **Secondary Objectives**
7. Fundamentals of Testing.
8. Testing Throughout SDLC.
9. Static Techniques.
10. Test Design Techniques.
11. Test Management.
12. Tools Support.
13. **WEEKLY OVERVIEW OF OJT ACTIVITIES**

|  |  |  |  |
| --- | --- | --- | --- |
| **WEEK NO** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| **WEEK NO - 1** | 05-10-2021 | Tuesday | What is software/software testing |
| 05-10-2021 | Tuesday | Need of software testing |
| 05-10-2021 | Tuesday | Seven Principle of Testing |
| 06-10-2021 | Wednesday | Psychology of Testing |
| 06-10-2021 | Wednesday | Code of Ethics |
| 07-10-2021 | Thursday | Fundamental Test Processes |
| 07-10-2021 | Thursday | Test Planning |
| 07-10-2021 | Thursday | Test Specification |
| 07-10-2021 | Thursday | Test Execution |
| 07-10-2021 | Thursday | Test Recording |
| 07-10-2021 | Thursday | Test Completion |
| 08-10-2021 | Friday | What is SDLC |
| 08-10-2021 | Friday | What is STLC |
| 08-10-2021 | Friday | SDLC Models - Waterfall, Spiral, V, Agile |
| 09-10-2021 | Saturday | How to derive expected result |
| 09-10-2021 | Saturday | Test Case format |
| 09-10-2021 | Saturday | Important aspects of Test Cases |
| 09-10-2021 | Saturday | Software testing Levels & Types |
| 09-10-2021 | Saturday | Maintenance Testing |
| 09-10-2021 | Saturday | Software Quality - QA (Static) & QC (Dynamic) |
| 09-10-2021 | Saturday | Review Process |
| 09-10-2021 | Saturday | Tools |
| **WEEK NO - 2** | 11-10-2021 | Monday | Testing Methods Black box |
| 11-10-2021 | Monday | Black box techniques |
| 11-10-2021 | Monday | Decision Tables |
| 11-10-2021 | Monday | White box testing |
| 11-10-2021 | Monday | Grey box testing & comparison |
| 12-10-2021 | Tuesday | Test Organisation |
| 12-10-2021 | Tuesday | Test Estimation |
| 12-10-2021 | Tuesday | How to decide priority |
| 12-10-2021 | Tuesday | Test Monitoring & Control |
| 12-10-2021 | Tuesday | Test Progress |
| 12-10-2021 | Tuesday | Configuration Management |
| 13-10-2021 | Wednesday | Risk & Testing |
| 13-10-2021 | Wednesday | What is defect |
| 13-10-2021 | Wednesday | Defect Priority & Severity |
| 13-10-2021 | Wednesday | Defect Life Cycle |
| 13-10-2021 | Wednesday | Attributes of defect |
| 14-10-2021 | Thursday | Stubs & Drivers |
| 14-10-2021 | Thursday | Types of Test Tools |
| 14-10-2021 | Thursday | Effective Use of Tools |
| 14-10-2021 | Thursday | Introducing Tool into Organisation |

1. **Introduction**

This Year’s OJT was on QA Engineer and to learn the role of QA Engineer, I learnt a lot of skills like Scenarios Writing, Test Cases, Test Management, Defect logging, Testing, Test Leading, Bug Reporting and FRS. As all of them were new skills for me they were quite challenging but, at the end, I managed to understand them properly thanks to our excellent lecturer Mr. Shashikant Karulkar.

On-the-job training (OJT) is training that is delivered while an individual is performing tasks or processes related to their particular occupation. The student typically performs tasks that are essential to their job function with the supervision of a manager, coach or mentor. This type of training is typically used to broaden a student’s skill set and to increase productivity.

On-the-job training is an important topic of human resource management. It helps develop the individual and the prosperous growth of the organization. On the job training is a form of training provided at the workplace. During the training, employees are familiarized with the working environment they will become part of organization. Employees also get a hands-on experience using machinery, equipment, tools, materials, etc. Part of on-the-job training is to face the challenges that occur during the performance of the job. An experienced employee or a manager is executing the role of the mentor who through written or verbal instructions and demonstrations are passing on his/her knowledge and company-specific skills to the new employee. Executing the training on at the job location, rather than the classroom, creates a stress-free environment for the employees. On-the-job training is the most popular method of training not only in the United States but in the most of the developed countries, such as the United Kingdom, China, Russia, etc. Its effectiveness is based on the use of existing workplace tools, machines, documents and equipment, and the knowledge of specialists who are working in this field. On-the-job training is easy to arrange and manage and it simplifies the process of adapting to the new workplace. On-the-job training is highly used for practical tasks. It is inexpensive, and it doesn’t require special equipment that is normally used for a specific job. Upon satisfaction of completion of the training, the employer is expected to retain participants as regular employees.

1. **OJT Discussion**

**Subsection 4.1: How the Objectives were achieved?**

The objectives were achieved by performing the job-related activities in a timely and systematic manner.

1. First, an introduction given about software testing.
2. Why software testing necessary – performed some activities like writing scenarios on ATM machine.
3. Introduction was given about test planning, test specification, test execution.
4. Activities related writing scenarios, execution, bug reporting were performed.
5. Introduction was given about white box, black box, grey box testing techniques.
6. Contribute to quality assurance of project.
7. Provide data/information in standard formats.

**Subsection 4.2: What skills were acquired as per the job role QP during the OJT?**

**4.2.1 What is Software Testing?**

1. Computer software or just software is any set of machine-readable instructions that directs computers processor to perform specific operations.
2. Software is a set of programs, which is design to perform a well-defined function. A program is a sequence of instruction written to meet a particular requirement.
3. In other words, Computer software is a set of instructions in the form of programs, procedures, functions, data etc. executed to meet desired requirements.
4. IEEE Definitions –
5. Software testing is the process of analysing a software item to detect the differences between existing and required conditions (that is bugs) and to evaluate the features of the software item.
6. Reliability is the ability of a system or component to perform its required functions under started conditions for specified period of time.
7. Software testing can also be started as the process of validating and verifying that a software program/application/product:
8. Meets the business and technical requirements that guided its design and development.
9. Works as expected.
10. Can be implemented with the same characteristics.
11. Testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.
12. Testing involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

**4.2.2 Need of Software Testing**

1. Anything and everything created by humans is prone to have faults and defects. Some errors/faults do not have any severe impact but some faults are critical, which acts as a show-stopper, or will break the whole system.
2. So, in such cases it is important that all such high severity defects/faults are detected and resolved in advance, before the s/w is launched or implemented.
3. Perfection is very difficult.
4. People make errors, Errors can cause problems.
5. Deliverables can be defective.
6. Defects can cause failures; Failure can be a big problem.
7. To Err is human.
8. Error - mistake made, we are not perfect.
9. Fault - the result of an error. Also commonly known as bugs or defects.
10. Failure - the result of wrong behaviour, deviation from the expected.
11. The solution is to TEST (validate) at each stage.
12. Unfortunately, nobody is perfect and we all make mistakes. Sometimes this can be a misunderstanding of what is required of us, we are working under pressure such as delivery deadlines or sometimes we just get it wrong! Errors made early have a nasty habit of growing and getting worse.
13. If errors are present in software, they may cause problems immediately but they can also lie dormant and it may take a while before they surface. When errors have been made and lie undiscovered, the delivered software will be defective, which can lead to failures, which could mean severe problems for the business.
14. E.g. A leading supermarket chain had a '2 for 1' promotion on a hair conditioner. A customer noticed that when she had purchased two conditioners, not only had she only been charged for one of them, but also £3.50 had been deducted from her bill. She performed some further 'research' and discovered that for every one of the special purchases, £3.50 was deducted from the final bill.

Over the next days the woman visited all of the supermarket branches in her area and cleared Ore shelves of the conditioners. She estimated she saved herself £1000 ! This was caused by absence of a simple validation in software

**4.2.3 Seven Principles of Testing**

1. Testing shows presence of defects - Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
2. Exhaustive testing is impossible - Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. Instead of exhaustive testing, risk analysis and priorities should be used to focus testing efforts.
3. Early testing - To find defects early, testing activities shall be started as early as possible in the software or system development life cycle, and shall be focused on defined objectives.
4. Defect clustering - Testing effort shall be focused proportionally to the expected and later observed defect density of modules. A small number of modules usually contains most of the defects discovered during pre-release testing, or is responsible for most of the operational failures. (80:20 principle)
5. Pesticide paradox - If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this “pesticide paradox”, test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to find potentially more defects.
6. Testing is context dependent - Testing is done differently in different contexts. For example, safety-critical software is tested differently from an e-commerce site. E.g. Testing Bill Payment, Rocket System, Mobile Game – Different considerations are applicable in each case.
7. Absence-of-errors fallacy - Finding and fixing defects does not help if the system built is unusable and does not fulfil the users’ needs and expectations.

**4.2.4 Psychology of Testing**

1. Why do we test?
2. Primarily to find faults in the software, rather than demonstrating correctness.
3. This can be perceived as being a destructive process, not constructive.
4. How does this fit with the mindsets of the developer and tester?
5. Developer Characteristics
6. Specialized
7. Trained
8. Creative
9. Sensitive to criticism
10. Tester Characteristics
11. Trained
12. Experienced
13. Methodical
14. Persistent
15. Happy when they find faults
16. Good Communicators
17. Independence
18. Independent testing is more effective. The author should not test their own work.
19. Assumptions made are carried into testing.
20. People see what they want to see.
21. There can be emotional attachment.
22. There can be a vested interest not to find faults.

**4.2.5 Code of Ethics**

Involvement in software testing enables individuals to learn confidential and privileged information. A code of ethics is necessary, among other reasons to ensure that the information is not put to inappropriate use. The ISTQB states the following code of ethics:

PUBLIC - Certified software testers shall act consistently with the public interest.

CLIENT AND EMPLOYER - Certified software testers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.

PRODUCT - Certified software testers shall ensure that the deliverables they provide (on the products and systems they test) meet the highest professional standards possible.

JUDGMENT- Certified software testers shall maintain integrity and independence in their professional judgment.

Involvement in software testing enables individuals to learn confidential and privileged information. A code of ethics is necessary, among other reasons to ensure that the information is not put to inappropriate use. The ISTQB states the following code of ethics:

MANAGEMENT - Certified software test managers and leaders shall subscribe to and promote an ethical approach to the management of software testing.

PROFESSION - Certified software testers shall advance the integrity and reputation of the profession consistent with the public interest.

COLLEAGUES - Certified software testers shall be fair to and supportive of their colleagues, and promote cooperation with software developers.

SELF - Certified software testers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

**4.2.6 Fundamental Test Processes**

1. Test Planning
2. Determines the scope of the testing.
3. If company or industry policy, strategy or regulations refer to testing, the test plan shows how these rules are met identifies all exceptions to the rules. These rules may lay down what type of testing is required or how much testing is required.
4. Should be based on and derived from project base documentation: For example, planning of acceptance test: May be based on the business requirements; define the business processes to be tested; assign business risk to business processes. For example, planning of component test: May be based on the component specification; define the program path coverage; assign technical risk to the component processes.
5. Test Specification
6. Test Specification Is the design of Test Cases.
7. Test techniques defined in the test planning phase must be used. Test
8. A Test Case should encompass.
9. Test Execution
10. Having identified the test cases (during test specification) identify which tests are selected for this test run.
11. Dependent upon a release of software and what is contained within it.
12. Test execution is the execution of the identified tests.
13. Test Recording
14. The version of software being executed.
15. The actual outcome of the test.
16. The comparison of the actual outcome against the expected outcome.

* Identify the fault
* Make an initial assessment of whether this is a test error or an environment problem or a product problem
* Log the fault
* Assess the impact of the fault
* Record the failure of the test.
* If the actual outcome is the same as the expected outcome Record that the test has passed

**4.2.7 What is SDLC & STLC?**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software’s. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

The Software Testing Life Cycle (STLC) is a sequence of specific actions performed during the testing process to ensure that the software quality objectives are met. The STLC includes both verification and validation. Contrary to popular belief, software testing is not just a separate activity. It consists of a series of methodological activities to help certify your software product.

**4.2.8 SDLC Models - Waterfall, Spiral, V, Agile**

If you look at the project plan generated using the Waterfall model, you can see that none of the test planning or test creation as well as the test execution is performed until after the programs have been coded.

The Waterfall Model Testing tends to start with the delivery of the code, not the documentation.

Errors are more costly to correct due to testing starting later in the SDLC.

If the project is over budget and/or exceeding timescales, the testing phase will be reduced.

Testing is seen as a test execution process.

1. Advantages:
2. This model is simple and easy to understand and use.
3. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
4. In this model phases are processed and completed one at a time. Phases do not overlap.
5. Waterfall model works well for smaller projects where requirements are very well understood.
6. Disadvantages:
7. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought.
8. No working software is produced until late during the life cycle.
9. High amounts of risk and uncertainty.
10. Not a good model for complex and object-oriented projects.
11. Poor model for long and ongoing projects.
12. Not suitable for the projects where requirements are at a moderate to high risk of changing.

It is an incremental approach to development and testing whereby the full system requirements may not be known at the start of the project (i.e. the users know their basic requirements but do not really know exactly what the complete system should look like).

The initial requirements are defined, designed, built and tested (with review points after each activity) and those requirements enhanced and built upon in further iterations of the define, design, build and test activities.

The system is implemented at the end of the required number of iterations.

1. Incremental approach
2. Build a little, test a little
3. Uses prototypes
4. Components/functions developed in parallel
5. Developments are time-boxed, delivered and assembled into a working prototype.
6. Advantages:
7. High amount of risk analysis hence, avoidance of Risk is enhanced.
8. Good for large and mission-critical projects.
9. Strong approval and documentation control.
10. Additional Functionality can be added at a later date.
11. Software is produced early in the software life cycle.
12. Disadvantages:
13. Can be a costly model to use.
14. Risk analysis requires highly specific expertise.
15. Project’s success is highly dependent on the risk analysis phase.
16. Doesn’t work well for smaller projects.

The V Model was introduced to address some of the problems associated with the Waterfall model. Within the V model testing is not seen as a phase that happens at the end of development, it is recognized that for every stage of development an equal stage of testing needs to occur.

It also recognizes that the test preparation, for example test planning and test creation can be separated from test execution. The test preparation is not dependant on the code being delivered and can occur much earlier in the SDLC.

The documentation shown on the (left-hand side of the V model is not rigid - organizations may call documents produced by different names, may merge the documents shown, or may have additional documents that they produce. The right-hand side of the V model is more rigid in terms of naming conventions, but organizations may choose to exclude certain levels of testing, depending upon the project in question

1. Advantages:
2. Simple and easy to use.
3. Testing activities like planning, test designing happen well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
4. Proactive defect tracking – that is defects are found at early stage.
5. Avoids the downward flow of the defects.
6. Works well for small / Medium projects where requirements are easily understood.
7. Disadvantages:
8. Software is developed during the implementation phase, so no early prototypes of the software are produced.
9. If any changes happen in midway, then the test documents along with requirement documents has to be updated.

Agile method involves Iterative development. This model is based on Agile Manifesto (initially published in Feb 2001)

Agile testing is testing practice for a project using agile methodologies. It treats development as the customer of testing and emphasizing the test-first code later paradigm (TDD).

It focuses on peer code reviews (white box testing), extensive unit testing and regression testing. This implies high usage of automation of test cases.

**4.2.9 How to Derive Expected Result**

Expected Result is an ideal result that the tester should get after the test case is performed. It’s usually documented together with the test case. It’s usually compared with actual result, and if the actual result differs from the expected one, the difference is documented and called a bug.

**4.2.10 Test Case Format**

A **test case format** is a document that comes under the FRS which allows testers to develop the test cases for a particular test scenario in order to verify whether the features of an application are working as intended or not. Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected results, post-conditions, and actual results.

**4.2.11 Important Aspects of Test Cases**

1. Objective: Here the tester mentions what he plans to achieve with that particular test case.
2. Steps to Follow: Here the tester mentions the steps that need to be followed to achieve the objective.
3. Expected Output: Here the tester mentions the output which is expected as per the requirements provided.
4. Actual Output: Here the tester mentions the actual output achieved by following the steps.
5. Pass/Fail: If the tester fails to achieve the ‘Expected Output’ by following the steps then he will mention ‘Fail’ against that particular test case. Similarly, if the tester is able to achieve the ‘Expected Output’ then he will mention ‘Pass’ against the test case.