Manual Testing

<https://www.javatpoint.com/manual-testing>

Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not.

Manual testing is essential because one of the [software testing](https://www.javatpoint.com/software-testing-tutorial) fundamentals is "100% automation is not possible."

## **Types of Manual Testing**

There are various methods used for manual testing. Each technique is used according to its testing criteria. Types of manual testing are given below:

* White Box Testing
* Black Box Testing
* Gray Box Testing

### **White-box testing**

The white box testing is done by Developer, where they check every line of a code before giving it to the Test Engineer. Since the code is visible for the Developer during the testing, that's why it is also known as White box testing.

### **Black box testing**

The black box testing is done by the Test Engineer, where they can check the functionality of an application or the software according to the customer /client's needs. In this, the code is not visible while performing the testing; that's why it is known as black-box testing.

### **Gray Box testing**

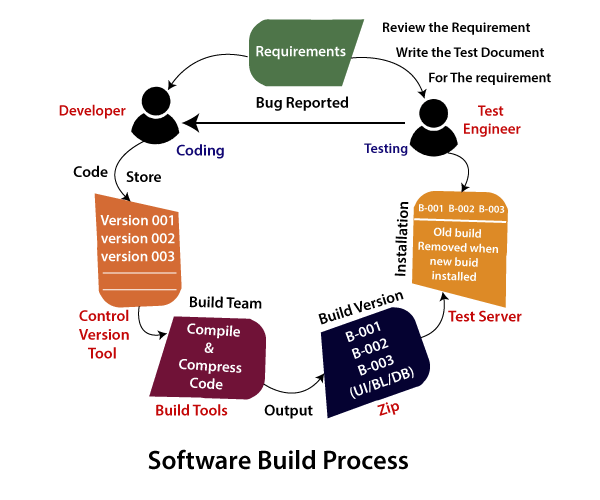
Gray box testing is a combination of white box and Black box testing. It can be performed by a person who knew both coding and testing. And if the single person performs white box, as well as black-box testing for the application, is known as Gray box testing.

## **How to perform Manual Testing**

* First, tester observes all documents related to software, to select testing areas.
* Tester analyses requirement documents to cover all requirements stated by the customer.
* Tester develops the test cases according to the requirement document.
* All test cases are executed manually by using Black box testing and white box testing.
* If bugs occurred then the testing team informs the development team.
* The Development team fixes bugs and handed software to the testing team for a retest.

## **Software Build Process**

* Once the requirement is collected, it will provide to the two different team development and testing team.
* After getting the requirement, the concerned developer will start writing the code.
* And in the meantime, the test engineer understands the requirement and prepares the required documents, up to now the developer may complete the code and store in the **Control Version tool**.
* After that, the code changes in the UI, and these changes handle by one separate team, which is known as the **build team**.
* This build team will take the code and start compile and compress the code with the help of a build tool. Once we got some output, the output goes in the zip file, which is known as **Build** (application or software).Each Build will have some unique number like (B001, B002).
* Then this particular Build will be installed in the test server. After that, the test engineer will access this test server with the help of the Test URL and start testing the application.
* If the test engineer found any bug, he/she will be reported to the concerned developer.
* Then the developer will reproduce the bug in the test server and fix the bug and again store the code in the Control version tool, and it will install the new updated file and remove the old file; this process is continued until we get the stable Build.
* Once we got the stable Build, it will be handed over to the customer.



### **Note1**

* Once we collect the file from the Control version tool, we will use the build tool to compile the code from high-level language to machine level language. After compilation, if the file size will increase, so we will compress that particular file and dumped into the test server.
* This process is done by **Build team**, **developer** (if build team is not there, a developer can do it) or the **test lead** (if the build team directly handle the zip and install the application to the test server and inform the test engineer).
* Generally, we can't get a new Build for every bug; else, most of the time will be wasted only in creating the builds.

### **Note2**

**Build team**

The main job of the build team is to create the application or the Build and converting the high-level language into low-level language.

**Build**

It is software, which is used to convert the code into application format. And it consists of some set of features and bug fixes that are handed over to the test engineer for testing purposes until it becomes stable.

**Control version tool**

It is a software or application, which is used for the following purpose:

* In this tool, we can save different types of files.
* It is always secured because we access the file from the tools using the same login credentials.
* The primary objective of the tools is to track the changes done for the existing files.

### **Example of Build process**

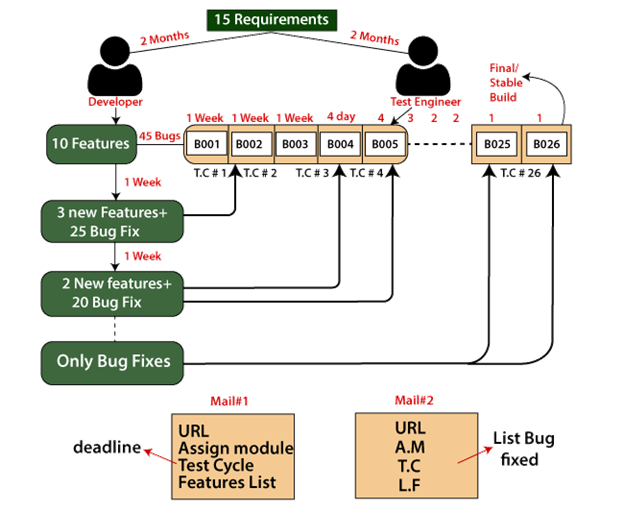
Let see one example to understand how to build process work on the real scenarios:

As soon as the test engineer gets the bug, they will send it to the developers, and they need some time to analyze; after that, he/she only fixes the bug (Test engineer can't give the collection of bug).

The developer is decided how many bugs he can fix according to their time. And the test engineer is decided, which bug should be fixed first according to their needs because the test engineers cannot afford to stop testing.

And the test engineer getting the mail, they can only know that which bug is fixed by the **list of the bug fixes**.

The time will increase because at the first Build, and developers should write the code in the different features. And at the end, he/she can only do the bug fixes and the numbers of days will be decreased.



### **Note3**

**Test cycle**

The test cycle is the time duration given to the test engineer to test every Build.

**Differences between the two build**

The bugs found in one build and can be fixed any of the future Build, which depends on the test engineer's requirement. Each new Build is the modified version of the old one, and these modifications could be the bug fixes or adding some new features.

**How frequently we were getting the new Build**

In the beginning, we used to get weekly builds, but in the latest stage of testing, when the application was getting stable, we used to get the new Build once in 3 days, two days, or a daily basis as well.

**How many builds we get**

If we consider one year of any project duration, we got 22-26 builds.

**When we get the bug fixes**

Generally, we understand the bug fixes only after the test cycle is completed, or the collection of bugs is fixed in one build, and handover in the next builds.

## **Advantages of Manual Testing**

* It does not require programming knowledge while using the Black box method.
* It is used to test dynamically changing GUI designs.
* Tester interacts with software as a real user so that they are able to discover usability and user interface issues.
* It ensures that the software is a hundred percent bug-free.
* It is cost-effective.
* Easy to learn for new testers.

## **Disadvantages of Manual Testing**

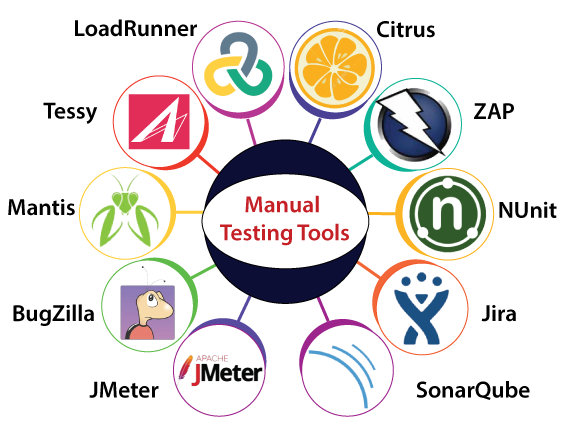
* It requires a large number of human resources.
* It is very time-consuming.
* Tester develops test cases based on their skills and experience. There is no evidence that they have covered all functions or not.
* Test cases cannot be used again. Need to develop separate test cases for each new software.
* It does not provide testing on all aspects of testing.
* Since two teams work together, sometimes it is difficult to understand each other's motives, it can mislead the process.

## **Manual testing tools**

In manual testing, different types of testing like unit, integration, security, performance, and bug tracking, we have various tools such as [Jira](https://www.javatpoint.com/jira-tutorial), [Bugzilla](https://www.javatpoint.com/bugzilla), Mantis, Zap, NUnit, Tessy, LoadRunner, Citrus, SonarQube, etc. available in the market. Some of the tools are open-source, and some are commercial.

For more information about testing tools, refers to the below link:

<https://www.javatpoint.com/software-testing-tools>



Let's us understand them one by one:

### **LoadRunner**

It is most commonly used performance testing tools. LoadRunner is mainly used to support performance testing for the wide range of procedures, number of approaches, and application environments.

The main purpose of executing the LoadRunner tool is to classify the most common sources of performance issues quickly.



**Features of LoadRunner**

* LoadRunner tool contains n-numbers of applications, which reduces the time to understand and describe the reports.
* We can get thorough performance test reports by using the LoadRunner tool.
* It will reduce the cost of distributed load testing and also offer the operational tool for deployment tracking.

### **Citrus**

Citrus is an integration testing tool, which is the most commonly used test framework. It is written in **Java programming** language. It is mostly used to request and respond to server-side and client-side and validate the XML [JSON](https://www.javatpoint.com/json-tutorial) files.

To accomplish the end-to-end use case testing, citrus supports several HTTP, JMS, and SOAP protocols.



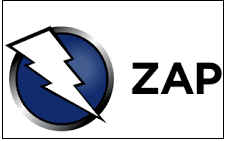
**Characteristics of Citrus**

Following are some of the important features of Citrus tool:

* It is used to send and receive messages.
* Citrus is available as both an open-source and a licensed in the market.
* It delivers a low-cost solution.
* We can authenticate the database by using the citrus tool.
* It will describe the sequence of messages, offer the test plan, and document the test coverage.
* It creates the message and verifies the responses.

### **ZAP**

ZAP is an open-source web application security scanner. It is stands for **Zed Attack Proxy**. Just like some other tools, it is also written in the [JAVA programming language](https://www.javatpoint.com/java-tutorial). It is the most effective **Open Web Application Security Projects** [OWASP].



**Features of ZAP**

* It supports many operating systems such as Windows, Linux, OS X.
* It has a plugin-based architecture.
* It contains an online marketplace that permits us to add new or updated features.
* ZAP's GUI control panel is easy to use.

### **NUnit**

NUnit is one of the most frequently used unit testing tools. It is an open-source tool and primarily derived from the **JUnit**.

It was completely written in the [C# programming language](https://www.javatpoint.com/c-sharp-tutorial) and suitable for all [.Net languages](https://www.javatpoint.com/net-framework).

In other words, we can say that the NUnit tool is entirely redesigned to become the advantage of many .Net language qualities. **For example:**

* **Reflection-related capabilities.**
* **Other custom attributes.**



**Characteristics of NUnit**

* It allows the assertions as a static method of the advantage class.
* It sustains the data-driven tests.
* It supports several platforms, like .NET core Xamarin mobile, Silverlight, and efficient framework.
* The ability of NUnit help us to execute the tests simultaneously.
* It uses a console runner to load and execute the tests.

### **JIRA**

The most regularly used bug tracking tool is **JIRA**, which is an open-source tool. It is used for bug tracking, project management, and issue tracking.

In this tool, we can easily track all kinds of bugs or defects related to the software and produced by the test engineers.



**Features of JIRA**

* It is a time-saving tool.
* Jira is used to track the defects and issues.
* It is used to establish the documentation tasks.
* Jira is a very useful tool in tracking the improvement of our documentation.

To get the complete information about the Jira tool, refer to the below link: <https://www.javatpoint.com/jira-tutorial>.

### **SonarQube**

Another testing tool of manual testing is SonarQube, which improves our workflow with continuous code quality and code security. It is flexible with the use of plug-ins.

It is completely written in the JAVA programming language. It offers fully automated evaluation and integration with Ant, Maven, Gradle, MSBuild, and constant integration tools. SonarQube has the ability to record a metrics history and gives the evolution graph.



**Features of Sonarqube**

Below are some of the significant features of the SonarQube tool:

* It supports several programming languages like C, C++, Python, JAVA, HTML, CSS, VB.NET, PHP, COBOL, PL/SQL, etc.
* Under the GNU Lesser General Public License, Sonarqube is freely available.
* SonarQube is affiliate with some important external tools like GitHub, Active Directory, LDAP, and others.
* SonarQube merged with Visual Studio, Eclipse, and IntelliJ IDEA development environments due to the **SonarLint** plug-ins.

### **JMeter**

JMeter is an open-source tool that is used to test the performance of both static and dynamic resources and dynamic web applications.

It is completely designed on the JAVA application to load the functional test behavior and measure the application's performance.

It facilitates users or developers to use the source code for the development of other applications.



**Features of JMeter**

Below are some of the essential characteristics of JMeter:

* It is platform-independent, which accepts a JVM like **Windows, Mac, and Linux, etc.**
* It supports a user-friendly GUI, which is interactive and straightforward.
* It is incredibly extensible to load the performance test in multiple types of servers.

For more information about JMeter, refers to the below link:

<https://www.javatpoint.com/jmeter-tutorial>.

### **Bugzilla**

Another bug tracking tool used in manual testing is [**Bugzilla**](https://www.javatpoint.com/bugzilla).

It is most widely used by many organizations to track the various bugs of the application.

Bugzilla is an open-source tool that helps the customer and the client to keep track of the defects. Bugzilla is also considered a test management tool because in this, we can easily link other test case management tools such as ALM, Quality Centre, etc.



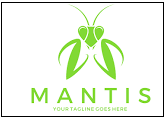
**Features of Bugzilla**

Bugzilla has some additional features which help us to report the bug easily:

* It supports various operating systems such as Windows, Linux, and Mac.
* With the help of Bugzilla, we can list a bug in several formats.
* User preferences can measure email notification.
* Bugzilla has advanced searching capabilities.

### **Mantis**

Mantis is a web-based bug tracking system. ManitsBT stands for **Mantis Bug Tracker**. It is used to follow the software defects and performed in the PHP programming language. It is also an open-source tool.



**Features of Mantis**

Some of the standard features of the particular tool are as follows:

* With the help of this tool, we have full-text search accessibility.
* Audit trails of changes made to issues.
* It provides the revision control system integration.
* Revision control of text fields and notes

To get more details about bug tracking tools, refer to the following link: <https://www.javatpoint.com/defect-or-bug-tracking-tool>.

### **Tessy**

Another integration testing tool is **Tessy**, which is used to perform the integration and unit testing for the embedded software. It also helps us to discover the code coverage of the software or an application.

It can easily manage the entire test organization, including business needs, test management, coverage quantity, and traceability.

Tessy contains three primary functions, which are as follows:

* Test Interface Editor (TIE)
* Test Data Editor (TDE)
* Workspace.



**Features of TESSY**

The standard features of the TESSY are as follows:

* It produces the test report for the test execution results.
* It supports various programming languages such as C and C++.
* Tessy is used to evaluate the interface of the function and describes the variable used by that function.

For more information about integration testing tools, refers to the following link: <https://www.javatpoint.com/integration-testing-tools>.

## **Overview**

In this article, we have seen detailed information about **Manual testing, which includes the definition of manual testing, the need of manual testing, type of manual testing, manual testing tools, the process of manual testing, and some important benefits and drawbacks of it.**

Finally, we can say that, it is a process where the test engineer needs to be very persistent, innovative, and responsive.

In manual testing, the test engineer needs to think and perform like end-user interpretation.

In order to implement manual testing, a test engineer needs productive skill and imagination. And they need to think of multiple situations or scenarios to test a specific application.

Even though we can test nearly all applications with the help of automation testing at present, still manual testing is necessary as it is the base of software testing.