* Execute these scenarios **with different user roles** e.g., admin users, guest users, etc.
* For web applications, these scenarios [should be tested on **multiple browsers**](https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/) like IE, FF, Chrome, and Safari with versions approved by the client.
* Test with different screen resolutions like 1024 x 768, 1280 x 1024, etc.
* An application should be tested on a **variety of displays** like LCD, CRT, Notebooks, Tablets, and Mobile phones.
* Test applications **on different platforms** like Windows, Mac, Linux operating systems etc.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2013/01/Software-Testing-Checklist.gif)

**180+ Web Application Testing Example Test Cases**

**Assumptions:** Assume that your application supports the following functionalities:

* Forms with various fields
* Child windows
* The application interacts with the 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 an **asterisk (\*) symbol**.  
2. Validation error messages should be displayed properly and in the correct position.  
3. All error messages should be displayed in the **same CSS style** (**For Example,** using red color)  
4. General confirmation messages should be displayed using CSS style other than error message style (**For Example,** using green color)  
5. Tooltips text should be meaningful.  
6. Drop-down fields should have the first entry as blank or text like “Select”.  
7. ‘Delete functionality’ for any record on the page should ask for a confirmation.  
8. Select/deselect all records option should be provided if page supports record add/delete/update functionality  
9. Amount values should be displayed with the 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 the max field value. Input values greater than the specified max limit should not be accepted or stored in the database.  
14. Check all input fields for **special characters**.  
15. **Field labels** should be standard e.g., the field accepting the 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 the operation timeout.  
18. Check the **cookies** used in the application.  
19. Check if the downloadable files are pointing to the **correct file path.**  
20. All resource keys should be configurable in config files or databases instead of hard coding.  
21. Standard conventions should be followed throughout for naming resource keys. ??  
22. Validate markups for all web pages (validate HTML and CSS for syntax errors) to make sure they are compliant with the standards.  
23. Application crashes or unavailable pages should be redirected to the **error page**.  
24. Check the text on all pages for spelling and grammatical errors.  
25. Check numeric input fields with character input values. A proper validation message should appear.  
26. Check for negative numbers if allowed for numeric fields.  
27. Check the number of fields with **decimal number** values.  
28. Check the **functionality of buttons** available on all pages.  
29. The user should **not** be able to **submit a page twice** by pressing the submit button in quick succession.  
30. Divide by zero errors should be handled for any calculations.  
31. Input data with the first and last position blank should be handled correctly.

**GUI and Usability Test Scenarios**

1**. All fields** on the page **(**For Example, text box, radio options, drop-down lists) should be **aligned properly.**  
2. Numeric values should be justified correctly unless specified otherwise.  
3. **Enough space** should be provided between field labels, columns, rows, error messages, etc.  
4. The **scrollbar** 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. The description text box should be **multi-lined.**  
7. Disabled fields should be greyed out and users should not be able to set focus on these fields.???  
8. Upon clicking on the input text field, the mouse arrow pointer should get **changed to the cursor.**9. The user should **not** be able to type in the drop-down select list.  
10. **Information filled** out by users should **remain intact** when there is an error message on the page submitted. The user should be able to submit the form again by correcting the errors.  
11. Check if proper field labels are being used in error messages.  
12. Drop-down field values should be displayed in **defined sort** order.  
13. Tab and Shift+Tab order should work properly.  
15. Field-specific and page-level **help messages** should be available.  
16. Check if the correct fields are highlighted in case of errors.  
17. Check if the drop-down list options are readable and not truncated due to field size limits.  
18. All buttons on the page should be accessible with keyboard shortcuts and the user should be able to perform all operations using a keyboard.  
19. Check all pages for broken images.  
20. Check all pages for broken links.  
21. All pages should have a title.  
22. Confirmation messages should be displayed before performing any updates or deleting operations.  
23. Hourglass should be displayed when the application is busy.  
24. Page text should be left-justified.  
25. The user should be able to select only one radio option and any combination for checkboxes.

Test Scenarios for Filter Criteria

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

Test Scenarios for Result Grid

1. The page loading symbol should be displayed when it’s taking longer than the default time to load the results page.  
2. Check if all the search parameters are used to fetch data shown on the result grid.  
3. The total number of results should be displayed in the result grid.  
4. Search criteria used for searching should be displayed in the result grid.  
5. Result grid values should be sorted by the default column.  
6. Sorted columns should be displayed with a sort icon.  
7. Result grids should include all the specified columns with the correct values.  
8. Ascending and descending sorting functionality should work for columns supported by 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 the results grid.  
13. Check if all the columns are visible and a horizontal scrollbar is enabled if necessary.  
14. Check the data for dynamic columns (columns whose values are calculated dynamically based on the other column values).  
15. For result grids showing reports, check the ‘Totals’ row and verify the total for every column.  
16. For result grids showing reports, check the ‘Totals’ row data when pagination is enabled and the user gets navigated to the 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 to see if the date range is enabled.

Test Scenarios for a Window

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

**Database Testing Test Scenarios**

1. Check if the correct data is getting saved in the database upon a successful page submit.  
2. Check values for columns that are not accepting null values.  
3. Check for data integrity. Data should be stored in single or multiple tables based on the design.  
4. Index names should be given as per the standards e.g. IND\_<Tablename>\_<ColumnName>  
5. Tables should have a 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 logs should be added.  
8. Required table indexes should be created.  
9. Check if data is committed to the 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 the database name (again this is not standard but helpful for DB maintenance).  
13. Stored procedures should not be named with a prefix “sp\_”  
14. Check if values for table audit columns (like created date, created by, updated, updated by, is deleted, deleted data, deleted by, etc.) are populated properly.  
15. Check if input data is not truncated while saving. The field length shown to the user on the page and in the database schema should be the 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 the radio button and drop-down list options are saved correctly in the database.  
19. Check if the database fields are designed with the 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 the database.  
23. Null values should not be allowed for the Primary key column.

Test Scenarios for Image Upload Functionality

*(Also applicable for other file upload functionality)*

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

Test Scenarios for Sending Emails

*(Test cases for composing or validating emails are not included here)*  
*(Make sure to use dummy email addresses before executing email related tests)*

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

Test Scenarios for Excel Export Functionality

1. The file should get exported with the proper file extension.  
2. The file name for the exported Excel file should be as per the standards, **For Example,** if the file name is using the timestamp, it should get replaced properly with an actual timestamp at the time of exporting the file.  
3. Check for date format if the exported Excel file contains the date columns.  
4. Check the number formatting for numeric or currency values. Formatting should be the same as shown on the page.  
5. The exported file should have columns with proper column names.  
6. Default page sorting should be carried out in the 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 the data displayed on the page and exported Excel file is the same.  
9. Check export functionality when pagination is enabled.  
10. Check if the export button is showing the proper icon according to the exported file type, **For Example,** 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 the Excel file.

**Performance Testing Test Scenarios**

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

**Security Testing Test Scenarios**

1. Check for SQL injection attacks.  
2. Secure pages should use the HTTPS protocol.  
3. Page crash should not reveal application or server info. The error page should be displayed for this.  
4. Escape special characters in the input.  
5. Error messages should not reveal any sensitive information.  
6. All credentials should be transferred over to an encrypted channel.  
7. Test password security and password policy enforcement.  
8. Check the 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 a secured channel.  
13. The 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 the browser address bar.  
17. Test file extension handling so that exe files are not uploaded or executed on the server.  
18. Sensitive fields like passwords and credit card information should not have to be autocomplete 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. Passwords 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 questions are asked before changing or requesting a new password.  
23. Verify CAPTCHA functionality.  
24. Check if important events are logged in log files.  
25. Check if access privileges are implemented correctly.

**Penetration Testing test cases** – I’ve listed around 41 test cases for Penetration Testing on [this page](https://www.softwaretestinghelp.com/penetration-testing-guide/).

*I ‘d really like to thank****Devanshu Lavaniya****(Sr. QA Engineer working for I-link Infosoft) for helping me to prepare this comprehensive testing checklist.*

I’ve tried to cover almost all standard test scenarios for Web and Desktop application functionality. I still know that this is not a complete checklist. Testers on different projects have their own testing checklist based on their experience.

**Updated:**

**100+ Ready-To-Execute Test Cases (Checklists)**

You Can Use this list to test the most common components of AUT

**How do you test the most common components of your AUT effectively, every single time?**

*This article is a list of common validations on the most widely found elements of AUT – that are put together for the convenience of testers (especially in the agile environment where frequent short-term releases happen).*

Each AUT (Application Under Test) is unique and has a very specific business purpose. The individual aspects (modules) of the AUT cater to different operations/actions that are crucial to the success of the business that the AUT supports.

Though each AUT is designed differently, individual components/fields that we encounter on most pages/screens/applications are the same with more or less similar behavior.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2013/12/ready-test-cases.png)

**Some Common Components of AUT:**

* Save, Update, Delete, Reset, Cancel, OK – links/buttons- whose functionality is the label of the object indicates.
* Text box, dropdowns, checkboxes, radio buttons, date control fields – that work the same way every time.
* Data grids, impacted areas, etc. to facilitate reports.

The way these individual elements contribute to the overall functionality of the application might be different but the steps to validate them are always the same.

Let’s continue with the list of the most common validations for [Web or Desktop application](https://www.softwaretestinghelp.com/difference-between-desktop-client-server-and-web-testing/) pages/forms.

**Note**: The actual results, expected results, test data and other parameters that are typically a part of a test case are omitted for the sake of simplicity – A general checklist approach is employed.

**Purpose of this comprehensive checklist:**

The primary purpose of these checklists (or test cases) is to ensure maximum test coverage on field level validations without spending too much time, and at the same time not compromise the quality of testing them.

After all, confidence in a product can only be attained by testing every single element to the best extent possible.

**A Complete Checklist (Test Cases) For Most Common Components Of AUT**

***Note: You can use these checklists as they are in Microsoft Excel format (download provided at the end of the article). You can even track the test execution in the same file with pass/fail results and status.***

*This could be an all-in-one resource for QA teams to test and track the most common co*