**Web Application Testing: 8 Step Guide to Web Testing:**

### What is Web Testing?

Web Testing in simple terms is checking your web application for potential bugs before its made live or before code is moved into the production environment.

During this stage issues such as that of web application security, the functioning of the site, its access to handicapped as well as regular users and its ability to handle traffic is checked.

## Web Application Testing Checklist:

Some or all of the following testing types may be performed depending on your web testing requirements.

## 1. Functionality Testing:

This is used to check if your product is as per the specifications you intended for it as well as the functional requirements you charted out for it in your developmental documentation. Testing Activities Included:

**Test**all **links**in your webpages are working correctly and make sure there are no broken links. Links to be checked will include -

* Outgoing links
* Internal links
* Anchor Links
* MailTo Links

**Test Forms**are working as expected. This will include-

* Scripting checks on the form are working as expected. For example- if a user does not fill a mandatory field in a form an error message is shown.
* Check default values are being populated
* Once submitted, the data in the forms is submitted to a live database or is linked to a working email address
* Forms are optimally formatted for better readability

**Test Cookies** are working as expected. Cookies are small files used by websites to primarily remember active user sessions so you do not need to log in every time you visit a website. Cookie Testing will include

* Testing cookies (sessions) are deleted either when cache is cleared or when they reach their expiry.
* Delete cookies (sessions) and test that login credentials are asked for when you next visit the site.

**Test HTML and CSS** to ensure that search engines can crawl your site easily. This will include

* Checking for Syntax Errors
* Readable Color Schemas
* Standard Compliance. Ensure standards such W3C, OASIS, IETF, ISO, ECMA, or WS-I are followed.

**Test business workflow**- This will include

* Testing your end - to - end workflow/ business scenarios which takes the user through a series of webpages to complete.
* Test negative scenarios as well, such that when a user executes an unexpected step, appropriate error message or help is shown in your web application.

**Tools that can be used**: [**QTP**](https://www.guru99.com/quick-test-professional-qtp-tutorial.html) , IBM Rational , [**Selenium**](https://www.guru99.com/selenium-tutorial.html)

## ****2. Usability testing****:

[Usability Testing](https://www.guru99.com/usability-testing-tutorial.html) has now become a vital part of any web based project. It can be **carried out by testers** like you **or a small focus group** similar to the target audience of the web application.

**Test**the site **Navigation**:

* Menus, buttons or Links to different pages on your site should be easily visible and consistent on all webpages

**Test**the **Content**:

* Content should be legible with no spelling or grammatical errors.
* Images if present should contain an "alt" text

**Tools that can be used**: Chalkmark, Clicktale, Clixpy and Feedback Army

## ****3.Interface Testing****:

Three areas to be tested here are - Application, Web and Database Server

* **Application:** Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.
* **Web Server**: Test Web server is handling all application requests without any service denial.
* **Database Server:**Make sure queries sent to the database give expected results.

**Test system response** when **connection between the three layers**(Application, Web and Database) **cannot be established** and appropriate message is shown to the end user.

**Tools that can be used**: AlertFox, Ranorex

## 4. Database Testing:

Database is one critical component of your web application and stress must be laid to test it thoroughly. Testing activities will include-

* Test if any errors are shown while executing queries
* Data Integrity is maintained while creating, updating or deleting data in database.
* Check response time of queries and fine tune them if necessary.
* Test data retrieved from your database is shown accurately in your web application

**Tools that can be used**: [**QTP**](https://www.guru99.com/quick-test-professional-qtp-tutorial.html), [**Selenium**](https://www.guru99.com/selenium-tutorial.html)

## 5. Compatibility testing.

Compatibility tests ensures that your web application displays correctly across different devices. This would include-

**Browser Compatibility Test**: Same website in different browsers will display differently. You need to test if your web application is being displayed correctly across browsers, JavaScript, AJAX and authentication is working fine. You may also check for[Mobile](https://www.guru99.com/mobile-testing.html)Browser Compatibility.

The rendering of web elements like buttons, text fields etc. changes with change in **Operating System**. Make sure your website works fine for various combination of Operating systems such as Windows, Linux, Mac and Browsers such as Firefox, Internet Explorer, Safari etc.

**Tools that can be used**: NetMechanic

## 6. Performance Testing:

This will ensure your site works under all loads. Testing activities will include but not limited to -

* Website application response times at different connection speeds
* Load test your web application to determine its behavior under normal and peak loads
* Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.
* Test if a crash occurs due to peak load, how does the site recover from such an event
* Make sure optimization techniques like gzip compression, browser and server side cache enabled to reduce load times

**Tools that can be used**: [**Loadrunner**](https://www.guru99.com/loadrunner-tutorials.html), [**JMeter**](https://www.guru99.com/jmeter-tutorials.html)

## 7. Security testing:

[Security Testing](https://www.guru99.com/what-is-security-testing.html) is vital for e-commerce website that store sensitive customer information like credit cards. Testing Activities will include-

* Test unauthorized access to secure pages should not be permitted
* Restricted files should not be downloadable without appropriate access
* Check sessions are automatically killed after prolonged user inactivity
* On use of SSL certificates, website should re-direct to encrypted SSL pages.

**Tools that can be used**: Babel Enterprise, BFBTester and CROSS

## 8. Crowd Testing:

You will select a large number of people (crowd) to execute tests which otherwise would have been executed a select group of people in the company. Crowdsourced testing is an interesting and upcoming concept and helps unravel many a unnoticed defects.

**Tools that can be used**: People like you and me !!!. And yes , loads of them!

This concludes almost all testing types applicable to your web application.

As a Web-tester its important to note that web testing is quite an arduous process and you are bound to come across many obstacles. One of the major problems you will face is of course **deadline pressure**. Everything is always needed yesterday! The number of times the **code will need changing**is also taxing. Make sure you **plan your work** and know clearly what is expected of you. Its best **define all the tasks** involved in your web testing and then **create a work chart for accurate estimates and planning**.

# Complete Web Application Testing Checklist

While testing the web applications, one should consider the below mentioned checklist. The below mentioned checklist is almost applicable for all types of web applications depending on the business requirements.

The web application testing checklist consists of-

* [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html)
* Functional Testing
* [Compatibility Testing](https://www.guru99.com/compatibility-testing.html)
* Database Testing
* [Security Testing](https://www.guru99.com/what-is-security-testing.html)
* [Performance Testing](https://www.guru99.com/performance-testing.html)

Now let's look each checklist in detail:

## Usability Testing

**What is Usability Testing?**

* Usability testing is nothing but the User-friendliness check.
* In Usability testing, the application flow is tested so that a new user can understand the application easily.
* Basically, system navigation is checked in Usability testing.

**What is the purpose or Goal of Usability testing?**

A Usability test establishes the ease of use and effectiveness of a product using a standard Usability test practices.

**Usability Test Scenarios:**

* Web page content should be correct without any spelling or grammatical errors
* All fonts should be same as per the requirements.
* All the text should be properly aligned.
* All the error messages should be correct without any spelling or grammatical errors and the error message should match with the field label.
* Tool tip text should be there for every field.
* All the fields should be properly aligned.
* Enough space should be provided between field labels, columns, rows, and error messages.
* All the buttons should be in a standard format and size.
* Home link should be there on every single page.
* Disabled fields should be grayed out.
* Check for broken links and images.
* Confirmation message should be displayed for any kind of update and delete operation.
* Check the site on different resolutions (640 x 480, 600x800 etc.?)
* Check the end user can run the system without frustration.
* Check the tab should work properly.
* Scroll bar should appear only if required.
* If there is an error message on submit, the information filled by the user should be there.
* Title should display on each web page
* All fields (Textbox, dropdown, radio button etc) and buttons should be accessible by keyboard shortcuts and the user should be able to perform all operations by using keyboard.
* Check if the dropdown data is not truncated due to the field size and also check whether the data is hardcoded or managed via administrator.

## Functional Testing:

**What is Functional Testing?**

* Testing the features and operational behavior of a product to ensure they correspond to its specifications.
* Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.

**What is the purpose or Goal of Functional testing?**

* The goal of[Functional Testing](https://www.guru99.com/functional-testing.html) is to verify whether your product meets the intended functional specifications mentioned in your development documentation.

**Functional Test Scenarios:**

* Test all the mandatory fields should be validated.
* Test the asterisk sign should display for all the mandatory fields.
* Test the system should not display the error message for optional fields.
* Test that leap years are validated correctly & do not cause errors/miscalculations.
* Test the numeric fields should not accept the alphabets and proper error message should display.
* Test for negative numbers if allowed for numeric fields.
* Test division by zero should be handled properly for calculations.
* Test the max length of every field to ensure the data is not truncated.
* Test the pop up message ("This field is limited to 500 characters") should display if the data reaches the maximum size of the field.
* Test that a confirmation message should display for update and delete operations.
* Test the amount values should display in currency format.
* Test all input fields for special characters.
* Test the timeout functionality.
* Test the Sorting functionality.
* Test the functionality of the buttons available
* Test the Privacy Policy & FAQ is clearly defined and should be available for users.
* Test if any functionality fails the user gets redirected to the custom error page.
* Test all the uploaded documents are opened properly.
* Test the user should be able to download the uploaded files.
* Test the email functionality of the system.
* Test the[Java](https://www.guru99.com/java-tutorial.html)script is properly working in different browsers (IE, Firefox, Chrome, safari and Opera).
* Test to see what happens if a user deletes cookies while in the site.
* Test to see what happens if a user deletes cookies after visiting a site.
* Test all the data inside combo/list box is arranged in chronological order.

## Compatibility Testing:

**What is Compatibility testing?**

* Compatibility testing is used to determine if your software is compatible with other elements of a system with which it should operate, e.g. Browsers, Operating Systems, or hardware.

**What is the purpose or Goal of Compatibility testing?**

* The purpose of Compatibility testing is to evaluate how well software performs in a particular browser, Operating Systems, hardware or software.

**Compatibility Test Scenarios:**

* Test the website in different browsers (IE, Firefox, Chrome, Safari and Opera) and ensure the website is displaying properly.
* Test the HTML version being used is compatible with appropriate browser versions.
* Test the images display correctly in different browsers.
* Test the fonts are usable in different browsers.
* Test the java script code is usable in different browsers.
* Test the Animated GIF's across different browsers.

**Tool for Compatibility Testing:**

Spoon.net: Spoon.net provides access to thousands of applications (Browsers) without any installs. This tool helps you to test your application on different browsers on one single machine**.**

## Database Testing:

**What is Database Testing?**

* In Database testing backend records are tested which have been inserted through the web or desktop applications. The data which is displaying in the web application should match with the data stored in the Database.

**To perform the Database testing, the tester should be aware of the below mentioned points**:

* The tester should understand the functional requirements, business logic, application flow and database design thoroughly.
* The tester should figure out the tables, triggers, store procedures, views and cursors used for the application.
* The tester should understand the logic of the triggers, store procedures, views and cursors created.
* The tester should figure out the tables which get affected when insert update and delete (DML) operations are performed through the web or desktop applications.

**With the help of the above mentioned points, the tester can easily write the test scenarios for Database testing.**

**Test Scenarios for Database Testing:**

* Verify the database name: The database name should match with the specifications.
* Verify the Tables, columns, column types and defaults: All things should match with the specifications.
* Verify whether the column allows a null or not.
* Verify the Primary and foreign key of each table.
* Verify the Stored Procedure:
* Test whether the Stored procedure is installed or not.
* Verify the Stored procedure name
* Verify the parameter names, types and number of parameters.
* Test the parameters if they are required or not.
* Test the stored procedure by deleting some parameters
* Test when the output is zero, the zero records should be affected.
* Test the stored procedure by writing simple[SQL](https://www.guru99.com/sql.html)queries.
* Test whether the stored procedure returns the values
* Test the stored procedure with sample input data.
* Verify the behavior of each flag in the table.
* Verify the data gets properly saved into the database after the each page submission.
* Verify the data if the DML (Update, delete and insert) operations are performed.
* Check the length of every field: The field length in the back end and front end must be same.
* Verify the database names of QA, UAT and production. The names should be unique.
* Verify the encrypted data in the database.
* Verify the database size. Also test the response time of each query executed.
* Verify the data displayed on the front end and make sure it is same in the back end.
* Verify the data validity by inserting the invalid data in the database.
* Verify the Triggers.

## What is Security Testing?

Security Testing involves the test to identify any flaws and gaps from a security point of view.

**Test Scenarios for Security Testing:**

1. Verify the web page which contains important data like password, credit card numbers, secret answers for security question etc should be submitted via HTTPS (SSL).
2. Verify the important information like password, credit card numbers etc should display in encrypted format.
3. Verify password rules are implemented on all authentication pages like Registration, forgot password, change password.
4. Verify if the password is changed the user should not be able to login with the old password.
5. Verify the error messages should not display any important information.
6. Verify if the user is logged out from the system or user session was expired, the user should not be able to navigate the site.
7. Verify to access the secured and non secured web pages directly without login.
8. Verify the “View Source code” option is disabled and should not be visible to the user.
9. Verify the user account gets locked out if the user is entering the wrong password several times.
10. Verify the cookies should not store passwords.
11. Verify if, any functionality is not working, the system should not display any application, server, or database information. Instead, it should display the custom error page.
12. Verify the SQL injection attacks.
13. Verify the user roles and their rights. For Example The requestor should not be able to access the admin page.
14. Verify the important operations are written in log files, and that information should be traceable.
15. Verify the session values are in an encrypted format in the address bar.
16. Verify the cookie information is stored in encrypted format.
17. Verify the application for Brute Force Attacks

## What is Performance Testing?

Performance Testing is conducted to evaluate the compliance of a system or component with specified performance requirements.

**General Test scenarios:**

* To determine the performance, stability and scalability of an application under different load conditions.
* To determine if the current architecture can support the application at peak user levels.
* To determine which configuration sizing provides the best performance level.
* To identify application and infrastructure bottlenecks.
* To determine if the new version of the software adversely had an impact on response time.
* To evaluate product and/or hardware to determine if it can handle projected load volumes.

**How to do Performance testing? By Manual Testing or by Automation**

Practically it is not possible to do the Performance Testing manually because of some drawbacks like:

* More number of resources will be required.
* Simultaneous actions are not possible.
* Proper system monitoring is not available.
* Not easy to perform the repetitive task.

Hence to overcome the above problems we should use Performance Testing tool. Below is the list of some popular testing tools.

* Apache JMeter
* Load Runner
* Borland Silk Performer.
* Rational Performance Tester
* WAPT
* NEO LOAD

# Banking Domain Application Testing

Banking Applications directly deal with confidential financial data. It is mandatory that all the activities performed by banking software run smoothly and without any error. Banking software perform various functions like transferring and depositing fund, balance inquiry, transaction history, withdrawal and so on. Testing banking application assures that these activities are not only executed well but also remain protected from hackers.

In this tutorial, we will learn

* [What is Domain in Testing?](https://www.guru99.com/banking-application-testing.html#1)
* [Why Domain Knowledge Matters?](https://www.guru99.com/banking-application-testing.html#2)
* [Introduction to Banking Domain](https://www.guru99.com/banking-application-testing.html#3)
* [Characteristics of a banking application](https://www.guru99.com/banking-application-testing.html#4)
* [Stages of testing banking applications](https://www.guru99.com/banking-application-testing.html#5)
* [Sample Test Case for Net Banking Login Application](https://www.guru99.com/banking-application-testing.html#6)
* [Challenges in testing banking domain & their Mitigation](https://www.guru99.com/banking-application-testing.html#7)
* [Summary](https://www.guru99.com/banking-application-testing.html#8)

[**Join our Live Banking Testing Project for Free**](https://www.guru99.com/live-testing-project.html)

## What is Domain in Testing?

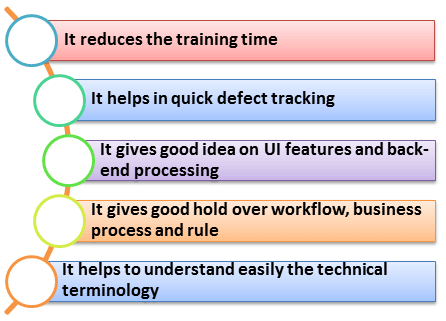
Domain is nothing but the industry for which the software testing project is created. When we talk about software project or development, this term is often referred. For example, Insurance domain, Banking domain, Retail Domain, Telecom Domain, etc.

[](https://www.guru99.com/images/6-2015/052515_0721_BankingAppl1.png)

Usually, while developing any specific domain project, domain expert help is sought out. Domain expert are master of the subject, and he may know the inside-out of the product or application.

## Why Domain Knowledge Matters?

Domain knowledge is quintessential for testing any software product, and it has its own benefits like

[](https://www.guru99.com/images/6-2015/052515_0721_BankingAppl2.png)

## Banking Domain Knowledge - Introduction

Banking domain is huge, and basically it is sub-characterized into two sectors

1. **Traditional banking sector**
2. **Service based banking sector**

Below is the table of the services these two sub-sectors of banking encompass

|  |  |
| --- | --- |
| **Traditional banking sector** | * Core banking * Corporate banking * Retail banking |
| **Service based banking sector** | * Core * Corporate * Retail * Loan * Trade finance * Private banking * Consumer finance * Islamic banking * Customer delivery channels/Front end delivery |

Based on the scope of your project you may need to test one or all of the above service offerings. Before you begin testing, ensure you have enough background on the service being tested.

## Characteristics of a Banking Application

Before you begin testing, it's important to note the standard features expected of any banking application. So that, you can gear your test efforts to achieve these characteristics.

A standard banking application should meet all these characteristics as mentioned below.

* It should support thousands of concurrent user sessions
* A banking application should integrate with other numerous applications like trading accounts, Bill pay utility, credit cards, etc.
* It should process fast and secure transactions
* It should include massive storage system.
* To troubleshoot customer issues it should have high auditing capability
* It should handle complex business workflows
* Need to support users on multiple platforms (Mac, Linux, Unix, Windows)
* It should support users from multiple locations
* It should support multi-lingual users
* It should support users on various payment systems (VISA, AMEX, MasterCard)
* It should support multiple service sectors (Loans, Retail banking etc.)
* Foolproof disaster management mechanism

## Test Phases in Testing Banking Applications

For testing banking applications, different stages of testing include

* **Requirement Analysis:** It is done by business analyst; requirements for a particular banking application are gathered and documented
* **Requirement Review:**Quality analysts, business analysts, and development leads are involved in this task. The requirement gathering document is reviewed at this stage, and cross-checked to ensure that it does not affect the workflow
* **Business Requirements Documentation:**Business requirements documents are prepared by quality analysts in which all reviewed business requirements are covered
* **Database Testing:**It is the most important part of bank application testing. This testing is done to ensure data integrity, data loading, data migration, stored procedures, and functions validation, rules testing, etc.
* **Integration Testing:**Under [Integration Testing](https://www.guru99.com/integration-testing.html) all components that are developed are integrated and validated
* **Functional Testing:**The usual software testing activities like[Test Case](https://www.guru99.com/test-case.html)preparation, test case review and test case execution is done during this phase
* **Security Testing:**It ensures that the software does not have any security flaws. During test preparation, QA team needs to include both negative as well as positive test scenarios so as to break into the system and report it before any unauthorized individual access it. While to prevent from hacking, the bank should also implement a multi-layer of access validation like a one-time password. For [Security Testing](https://www.guru99.com/what-is-security-testing.html), automation tools like IBM AppScan and HPWebInspect are used while for [Manual Testing](https://www.guru99.com/manual-testing.html) tools like Proxy Sniffer, Paros proxy, HTTP watch, etc. are used
* **Usability Testing:**It ensures that differently able people should be able to use the system as normal user. For example, ATM with hearing and Braille facility for disabled
* **User Acceptance Testing:**It is the final stage of testing done by the end users to ensure the compliance of the application with the real world scenario.

## Sample Test Case for Net Banking Login Application

Security is prime for any banking application. Therefore, during test preparation, QA team should include both negative and positive test scenarios in order to sneak into the system and report for any vulnerabilities before any unauthorized individual get access to it. It not only involves writing negative test cases but may also include destructive testing.

Following are generic test cases to check any banking application

|  |
| --- |
| Sample test cases |
| For Admin |
| * Verify Admin login with valid and Invalid data |
| * Verify admin login without data |
| * Verify all admin home links |
| * Verify admin change password with valid and invalid data |
| * Verify admin change password without data |
| * Verify admin change password with existing data |
| * Verify admin logout |
| For new Branch |
| * Create a new branch with valid and invalid data |
| * Create a new branch without data |
| * Create a new branch with existing branch data |
| * Verify reset and cancel option |
| * Update branch with valid and invalid data |
| * Update branch without data |
| * Update branch with existing branch data |
| * Verify cancel option |
| * Verify branch deletion with and without dependencies |
| * Verify branch search option |
| For New Role |
| * Create a new role with valid and invalid data |
| * Create a new role without data |
| * Verify new role with existing data |
| * verify role description and role types |
| * Verify cancel and reset option |
| * Verify role deletion with and without dependency |
| * verify links in role details page |
| For customer & Visitors |
| * Verify all visitor or customer links |
| * Verify customers login with valid and invalid data |
| * Verify customers login without data |
| * Verify bankers login without data |
| * Verify bankers login with valid or invalid data |
| For New users |
| * Create a new user with valid and invalid data |
| * Create a new user without data |
| * Create a new user with existing branch data |
| * Verify cancel and reset option |
| * Update user with valid and invalid data |
| * Update user with existing data |
| * Verify cancel option |
| * Verify deletion of the user |

## Challenges in testing Banking domain & their Mitigation

Challenges tester might face during testing banking domain are

|  |  |
| --- | --- |
| **Challenge** | **Mitigation** |
| * Getting access to production data and replicating it as test data, for testing is challenging | * Ensure that test data meets regulatory compliances requirements and guidelines * Maintain the data confidentiality by following techniques like data masking, synthetic test data, testing system integration, etc. |
| * The biggest challenge in testing banking system is during the migration of the system from the old system to the new system like testing of all the routines, procedures and plans. Also how the data will be fetched, uploaded and transferred to the new system after migration | * Ensure Data Migration Testing is complete * Ensure Regression Test cases are executed on old and new systems, and the results match. |
| * There may be the cases where requirements are not documented well and may lead to functional gaps in test plan * Many non-functional requirements are not fully documented, and testers do not know whether to test it or not | * The test should participate in the project right from Requirement Analysis phases and should actively review the Business Requirements |
| * The most important point is to check whether the said system follows the desired policies and procedures | * Compliance or Regulatory Policies testing must be done |
| * The scope and the timelines increases as banking application are integrated with other application like internet or[Mobile](https://www.guru99.com/mobile-testing.html)banking | * Ensure Time budget for Integration Testing is accounted if your banking application has many external interfaces |

## Summary

Banking domain is the most vulnerable area for cyber-theft, and safeguarding the software requires precise testing. This tutorial gives a clear idea of what it takes for banking domain testing and how important it is. One must understand that -

* Majority of banking software are developed on **Mainframe**and**Unix**
* Testing helps to lessen possible glitches encounter during software development
* Proper testing and compliance to industry standards, save companies from penalties
* Good practices help develop good results, reputation and more business for companies
* Both manual and automated testing have respective merits and usability

**Join our** [Live Banking Domain Testing Project](https://www.guru99.com/live-testing-project.html)

# Payment Gateway Testing Tutorial with Sample Test Cases

A payment gate-way system is an e-commerce application service that **approves** credit card payment for online purchases. Payment gateways safeguard the credit card details by encrypting sensitive information like credit card numbers, account holder details and so on. This information is passed safely between the customer and the merchant and vice versa.

Modern payment gateways also **securely approve**payments via debit cards, electronic bank transfers, cash cards, reward points etc.

[](https://www.guru99.com/images/6-2015/payment_gateway.png)

In this tutorial we will learn

* [Types of Payment Gateway System](https://www.guru99.com/payment-gateway-testing-tutorial-with-sample-test-cases.html#1)
* [Testing Types for Payment Gateway System](https://www.guru99.com/payment-gateway-testing-tutorial-with-sample-test-cases.html#2)
* [Test Preparation for Testing Payment Gateway](https://www.guru99.com/payment-gateway-testing-tutorial-with-sample-test-cases.html#3)
* [Sample Test Cases for Payment Gateway Testing](https://www.guru99.com/payment-gateway-testing-tutorial-with-sample-test-cases.html#4)
* [Things to consider before Buying Gateway Package](https://www.guru99.com/payment-gateway-testing-tutorial-with-sample-test-cases.html#5)

[**Join our Live Payment Gateway Testing Project for Free**](https://www.guru99.com/live-payment-gateway-project.html)

## Types of Payment Gateway System

* **Hosted Payment Gateway**:

Hosted payment gateway system direct customer away from e-commerce site to gateway link during payment process. Once the payment is done, it will bring customer back to e-commerce site. For such type of payment you don't need merchant id, example of hosted payment gateway are PayPal, Noche and WorldPay.

* **Shared Payment Gateway**:

In shared payment gateway, while processing payment customer is directed to payment page and stays on the e-commerce site. Once the payment detail is filled, the payment process proceeds. Since it does not leave the e-commerce site while processing payment, this mode is easy and more preferable, example of shared payment gateway is eWay, Stripe.

## Testing Types for Payment Gateway System

Testing for Payment Gateway should include

**Functional Testing**: It is the act of testing base functionality of the payment gateway. It is to verify whether the application behaves in same way as it is supposed to be like handling orders, calculation, addition of VAT as per the country etc.

**Integration**: Test integration with your credit card service.

**Performance**: Identify various performance metrics like highest possible number of users coming through gateways during specific day and converting them to concurrent users

**Security**: You need to perform a deep security pass for Payment Gateway.

## Test Preparation for Testing Payment Gateway

Before you begin testing -

* Collect proper test data for the dummy credit card number for maestro, visa, master etc.
* Collect payment gateway information like Google wallet, Paypal or else
* Collect payment gateway document with error codes
* Understand the session and parameters passed through application and payment gateway
* Understand and test the amount related information passed through query string or variable or session
* Along with payment gateway language check the language of the application
* Under the various settings of payment gateway like currency format, subscriber data collected.

## Sample Test Cases for Payment Gateway Testing

|  |  |
| --- | --- |
| **Sr#** | **Test Cases** |
| 1 | During the payment process try to change the payment gateway language |
| 2 | After successful payment, test all the necessary components, whether it is retrieved or not |
| 3 | Check what happens if payment gateway stops responding during payment |
| 4 | During the payment process check what happens if session ends |
| 5 | During the payment process check what happens in back end |
| 6 | Check what happens if payment process fails |
| 7 | Check the Data-base entries whether they store credit card details or not |
| 8 | During payment process check error pages and security pages |
| 9 | Check settings of pop-up blocker, and see what happens if pop up blocker is on and off |
| 10 | Between payment gateway and application check buffer pages |
| 11 | Check on successful payment, a success code is send to the application and a confirmation page is show to the user |
| 12 | Verify whether the transaction processes immediately or processing is hand to your bank |
| 13 | After successful transaction check if the payment gateway returns to your application |
| 14 | Check all format and messages when successful payment process |
| 15 | Unless you don't have an authorization receipt from payment gateway, good should not be shipped |
| 16 | Inform the owner for any transaction processed through e-mail. Encrypt the content of the mail |
| 17 | Check the amount format with currency format |
| 18 | Check if each of the payment options are selectable |
| 19 | Check if each listed payment option opens the respective payment option according to specification |
| 20 | Verify whether the payment gateway defaults to the desired debit/credit card option |
| 21 | Verify the default option for debit card shows card selection drop down menu |

## Things to consider before Buying Gateway Package

* If you have bought a shopping cart package, find out about its compatibility
* If shopping gateway package is due, ask the payment gateway provider for a list of supported applications
* The gateway must offer Address Verification System Protection
* Find out the types of transaction protection being offered
* Check what types of debit or credit cards are accepted by your chosen payment gateway
* Check the transaction fees levied by payment gateway
* Check whether the gateways collect the payment right on the form or direct to another page to complete the purchase

Use the comments section below to contribute more test cases on Payment Gateway Testing

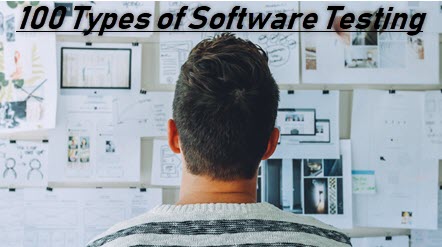
**Types of Software Testing: 100 Examples of Different Testing Types**

### What is a Software Testing Type?

Software Testing Type is a classification of different testing activities into categories, each having, a defined test objective, test strategy, and test deliverables. The goal of having a testing type is to validate the Application Under Test(AUT) for the defined Test Objective.

For instance, the goal of Accessibility testing is to validate the AUT to be accessible by disabled people. So, if your Software solution must be disabled friendly, you check it against Accessibility Test Cases.

A list of**100 types of Software Testing Types** along with definitions. A must read for any QA professional.

[](https://www.guru99.com/images/1/typesOfTesting.jpg)

1. **Acceptance Testing:**Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is usually performed by the customer. Read More on [Acceptance Testing](https://www.guru99.com/user-acceptance-testing.html)
2. **Accessibility Testing:**Type of testing which determines the usability of a product to the people having disabilities (deaf, blind, mentally disabled etc). The evaluation process is conducted by persons having disabilities. Read More on [Accessibility Testing](https://www.guru99.com/accessibility-testing.html)
3. **Active Testing:** Type of testing consisting in introducing test data and analyzing the execution results. It is usually conducted by the testing team.
4. **Agile Testing:**Software testing practice that follows the principles of the agile manifesto, emphasizing testing from the perspective of customers who will utilize the system. It is usually performed by the QA teams. Read More on [Agile Testing](https://www.guru99.com/agile-scrum-extreme-testing.html)
5. **Age Testing:**Type of testing which evaluates a system's ability to perform in the future. The evaluation process is conducted by testing teams.
6. **Ad-hoc Testing:**Testing performed without planning and documentation - the tester tries to 'break' the system by randomly trying the system's functionality. It is performed by the testing team. Read More on [Ad-hoc Testing](https://www.guru99.com/adhoc-testing.html)
7. **Alpha Testing:**Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users. Read More on [Alpha Testing](https://www.guru99.com/alpha-testing.html)
8. **Assertion Testing:**Type of testing consisting in verifying if the conditions confirm the product requirements. It is performed by the testing team.
9. **API Testing:**Testing technique similar to Unit Testing in that it targets the code level. Api Testing differs from Unit Testing in that it is typically a QA task and not a developer task. Read More on [API Testing](https://www.guru99.com/api-testing.html)
10. **All-pairs Testing:**Combinatorial testing method that tests all possible discrete combinations of input parameters. It is performed by the testing teams.
11. **Automated Testing:**Testing technique that uses Automation Testing tools to control the environment set-up, test execution and results reporting. It is performed by a computer and is used inside the testing teams. Read More on [Automated Testing](https://www.guru99.com/automation-testing.html)
12. **Basis Path Testing:**A testing mechanism which derives a logical complexity measure of a procedural design and use this as a guide for defining a basic set of execution paths. It is used by testing teams when defining test cases. Read More on [Basis Path Testing](https://www.guru99.com/basis-path-testing.html)
13. **Backward Compatibility Testing:**Testing method which verifies the behavior of the developed software with older versions of the test environment. It is performed by testing team.
14. **Beta Testing:**Final testing before releasing application for commercial purpose. It is typically done by end-users or others.
15. **Benchmark Testing:**Testing technique that uses representative sets of programs and data designed to evaluate the performance of computer hardware and software in a given configuration. It is performed by testing teams. Read More on [Benchmark Testing](https://www.guru99.com/benchmark-testing.html)
16. **Big Bang Integration Testing:**Testing technique which integrates individual program modules only when everything is ready. It is performed by the testing teams.
17. **Binary Portability Testing:**Technique that tests an executable application for portability across system platforms and environments, usually for conformation to an ABI specification. It is performed by the testing teams.
18. **Boundary Value Testing:**Software testing technique in which tests are designed to include representatives of boundary values. It is performed by the QA testing teams. Read More on [Boundary Value Testing](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html)
19. **Bottom Up Integration Testing:**In bottom-up Integration Testing, module at the lowest level are developed first and other modules which go towards the 'main' program are integrated and tested one at a time. It is usually performed by the testing teams.
20. **Branch Testing:**Testing technique in which all branches in the program source code are tested at least once. This is done by the developer.
21. **Breadth Testing:**A test suite that exercises the full functionality of a product but does not test features in detail. It is performed by testing teams.
22. **Black box Testing:**A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality. It is performed by QA teams. Read More on [Black box Testing](https://www.guru99.com/black-box-testing.html)
23. **Code-driven Testing:**Testing technique that uses testing frameworks (such as xUnit) that allow the execution of unit tests to determine whether various sections of the code are acting as expected under various circumstances. It is performed by the development teams.
24. **Compatibility Testing:**Testing technique that validates how well a software performs in a particular hardware/software/operating system/network environment. It is performed by the testing teams. Read More on [Compatibility Testing](https://www.guru99.com/compatibility-testing.html)
25. **Comparison Testing:**Testing technique which compares the product strengths and weaknesses with previous versions or other similar products. Can be performed by tester, developers, product managers or product owners. Read More on [Component Testing](https://www.guru99.com/component-testing.html)
26. **Component Testing:**Testing technique similar to unit testing but with a higher level of integration - testing is done in the context of the application instead of just directly testing a specific method. Can be performed by testing or development teams.
27. **Configuration Testing:**Testing technique which determines minimal and optimal configuration of hardware and software, and the effect of adding or modifying resources such as memory, disk drives and CPU. Usually it is performed by the Performance Testing engineers. Read More on [Configuration Testing](https://www.guru99.com/configuration-testing.html)
28. **Condition Coverage Testing:**Type of software testing where each condition is executed by making it true and false, in each of the ways at least once. It is typically made by the Automation Testing teams.
29. **Compliance Testing:**Type of testing which checks whether the system was developed in accordance with standards, procedures and guidelines. It is usually performed by external companies which offer "Certified OGC Compliant" brand.
30. **Concurrency Testing:**Multi-user testing geared towards determining the effects of accessing the same application code, module or database records. It it usually done by performance engineers. Read More on [Concurrency Testing](https://www.guru99.com/concurrency-testing.html)
31. **Conformance Testing:**The process of testing that an implementation conforms to the specification on which it is based. It is usually performed by testing teams. Read More on [Conformance Testing](https://www.guru99.com/conformance-testing.html)
32. **Context Driven Testing:**An Agile Testing technique that advocates continuous and creative evaluation of testing opportunities in light of the potential information revealed and the value of that information to the organization at a specific moment. It is usually performed by Agile testing teams.
33. **Conversion Testing:**Testing of programs or procedures used to convert data from existing systems for use in replacement systems. It is usually performed by the QA teams.
34. **Decision Coverage Testing:**Type of software testing where each condition/decision is executed by setting it on true/false. It is typically made by the automation testing teams.
35. **Destructive Testing:**Type of testing in which the tests are carried out to the specimen's failure, in order to understand a specimen's structural performance or material behavior under different loads. It is usually performed by QA teams. Read More on [Destructive Testing](https://www.guru99.com/destructive-testing.html)
36. **Dependency Testing:**Testing type which examines an application's requirements for pre-existing software, initial states and configuration in order to maintain proper functionality. It is usually performed by testing teams.
37. **Dynamic Testing:**Term used in software engineering to describe the testing of the dynamic behavior of code. It is typically performed by testing teams. Read More on [Dynamic Testing](https://www.guru99.com/dynamic-testing.html)
38. **Domain Testing:**White box testing technique which contains checkings that the program accepts only valid input. It is usually done by software development teams and occasionally by automation testing teams.
39. **Error-Handling Testing:**Software testing type which determines the ability of the system to properly process erroneous transactions. It is usually performed by the testing teams.
40. **End-to-end Testing:**Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate. It is performed by QA teams. Read More on [End-to-end Testing](https://www.guru99.com/end-to-end-testing.html)
41. **Endurance Testing:**Type of testing which checks for memory leaks or other problems that may occur with prolonged execution. It is usually performed by performance engineers. Read More on [Endurance Testing](https://www.guru99.com/endurance-testing.html)
42. **Exploratory Testing:**Black box testing technique performed without planning and documentation. It is usually performed by manual testers. Read More on [Exploratory Testing](https://www.guru99.com/exploratory-testing.html)
43. **Equivalence Partitioning Testing:**Software testing technique that divides the input data of a software unit into partitions of data from which test cases can be derived. it is usually performed by the QA teams. Read More on [Equivalence Partitioning Testing](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html)
44. **Fault injection Testing:**Element of a comprehensive test strategy that enables the tester to concentrate on the manner in which the application under test is able to handle exceptions. It is performed by QA teams.
45. **Formal verification Testing:**The act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics. It is usually performed by QA teams.
46. **Functional Testing:**Type of black box testing that bases its test cases on the specifications of the software component under test. It is performed by testing teams. Read More on [Functional Testing](https://www.guru99.com/functional-testing.html)
47. **Fuzz Testing:**Software testing technique that provides invalid, unexpected, or random data to the inputs of a program - a special area of mutation testing. Fuzz testing is performed by testing teams. Read More on [Fuzz Testing](https://www.guru99.com/fuzz-testing.html)
48. **Gorilla Testing:**Software testing technique which focuses on heavily testing of one particular module. It is performed by quality assurance teams, usually when running full testing.
49. **Gray Box Testing:**A combination of Black Box and White Box testing methodologies: testing a piece of software against its specification but using some knowledge of its internal workings. It can be performed by either development or testing teams.
50. **Glass box Testing:**Similar to white box testing, based on knowledge of the internal logic of an application's code. It is performed by development teams.
51. **GUI software Testing:**The process of testing a product that uses a graphical user interface, to ensure it meets its written specifications. This is normally done by the testing teams. Read More on [GUI software Testing](https://www.guru99.com/gui-testing.html)
52. **Globalization Testing:**Testing method that checks proper functionality of the product with any of the culture/locale settings using every type of international input possible. It is performed by the testing team. Read More on [Globalization Testing](https://www.guru99.com/globalization-vs-localization-testing.html)
53. **Hybrid Integration Testing:**Testing technique which combines top-down and bottom-up integration techniques in order leverage benefits of these kind of testing. It is usually performed by the testing teams.
54. **Integration Testing:**The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams. Read More on [Integration Testing](https://www.guru99.com/integration-testing.html)
55. **Interface Testing:**Testing conducted to evaluate whether systems or components pass data and control correctly to one another. It is usually performed by both testing and development teams. Read More on [Interface Testing](https://www.guru99.com/interface-testing.html)
56. **Install/uninstall Testing:**Quality assurance work that focuses on what customers will need to do to install and set up the new software successfully. It may involve full, partial or upgrades install/uninstall processes and is typically done by the software testing engineer in conjunction with the configuration manager.
57. **Internationalization Testing:**The process which ensures that product's functionality is not broken and all the messages are properly externalized when used in different languages and locale. It is usually performed by the testing teams.
58. **Inter-Systems Testing:**Testing technique that focuses on testing the application to ensure that interconnection between application functions correctly. It is usually done by the testing teams.
59. **Keyword-driven Testing:**Also known as table-driven testing or action-word testing, is a software testing methodology for automated testing that separates the test creation process into two distinct stages: a Planning Stage and an Implementation Stage. It can be used by either manual or automation testing teams. Read More on [Keyword-driven Testing](https://www.guru99.com/keyword-driven-testing.html)
60. **Load Testing:**Testing technique that puts demand on a system or device and measures its response. It is usually conducted by the performance engineers. Read More on [Load Testing](https://www.guru99.com/load-testing-tutorial.html)
61. **Localization Testing:**Part of software testing process focused on adapting a globalized application to a particular culture/locale. It is normally done by the testing teams. Read More on [Localization Testing](https://www.guru99.com/localization-testing.html)
62. **Loop Testing:**A white box testing technique that exercises program loops. It is performed by the development teams. Read More on [Loop Testing](https://www.guru99.com/loop-testing.html)
63. **Manual Scripted Testing:**Testing method in which the test cases are designed and reviewed by the team before executing it. It is done by Manual Testing teams.
64. **Manual-Support Testing:**Testing technique that involves testing of all the functions performed by the people while preparing the data and using these data from automated system. it is conducted by testing teams.
65. **Model-Based Testing:**The application of Model based design for designing and executing the necessary artifacts to perform software testing. It is usually performed by testing teams. Read More on [Model-Based Testing](https://www.guru99.com/model-based-testing-tutorial.html)
66. **Mutation Testing:**Method of software testing which involves modifying programs' source code or byte code in small ways in order to test sections of the code that are seldom or never accessed during normal tests execution. It is normally conducted by testers. Read More on [Mutation Testing](https://www.guru99.com/mutation-testing.html)
67. **Modularity-driven Testing:**Software testing technique which requires the creation of small, independent scripts that represent modules, sections, and functions of the application under test. It is usually performed by the testing team.
68. **Non-functional Testing:**Testing technique which focuses on testing of a software application for its non-functional requirements. Can be conducted by the performance engineers or by manual testing teams. Read More on [Non-functional Testing](https://www.guru99.com/non-functional-testing.html)
69. **Negative Testing:**Also known as "test to fail" - testing method where the tests' aim is showing that a component or system does not work. It is performed by manual or automation testers. Read More on [Negative Testing](https://www.guru99.com/negative-testing.html)
70. **Operational Testing:**Testing technique conducted to evaluate a system or component in its operational environment. Usually it is performed by testing teams. Read More on [Operational Testing](https://www.guru99.com/operational-testing.html)
71. **Orthogonal array Testing:**Systematic, statistical way of testing which can be applied in user interface testing, system testing, Regression Testing, configuration testing and Performance Testing. It is performed by the testing team. Read More on [Orthogonal array Testing](https://www.guru99.com/orthogonal-array-testing.html)
72. **Pair Testing:**Software development technique in which two team members work together at one keyboard to test the software application. One does the testing and the other analyzes or reviews the testing. This can be done between one Tester and Developer or Business Analyst or between two testers with both participants taking turns at driving the keyboard.
73. **Passive Testing:**Testing technique consisting in monitoring the results of a running system without introducing any special test data. It is performed by the testing team.
74. **Parallel Testing:**Testing technique which has the purpose to ensure that a new application which has replaced its older version has been installed and is running correctly. It is conducted by the testing team. Read More on [Parallel Testing](https://www.guru99.com/parallel-testing.html)
75. **Path Testing:**Typical white box testing which has the goal to satisfy coverage criteria for each logical path through the program. It is usually performed by the development team. Read More on [Path Testing](https://www.guru99.com/basis-path-testing.html)
76. **Penetration Testing:**Testing method which evaluates the security of a computer system or network by simulating an attack from a malicious source. Usually they are conducted by specialized penetration testing companies. Read More on [Penetration Testing](https://www.guru99.com/learn-penetration-testing.html)
77. **Performance Testing:**Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer. Read More on [Performance Testing](https://www.guru99.com/performance-testing.html)
78. **Qualification Testing:**Testing against the specifications of the previous release, usually conducted by the developer for the consumer, to demonstrate that the software meets its specified requirements.
79. **Ramp Testing:**Type of testing consisting in raising an input signal continuously until the system breaks down. It may be conducted by the testing team or the performance engineer.
80. **Regression Testing:** Type of software testing that seeks to uncover software errors after changes to the program (e.g. bug fixes or new functionality) have been made, by retesting the program. It is performed by the testing teams. Read More on [Regression Testing](https://www.guru99.com/regression-testing.html)
81. **Recovery Testing:**Testing technique which evaluates how well a system recovers from crashes, hardware failures, or other catastrophic problems. It is performed by the testing teams. Read More on [Recovery Testing](https://www.guru99.com/recovery-testing.html)
82. **Requirements Testing:**Testing technique which validates that the requirements are correct, complete, unambiguous, and logically consistent and allows designing a necessary and sufficient set of test cases from those requirements. It is performed by QA teams.
83. **Security Testing:**A process to determine that an information system protects data and maintains functionality as intended. It can be performed by testing teams or by specialized security-testing companies. Read More on [Security Testing](https://www.guru99.com/what-is-security-testing.html)
84. **Sanity Testing:**Testing technique which determines if a new software version is performing well enough to accept it for a major testing effort. It is performed by the testing teams. Read More on [Sanity Testing](https://www.guru99.com/smoke-sanity-testing.html)
85. **Scenario Testing:**Testing activity that uses scenarios based on a hypothetical story to help a person think through a complex problem or system for a testing environment. It is performed by the testing teams. Read More on [Scenario Testing](https://www.guru99.com/test-scenario.html)
86. **Scalability Testing:**Part of the battery of non-functional tests which tests a software application for measuring its capability to scale up - be it the user load supported, the number of transactions, the data volume etc. It is conducted by the performance engineer. Read More on [Scalability Testing](https://www.guru99.com/scalability-testing.html)
87. **Statement Testing:**White box testing which satisfies the criterion that each statement in a program is executed at least once during program testing. It is usually performed by the development team.
88. **Static Testing:**A form of software testing where the software isn't actually used it checks mainly for the sanity of the code, algorithm, or document. It is used by the developer who wrote the code. Read More on [Static Testing](https://www.guru99.com/static-dynamic-testing.html)
89. **Stability Testing:**Testing technique which attempts to determine if an application will crash. It is usually conducted by the performance engineer. Read More on [Stability Testing](https://www.guru99.com/stability-testing.html)
90. **Smoke Testing:**Testing technique which examines all the basic components of a software system to ensure that they work properly. Typically, smoke testing is conducted by the testing team, immediately after a software build is made. Read More on [Smoke Testing](https://www.guru99.com/smoke-sanity-testing.html)
91. **Storage Testing:**Testing type that verifies the program under test stores data files in the correct directories and that it reserves sufficient space to prevent unexpected termination resulting from lack of space. It is usually performed by the testing team. Read More on [Storage Testing](https://www.guru99.com/storage-testing.html)
92. **Stress Testing:**Testing technique which evaluates a system or component at or beyond the limits of its specified requirements. It is usually conducted by the performance engineer. Read More on [Stress Testing](https://www.guru99.com/stress-testing-tutorial.html)
93. **Structural Testing:**White box testing technique which takes into account the internal structure of a system or component and ensures that each program statement performs its intended function. It is usually performed by the software developers.
94. **System Testing:**The process of testing an integrated hardware and software system to verify that the system meets its specified requirements. It is conducted by the testing teams in both development and target environment. Read More on [System Testing](https://www.guru99.com/system-testing.html)
95. **System integration Testing:**Testing process that exercises a software system's coexistence with others. It is usually performed by the testing teams. Read More on [System integration Testing](https://www.guru99.com/system-integration-testing.html)
96. **Top Down Integration Testing:**Testing technique that involves starting at the top of a system hierarchy at the user interface and using stubs to test from the top down until the entire system has been implemented. It is conducted by the testing teams.
97. **Thread Testing:**A variation of top-down testing technique where the progressive integration of components follows the implementation of subsets of the requirements. It is usually performed by the testing teams. Read More on [Thread Testing](https://www.guru99.com/thread-testing.html)
98. **Upgrade Testing:**Testing technique that verifies if assets created with older versions can be used properly and that user's learning is not challenged. It is performed by the testing teams.
99. **Unit Testing:**Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team. Read More on [Unit Testing](https://www.guru99.com/unit-testing-guide.html)
100. **User Interface Testing:**Type of testing which is performed to check how user-friendly the application is. It is performed by testing teams. Read More on [User Interface Testing](https://www.guru99.com/interface-testing.html)

**Bonus !!! Its always good to know a few extra**

1. **Usability Testing:**Testing technique which verifies the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. It is usually performed by end users. Read More on [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html)
2. **Volume Testing:**Testing which confirms that any values that may become large over time (such as accumulated counts, logs, and data files), can be accommodated by the program and will not cause the program to stop working or degrade its operation in any manner. It is usually conducted by the performance engineer. Read More on [Volume Testing](https://www.guru99.com/volume-testing.html)
3. **Vulnerability Testing:**Type of testing which regards application security and has the purpose to prevent problems which may affect the application integrity and stability. It can be performed by the internal testing teams or outsourced to specialized companies. Read More on [Vulnerability Testing](https://www.guru99.com/vulnerability-assessment-testing-analysis.html)
4. **White box Testing:**Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers. Read More on [White box Testing](https://www.guru99.com/white-box-testing.html)
5. **Workflow Testing:**Scripted end-to-end testing technique which duplicates specific workflows which are expected to be utilized by the end-user. It is usually conducted by testing teams. Read More on [Workflow Testing](https://www.guru99.com/workflow-testing.html)

That concludes the list. Hope You enjoyed reading it.