**Software Testing Basics**

**Software testing** is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect-free in order to produce a quality product.

**Types:** How we actually test

1. Manual Testing
2. Automation Testing

**Methods:** Before/After product is code complete

1. Static Testing (Verification)
2. Dynamic Testing (Validation)

**Approaches:**

1. White box
2. Black box : Functional vs Non-Functional
3. Grey box

**Levels:**

1. Unit Testing
2. Integration Testing
3. System Testing ( end-to-end)
4. Acceptance Testing

**Test Artifacts:**

1. Test plan
2. Test case
3. Traceability matrix
4. Test script
5. Test suite
6. Release Note
7. Test data or Test Fixture
8. Test harness

**Principles of Software Testing:**

1. Testing shows the presence of defects
2. Exhaustive testing is impossible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context-dependent
7. Absence of error – a fallacy

**STLC – Software Testing Life Cycle**

**case\_esac Demo Program – README**

* This program 03\_case\_esac.sh reads an input parameter from command line.
* The input is assumed to be one of the tools supported by program.
* If input tool supported, then program with print a one-line description of the tool and exit.
* Program will give an error for unsupported tool input
* Additionally, program also validates if the program received the cli argument as input or not.

**Usage:**

**$ bash 03\_case\_esac.sh <input\_tool\_name>**

Valid options for input\_tool\_name are jira, git, jenkins, sonarqube

**Example:**

$ **bash 03\_case\_esac.sh git**

git is one of source-control management tool.

**Best Testcases for** case..esac **program**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TestcaseID** | **Description** | **Type** | **Expected** | **Actual** | **Test Result** |
| case\_testcase\_001 | Run the program with ‘jenkins’ as input at cli | Functional Regression | jenkins is a CI/CD tool. | **jenkins is a CI/CD tool.** | PASS |
| case\_testcase\_002 | Run the program and give ‘jira’ as input as cli | Functional Regression | jira is a bug reporting and agile tool. | **jira is a bug reporting and agile tool.** | PASS |
| case\_testcase\_003 | Run the program and give ‘git’ as input at cli | Functional Regression | git is one of source-control management tool. | **git is one of source-control management tool.** | PASS |
| case\_testcase\_004 | Run the program and give ‘sonarqube’ as input at cli | Functional Regression | sonarqube is code coverage and static analysis tool | **sonarqube is code coverage and static analysis tool** | PASS |
| case\_testcase\_005 | Run the program without any input at cli | Negative | ERROR: At least one argument is required for this script | ERROR: At least one argument is required for this scriptl | PASS |
| case\_testcase\_006 | Run the program and give ‘java’ as input at cli | Negative | invalid tool ‘java’. valid inputs are: jenkins, git, jira, sonarqube. | invalid tool ‘java’. valid inputs are: jenkins, git, jira, sonarqube. | PASS |