**180+ Sample Test Cases for Testing Web and Desktop Applications – Comprehensive Testing Checklist**

This is a testing checklist for web and desktop applications.

Make testing checklist as an integral part of test cases writing process. Using this checklist you can easily create hundreds of [test cases](http://www.softwaretestinghelp.com/test-case-template-examples/" \o "sample test case template) for testing web or desktop applications. These are all general test cases and should be applicable for almost all kind of applications. Refer these tests while writing test cases for your project and I’m sure you will cover most [testing types](http://www.softwaretestinghelp.com/types-of-software-testing/" \o "testing types) except the application specific business rules provided in your SRS documents.

[](http://www.softwaretestinghelp.com/sample-test-cases-testing-web-desktop-applications/software-testing-checklist/)

Though this is a common checklist, I recommend preparing a standard testing checklist tailored to your specific needs using below test cases in addition with application specific tests.

**Importance of Using Checklist for Testing:**  
- Maintaining a standard repository of reusable test cases for your application will ensure the most common bugs will be caught more quickly.  
- Checklist helps to quickly complete writing test cases for new versions of the application.  
- Reusing test cases help to save money on resources to write repetitive tests.  
- Important test cases will be covered always making it almost impossible to forget.  
- Testing checklist can be referred by developers to ensure most common issues are fixed in development phase itself.

*Few notes to remember:*  
1) Execute these scenarios with different user roles e.g. admin user, guest user etc.  
2) For web applications these scenarios should be tested on [multiple browsers](http://www.softwaretestinghelp.com/best-cross-browser-testing-tools-to-ease-your-browser-compatibility-testing-efforts/" \o "Cross browser testing tools) like IE, FF, Chrome, and Safari with versions approved by client.  
3) Test with different screen resolutions like 1024 x 768, 1280 x 1024, etc.  
4) Application should be tested on variety of displays like LCD, CRT, Notebooks, Tablets, and Mobile phones.  
4) Test application on different platforms like Windows, Mac, Linux operating systems.

**Comprehensive Testing Checklist for Testing Web and Desktop Applications:**

**Assumptions:** Assuming that your application supports following functionality  
- Forms with various fields  
- Child windows  
- Application interacts with database  
- Various search filter criteria and display results  
- Image upload  
- Send email functionality  
- Data export functionality

**General Test Scenarios**

1. All mandatory fields should be validated and indicated by asterisk (\*) symbol  
2. Validation error messages should be displayed properly at correct position  
3. All error messages should be displayed in same CSS style (e.g. using red color)  
4. General confirmation messages should be displayed using CSS style other than error messages style (e.g. using green color)  
5. Tool tips text should be meaningful  
6. Dropdown fields should have first entry as blank or text like ‘Select’  
7. Delete functionality for any record on page should ask for confirmation  
8. Select/deselect all records options should be provided if page supports record add/delete/update functionality  
9. Amount values should be displayed with correct currency symbols  
10. Default page sorting should be provided  
11. Reset button functionality should set default values for all fields  
12. All numeric values should be formatted properly  
13. Input fields should be checked for max field value. Input values greater than specified max limit should not be accepted or stored in database  
14. Check all input fields for special characters  
15. Field labels should be standard e.g. field accepting user’s first name should be labeled properly as ‘First Name’  
16. Check page sorting functionality after add/edit/delete operations on any record  
17. Check for timeout functionality. Timeout values should be configurable. Check application behavior after operation timeout  
18. Check cookies used in an application  
19. Check if downloadable files are pointing to correct file paths  
20. All resource keys should be configurable in config files or database instead of hard coding  
21. Standard conventions should be followed throughout for naming resource keys  
22. Validate markup for all web pages (validate HTML and CSS for syntax errors) to make sure it is compliant with the standards  
23. Application crash or unavailable pages should be redirected to error page  
24. Check text on all pages for spelling and grammatical errors  
25. Check numeric input fields with character input values. Proper validation message should appear  
26. Check for negative numbers if allowed for numeric fields  
27. Check amount fields with decimal number values  
28. Check functionality of buttons available on all pages  
29. User should not be able to submit page twice by pressing submit button in quick succession.  
30. Divide by zero errors should be handled for any calculations  
31. Input data with first and last position blank should be handled correctly

**[GUI](http://www.softwaretestinghelp.com/gui-testing-on-smart-devices-%e2%80%93-testing-guidelines/" \o "GUI testing on smart devices) and Usability Test Scenarios**

1. All fields on page (e.g. text box, radio options, dropdown lists) should be aligned properly  
2. Numeric values should be right justified unless specified otherwise  
3. Enough space should be provided between field labels, columns, rows, error messages etc.  
4. Scroll bar should be enabled only when necessary  
5. Font size, style and color for headline, description text, labels, infield data, and grid info should be standard as specified in SRS  
6. Description text box should be multi-line  
7. Disabled fields should be grayed out and user should not be able to set focus on these fields  
8. Upon click of any input text field, mouse arrow pointer should get changed to cursor  
9. User should not be able to type in drop down select lists  
10. Information filled by users should remain intact when there is error message on page submit. User should be able to submit the form again by correcting the errors  
11. Check if proper field labels are used in error messages  
12. Dropdown field values should be displayed in defined sort order  
13. Tab and Shift+Tab order should work properly  
14. Default radio options should be pre-selected on page load  
15. Field specific and page level help messages should be available  
16. Check if correct fields are highlighted in case of errors  
17. Check if dropdown list options are readable and not truncated due to field size limit  
18. All buttons on page should be accessible by keyboard shortcuts and user should be able to perform all operations using keyboard  
19. Check all pages for broken images  
20. Check all pages for broken links  
21. All pages should have title  
22. Confirmation messages should be displayed before performing any update or delete operation  
23. Hour glass should be displayed when application is busy  
24. Page text should be left justified  
25. User should be able to select only one radio option and any combination for check boxes.

**Test Scenarios for Filter Criteria**

1. User should be able to filter results using all parameters on the page  
2. Refine search functionality should load search page with all user selected search parameters  
3. When there is at least one filter criteria is required to perform search operation, make sure proper error message is displayed when user submits the page without selecting any filter criteria.  
4. When at least one filter criteria selection is not compulsory user should be able to submit page and default search criteria should get used to query results  
5. Proper validation messages should be displayed for invalid values for filter criteria

**Test Scenarios for Result Grid**

1. Page loading symbol should be displayed when it’s taking more than default time to load the result page  
2. Check if all search parameters are used to fetch data shown on result grid  
3. Total number of results should be displayed on result grid  
4. Search criteria used for searching should be displayed on result grid  
5. Result grid values should be sorted by default column.  
6. Sorted columns should be displayed with sorting icon  
7. Result grids should include all specified columns with correct values  
8. Ascending and descending sorting functionality should work for columns supported with data sorting  
9. Result grids should be displayed with proper column and row spacing  
10. Pagination should be enabled when there are more results than the default result count per page  
11. Check for Next, Previous, First and Last page pagination functionality  
12. Duplicate records should not be displayed in result grid  
13. Check if all columns are visible and horizontal scroll bar is enabled if necessary  
14. Check data for dynamic columns (columns whose values are calculated dynamically based on the other column values)  
15. For result grids showing reports check ‘Totals’ row and verify total for every column  
16. For result grids showing reports check ‘Totals’ row data when pagination is enabled and user navigates to next page  
17. Check if proper symbols are used for displaying column values e.g. % symbol should be displayed for percentage calculation  
18. Check result grid data if date range is enabled

**Test Scenarios for a Window**

1. Check if default window size is correct  
2. Check if child window size is correct  
3. Check if there is any field on page with default focus (in general, the focus should be set on first input field of the screen)  
4. Check if child windows are getting closed on closing parent/opener window  
5. If child window is opened, user should not be able to use or update any field on background or parent window  
6. Check window minimize, maximize and close functionality  
7. Check if window is re-sizable  
8. Check scroll bar functionality for parent and child windows  
9. Check cancel button functionality for child window

**[Database Testing](http://www.softwaretestinghelp.com/database-testing-%e2%80%93-practical-tips-and-insight-on-how-to-test-database/" \o "Database testing tips ) Test Scenarios**

1. Check if correct data is getting saved in database upon successful page submit  
2. Check values for columns which are not accepting null values  
3. Check for data integrity. Data should be stored in single or multiple tables based on design  
4. Index names should be given as per the standards e.g. IND\_<Tablename>\_<ColumnName>  
5. Tables should have primary key column  
6. Table columns should have description information available (except for audit columns like created date, created by etc.)  
7. For every database add/update operation log should be added  
8. Required table indexes should be created  
9. Check if data is committed to database only when the operation is successfully completed  
10. Data should be rolled back in case of failed transactions  
11. Database name should be given as per the application type i.e. test, UAT, sandbox, live (though this is not a standard it is helpful for database maintenance)  
12. Database logical names should be given according to database name (again this is not standard but helpful for DB maintenance)  
13. Stored procedures should not be named with prefix “sp\_”  
14. Check is values for table audit columns (like createddate, createdby, updatedate, updatedby, isdeleted, deleteddate, deletedby etc.) are populated properly  
15. Check if input data is not truncated while saving. Field length shown to user on page and in database schema should be same  
16. Check numeric fields with minimum, maximum, and float values  
17. Check numeric fields with negative values (for both acceptance and non-acceptance)  
18. Check if radio button and dropdown list options are saved correctly in database  
19. Check if database fields are designed with correct data type and data length  
20. Check if all table constraints like Primary key, Foreign key etc. are implemented correctly  
21. Test stored procedures and triggers with sample input data  
22. Input field leading and trailing spaces should be truncated before committing data to database  
23. Null values should not be allowed for Primary key column

**Test Scenarios for Image Upload Functionality**

*(Also applicable for other file upload functionality)*  
1. Check for uploaded image path  
2. Check image upload and change functionality  
3. Check image upload functionality with image files of different extensions (e.g. JPEG, PNG, BMP etc.)  
4. Check image upload functionality with images having space or any other allowed special character in file name  
5. Check duplicate name image upload  
6. Check image upload with image size greater than the max allowed size. Proper error message should be displayed.  
7. Check image upload functionality with file types other than images (e.g. txt, doc, pdf, exe etc.). Proper error message should be displayed  
8. Check if images of specified height and width (if defined) are accepted otherwise rejected  
9. Image upload progress bar should appear for large size images  
10. Check if cancel button functionality is working in between upload process  
11. Check if file selection dialog shows only supported files listed  
12. Check multiple images upload functionality  
13. Check image quality after upload. Image quality should not be changed after upload  
14. Check if user is able to use/view the uploaded images

**Test Scenarios for Sending Emails**

*(Test cases for composing or validating emails are not included)*  
*(Make sure to use dummy email addresses before executing email related tests)*  
1. Email template should use standard CSS for all emails  
2. Email addresses should be validated before sending emails  
3. Special characters in email body template should be handled properly  
4. Language specific characters (e.g. Russian, Chinese or German language characters) should be handled properly in email body template  
5. Email subject should not be blank  
6. Placeholder fields used in email template should be replaced with actual values e.g. {Firstname} {Lastname} should be replaced with individuals first and last name properly for all recipients  
7. If reports with dynamic values are included in email body, report data should be calculated correctly  
8. Email sender name should not be blank  
9. Emails should be checked in different email clients like Outlook, Gmail, Hotmail, Yahoo! mail etc.  
10. Check send email functionality using TO, CC and BCC fields  
11. Check plain text emails  
12. Check HTML format emails  
13. Check email header and footer for company logo, privacy policy and other links  
14. Check emails with attachments  
15. Check send email functionality to single, multiple or distribution list recipients  
16. Check if reply to email address is correct  
17. Check sending high volume of emails

**Test Scenarios for Excel Export Functionality**

1. File should get exported in proper file extension  
2. File name for the exported Excel file should be as per the standards e.g. if file name is using timestamp, it should get replaced properly with actual timestamp at the time of exporting the file  
3. Check for date format if exported Excel file contains date columns  
4. Check number formatting for numeric or currency values. Formatting should be same as shown on page  
5. Exported file should have columns with proper column names  
6. Default page sorting should be carried in exported file as well  
7. Excel file data should be formatted properly with header and footer text, date, page numbers etc. values for all pages  
8. Check if data displayed on page and exported Excel file is same  
9. Check export functionality when pagination is enabled  
10. Check if export button is showing proper icon according to exported file type e.g. Excel file icon for xls files  
11. Check export functionality for files with very large size  
12. Check export functionality for pages containing special characters. Check if these special characters are exported properly in Excel file

**Performance Testing Test Scenarios**

1. Check if page load time is within acceptable range  
2. Check page load on slow connections  
3. Check response time for any action under light, normal, moderate and heavy load conditions  
4. Check performance of database stored procedures and triggers  
5. Check database query execution time  
6. Check for load testing of application  
7. Check for stress testing of application  
8. Check CPU and memory usage under peak load condition

**[Security Testing](http://www.softwaretestinghelp.com/security-testing-of-web-applications/" \o "Securtiy testing of web applications) Test Scenarios**

1. Check for SQL injection attacks  
2. Secure pages should use HTTPS protocol  
3. Page crash should not reveal application or server info. Error page should be displayed for this  
4. Escape special characters in input  
5. Error messages should not reveal any sensitive information  
6. All credentials should be transferred over an encrypted channel  
7. Test password security and password policy enforcement  
8. Check application logout functionality  
9. Check for Brute Force Attacks  
10. Cookie information should be stored in encrypted format only  
11. Check session cookie duration and session termination after timeout or logout  
11. Session tokens should be transmitted over secured channel  
13. Password should not be stored in cookies  
14. Test for Denial of Service attacks  
15. Test for memory leakage  
16. Test unauthorized application access by manipulating variable values in browser address bar  
17. Test file extension handing so that exe files are not uploaded and executed on server  
18. Sensitive fields like passwords and credit card information should not have auto complete enabled  
19. File upload functionality should use file type restrictions and also anti-virus for scanning uploaded files  
20. Check if directory listing is prohibited  
21. Password and other sensitive fields should be masked while typing  
22. Check if forgot password functionality is secured with features like temporary password expiry after specified hours and security question is asked before changing or requesting new password  
23. Verify CAPTCHA functionality  
24. Check if important events are logged in log files  
25. Check if access privileges are implemented correctly

**What is Penetration Testing?**  
It’s the process to identify security vulnerabilities in an application by evaluating the system or network with various malicious techniques. Purpose of this test is to secure important data from outsiders like hackers who can have unauthorized access to system. Once vulnerability is identified it is used to exploit system in order to gain access to sensitive information.

**Causes of vulnerabilities:**  
- Design and development errors  
- Poor system configuration  
- Human errors

**Why Penetration testing?**

- Financial data must be secured while transferring between different systems  
- Many clients are asking for pen testing as part of the software release cycle  
- To secure user data  
- To find security vulnerabilities in an application

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2012/06/Penetration-testing.jpg)

It’s very important for any organization to identify security issues present in internal network and computers. Using this information organization can plan defense against any hacking attempt. User privacy and data security are the biggest concerns nowadays. Imagine if any hacker manage to get user details of social networking site like Facebook. Organization can face legal issues due to a small loophole left in a software system. Hence big organizations are looking for PCI compliance certifications before doing any business with third party clients.

What should be tested?  
- Software  
- Hardware  
- Network  
- Process

**Penetration Testing Types:**

**1) Social Engineering:** Human errors are the main causes of security vulnerability. Security standards and policies should be followed by all staff members to avoid social engineering penetration attempt. Example of these standards include not to mention any sensitive information in email or phone communication. Security audits can be conducted to identify and correct process flaws.

**2) Application Security Testing:** Using software methods one can verify if the system is exposed to security vulnerabilities.

**3) Physical Penetration Test:** Strong physical security methods are applied to protect sensitive data. This is generally useful in military and government facilities. All physical network devices and access points are tested for possibilities of any security breach.

Pen Testing Techniques:  
1) Manual penetration test  
2) Using automated penetration test tools  
3) Combination of both manual and automated process  
The third process is more common to identify all kinds of vulnerabilities.

**Penetration Testing Tools:**

Automated tools can be used to identify some standard vulnerability present in an application. Pentest tools scan code to check if there is malicious code present which can lead to potential security breach. Pentest tools can verify security loopholes present in the system like data encryption techniques and hard coded values like username and password.

**Criteria to select the best penetration tool:**  
- It should be easy to deploy, configure and use.  
- It should scan your system easily.  
- It should categorize vulnerabilities based on severity that needs immediate fix.  
- It should be able to automate verification of vulnerabilities.  
- It should re-verify exploits found previously.  
- It should generate detailed vulnerability reports and logs.

Once you know what tests you need to perform you can either train your internal test resources or hire expert consultants to do the penetration task for you.

**Examples of Free and Commercial Tools** -  
[Nmap](http://nmap.org/), [Nessus](http://www.nessus.org/), [Metasploit](http://www.metasploit.com/), [Wireshark](http://www.wireshark.org/), [OpenSSL](http://www.openssl.org/), [Cain & Abel](http://www.oxid.it/cain.html), [THC Hydra](http://www.thc.org/thc-hydra/), [w3af](http://w3af.sourceforge.net/)  
Commercial services: [Pure Hacking](http://www.purehacking.com/), [Torrid Networks](http://www.torridnetworks.com/), [SecPoint](http://www.secpoint.com/), [Veracode](http://www.veracode.com/%20).

Limitations of Pentest tools: Sometimes these tools can flag false positive output which results in spending more developer time on analyzing such vulnerabilities which are not present.

**Manual Penetration Test:**

It’s difficult to find all vulnerabilities using automated tools. There are some vulnerabilities which can be identified by manual scan only. Penetration testers can perform better attacks on application based on their skills and knowledge of system being penetrated. The methods like social engineering can be done by humans only. Manual checking includes design, business logic as well as code verification.

Penetration Test Process:  
Let’s discuss the actual process followed by test agencies or penetration testers. Identifying vulnerabilities present in system is the first important step in this process. Corrective action is taken on these vulnerability and same penetration tests are repeated until system is negative to all those tests.

**We can categorize this process in following methods:**  
**1) Data collection:** Various methods including Google search are used to get target system data. One can also use web page source code analysis technique to get more info about the system, software and plugin versions. There are many free tools and services available in the market which can give you information like database or table names, DB versions, software versions, hardware used and various third party plugins used in the target system.

**2) Vulnerability Assessment:** Based on the data collected in first step one can find the security weakness in the target system. This helps penetration testers to launch attacks using identified entry points in the system.

**3) Actual Exploit:** This is crucial step. It requires special skills and techniques to launch attack on target system. Experienced penetration testers can use their skills to launch attack on the system.

**4) Result analysis and report preparation:** After completion of penetration tests detailed reports are prepared for taking corrective actions. All identified vulnerabilities and recommended corrective methods are listed in these reports. You can customize vulnerability report format (HTML, XML, MS Word or PDF) as per your organization needs.

**Penetration testing sample test cases (test scenarios):**

Remember this is not functional testing. In Pentest your goal is to find security holes in the system. Below are some generic test cases and not necessarily applicable for all applications.

**1)** Check if web application is able to identify spam attacks on contact forms used in the website.  
**2)** Proxy server – Check if network traffic is monitored by proxy appliances. Proxy server make it difficult for hackers to get internal details of the network thus protecting the system from external attacks.  
**3)** Spam email filters – Verify if incoming and outgoing email traffic is filtered and unsolicited emails are blocked. Many email clients come with in-build spam filters which needs to be configured as per your needs. These configuration rules can be applied on email headers, subject or body.  
**4)** Firewall – Make sure entire network or computers are protected with Firewall. Firewall can be a software or hardware to block unauthorized access to system. Firewall can prevent sending data outside the network without your permission.  
**5)** Try to exploit all servers, desktop systems, printers and network devices.  
**6)** Verify that all usernames and passwords are encrypted and transferred over secured connection like https.  
**7)** Verify information stored in [website cookies](http://www.softwaretestinghelp.com/website-cookie-testing-test-cases/" \o "Cookie testing). It should not be in readable format.  
**8 )** Verify previously found vulnerabilities to check if the fix is working.  
**9)** Verify if there is no open port in network.  
**11)** Verify all telephone devices.  
**12)** Verify WIFI network security.  
**13)** Verify all HTTP methods. PUT and Delete methods should not be enabled on web server .  
**14)** Password should be at least 8 character long containing at least one number and one special character.  
**15)** Username should not be like “admin” or “administrator”.  
**16)** Application login page should be locked upon few unsuccessful login attempts.  
**17)** Error messages should be generic and should not mention specific error details like “Invalid username” or “Invalid password”.  
**19)** Verify if special characters, html tags and scripts are handled properly as an input value.  
**20)** Internal system details should not be revealed in any of the error or alert messages.  
**21)** Custom error messages should be displayed to end user in case of web page crash.  
**22)** Verify use of registry entries. Sensitive information should not be kept in registry.  
**23)** All files must be scanned before uploading to server.  
**24)** Sensitive data should not be passed in urls while communicating with different internal modules of the web application.  
**25)** There should not be any hard coded username or password in the system.  
**26)** Verify all input fields with long input string with and without spaces.  
**27)** Verify if reset password functionality is secure.  
**28)** Verify application for [SQL Injection](http://www.softwaretestinghelp.com/sql-injection-%E2%80%93-how-to-test-application-for-sql-injection-attacks/" \o "SQL injection).  
**29)** Verify application for [Cross Site Scripting](http://www.softwaretestinghelp.com/security-testing-of-web-applications/" \o "Cross site scripting).  
**31)** Important input validations should be done at server side instead of JavaScript checks at client side.  
**32)** Critical resources in the system should be available to authorized persons and services only.  
**33)** All access logs should be maintained with proper access permissions.  
**34)** Verify user session ends upon log off.  
**35)** Verify that directory browsing is disabled on server.  
**36)** Verify that all applications and database versions are up to date.  
**37)** Verify url manipulation to check if web application is not showing any unwanted information.  
**38)** Verify memory leak and buffer overflow.  
**39)** Verify if incoming network traffic is scanned to find Trojan attacks.  
**40)** Verify if system is safe from Brute Force Attacks – a trial and error method to find sensitive information like passwords.  
**41)** Verify if system or network is secured from DoS (denial-of-service) attacks. Hacker can target network or single computer with continuous requests due to which resources on target system gets overloaded resulting in denial of service for legit requests.