Whenever you will be asked to write the test cases for the *‘Form with some controls’,* you need to follow the list of rules for writing test cases as mentioned below:

* Write a test case on each form object.
* Written test cases should be a combination of both [negative and positive test cases](https://www.softwaretestinghelp.com/positive-and-negative-test-scenarios/).
* Also, test cases should always be a combination of functional, performance, UI, usability, and compatibility test cases.

Login functionality-

1. Username – Characters or numbers or combination of both

* 7 – 10 characters in length

1. Password – Combination of characters, numbers and special character

Should contains atleast 1 upperase, 1 lowercase and 1 special characters

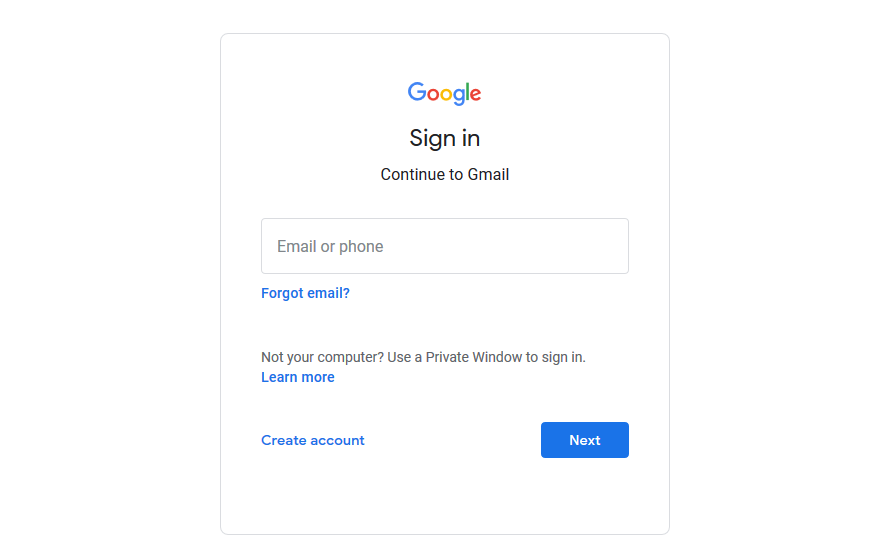
7 – 10 characters in length



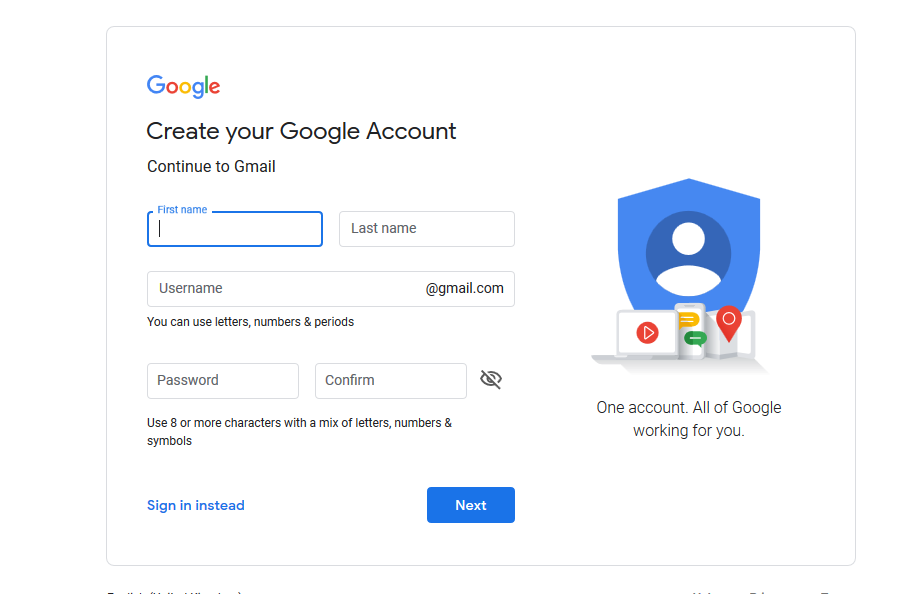
Wireframes f Login screen-



Gmail sign in



Sign up



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Test Cases – Login Page**

**Following is the possible list of functional and non-functional test cases for a login page:**

**Functional Test Cases:**

| **Sr. No.** | **Functional Test Cases** | **Type- Negative/ Positive Test Case** |
| --- | --- | --- |
| 1 | Verify if a user will be able to login with a valid username and valid password. | Positive |
| 2 | Verify if a user cannot login with a valid username and an invalid password. | Negative |
| 3 | Verify the login page for both, when the field is blank and Submit button is clicked. | Negative |
| 4 | Verify the ‘Forgot Password’ functionality. | Positive |
| 5 | Verify the messages for invalid login. | Positive |
| 6 | Verify the ‘Remember Me’ functionality. | Positive |
| 7 | Verify if the data in password field is either visible as asterisk or bullet signs. | Positive |
| 8 | Verify if a user is able to login with a new password only after he/she has changed the password. | Positive |
| 9 | Verify if the login page allows to log in simultaneously with different credentials in a different browser. | Positive |
| 10 | Verify if the ‘Enter’ key of the keyboard is working correctly on the login page. | Positive |
|  | Other Test Cases | |
| 11 | Verify the time taken to log in with a valid username and password. | Performance & Positive Testing |
| 12 | Verify if the font, text color, and color coding of the Login page is as per the standard. | UI Testing & Positive Testing |
| 13 | Verify if there is a ‘Cancel’ button available to erase the entered text. | Usability Testing |
| 14 | Verify the login page and all its controls in different browsers | Browser Compatibility & Positive Testing. |

**Non-functional Security Test Cases:**

| **Sr. No.** | **Security test cases** | **Type- Negative/ Positive Test Case** |
| --- | --- | --- |
| 1 | Verify if a user cannot enter the characters more than the specified range in each field (Username and Password). | Negative |
| 2 | Verify if a user cannot enter the characters more than the specified range in each field (Username and Password). | Positive |
| 3 | Verify the login page by pressing ‘Back button’ of the browser. It should not allow you to enter into the system once you log out. | Negative |
| 4 | Verify the timeout functionality of the login session. | Positive |
| 5 | Verify if a user should not be allowed to log in with different credentials from the same browser at the same time. | Negative |
| 6 | Verify if a user should be able to login with the same credentials in different browsers at the same time. | Positive |
| 7 | Verify the Login page against SQL injection attack. | Negative |
| 8 | Verify the implementation of SSL certificate. | Positive |
|  |  |  |

#### ****Test Cases for Gmail Login page****

| **Sr. No.** | **Test Scenarios** |
| --- | --- |
| 1 | Enter the valid email address & click next. Verify if the user gets an option to enter the password. |
| 2 | Don’t enter an email address or phone number & just click the Next button. Verify if the user will get the correct message or if the blank field will get highlighted. |
| 3 | Enter the invalid email address & click the Next button. Verify if the user will get the correct message. |
| 4 | Enter an invalid phone number & click the Next button. Verify if the user will get the correct message. |
| 5 | Verify if a user can log in with a valid email address and password. |
| 6 | Verify if a user can log in with a valid phone number and password. |
| 7 | Verify if a user cannot log in with a valid phone number and an invalid password. |
| 8 | Verify if a user cannot log in with a valid email address and a wrong password. |
| 9 | Verify the ‘Forgot email’ functionality. |
| 10 | Verify the ‘Forgot password’ functionality. |

#### Test Scenarios for the Sign-up page

**#1)** Verify the messages for each mandatory field.

**#2)** Verify if the user cannot proceed without filling all the mandatory fields.

**#3)** Verify the age of the user when the DOB is selected.

**#4)** Verify if the numbers and special characters are not allowed in the First and Last name.

**#5)** Verify if a user can sign-up successfully with all the mandatory details.

**#6)** Verify if a user can log in with the valid details.

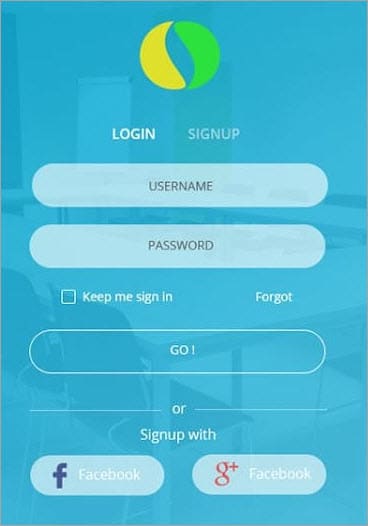
**#7)** Verify if the Password and Confirm Password fields are accepting similar strings only.

**#8)** Verify if the Password field will prompt you for the weak passwords.

**#9)** Verify if duplicate email address will not get assigned.

**#10)** Verify that hints are provided for each field on the form, for the ease of use.

#### ****Test Scenarios for the Login page of Mobile Application****



[image [*source]*](https://www.youtube.com/watch?v=BvDERgRI82g)

**#1)** Verify if a user can log in with a valid username and password.

**#2)** Verify if a user cannot log in with an invalid username or password. Check permutation and combinations of this.

**#3)** Verify the ‘Keep me Sign In’ option. If this check box is selected, then the user should not get logged out even after exiting the app.

**#4)** Verify if this check box is not selected by default.

**#5)** If the user has signed up with Facebook or social media, verify that the user can log in with those credentials or not.

**#6)** Verify the Forgot password functionality.

**#7)** Verify if the login page fits the mobile screen. The user should not have to scroll the screen.

**Conclusion**

While writing test cases for login or sign-up page write the test cases for all the fields. There should be a combination of both positive and negative test cases. Try to cover the performance, security, and functional scenarios.

The login page is the page with fewer controls, so even though it is looking simple for testing, it should not be considered as an easy task.

=========================================================================================================================================================================================================================================================================================================================================================================================================================================

**General Test Scenarios**

1. All mandatory fields should be validated and indicated by an asterisk (\*) symbol.  
2. Validation error messages should be displayed properly 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 messages 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 a 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 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. 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 the operation timeout.  
18. Check cookies used in an application.  
19. Check if downloadable files are pointing to the 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 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 a 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 click of an input text field, the mouse arrow pointer should get changed to the cursor.  
9. The user should not be able to type in drop-down select lists.  
10. Information filled by users should remain intact when there is an error message on page submit. The 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. Drop-down 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 the page load.  
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 a page should be accessible by 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 update or delete operation.  
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, make sure 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 get used to query results.  
5. Proper validation messages should be displayed for all 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 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 default column.  
6. Sorted columns should be displayed with a sort icon.  
7. Result grids should include all the specified columns with 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 result 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 ‘Totals' row and verify the total for every column.  
16. For result grids showing reports check ‘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 know 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 on closing 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 window minimize, maximize, and close functionality.  
7. Check if the window is re-sizable.  
8. Check scroll bar functionality for parent and child windows.  
9. Check cancel button functionality for the child window.

**Database Testing Test Scenarios**

1. Check if 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 log 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. Field length shown to the user on the page and in 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 the table constraints like a 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 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 having space or any other allowed special character in the file name.  
5. Check duplicate name image upload.  
6. Check image upload with image size greater than the max allowed size. The Proper error message 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 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 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 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. 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 the recipients.  
7. If reports with dynamic values are included in the email body and 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 to 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 to send email functionality to single, multiple or distribution list recipients.  
16. Check if a 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 in 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 exported Excel file contains the date columns.  
4. Check 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 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 a page and exported Excel file is the same.  
9. Check export functionality when pagination is enabled.  
10. Check if the export button is showing 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 the page load on slow connections.  
3. Check the response time for any action under a 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 the 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 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 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 handing so that exe files are not uploaded and executed on the server.  
18. Sensitive fields like passwords and credit card information should not have to 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 question is 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 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 to all applications.

1. Check if the web application is able to identify spam attacks on contact forms used on the website.
2. Proxy server – Check if network traffic is monitored by proxy appliances. The proxy server makes 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.
4. Many email clients come with inbuilt spam filters that need to be configured as per your needs. These configuration rules can be applied to email headers, subject or body.
5. Firewall – Make sure the entire network or computers are protected with firewalls. A Firewall can be software or hardware to block unauthorized access to a system. A Firewall can prevent sending data outside the network without your permission.
6. Try to exploit all servers, desktop systems, printers, and network devices.
7. Verify that all usernames and passwords are encrypted and transferred over secure connections like https.
8. Verify information stored in [website cookies](https://www.softwaretestinghelp.com/website-cookie-testing-test-cases/). It should not be in a readable format.
9. Verify previously found vulnerabilities to check if the fix is working.
10. Verify if there is no open port in the 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 a web server.
14. Verify if the password meets the required standards. The password should be at least 8 characters long containing at least one number and one special character.
15. Username should not be like “admin” or “administrator”.
16. The application login page should be locked upon a few unsuccessful login attempts.
17. Error messages should be generic and should not mention specific error details like “Invalid username” or “Invalid password”.
18. Verify if special characters, HTML tags, and scripts are handled properly as an input value.
19. Internal system details should not be revealed in any of the error or alert messages.
20. Custom error messages should be displayed to end-users in case of a web page crash.
21. Verify the use of registry entries. Sensitive information should not be kept in the registry.
22. All files must be scanned before uploading them to the server.
23. Sensitive data should not be passed in URLs while communicating with different internal modules of the web application.
24. There should not be any hardcoded username or password in the system.
25. Verify all input fields with long input string with and without spaces.
26. Verify if reset password functionality is secure.
27. Verify application for [SQL Injection](https://www.softwaretestinghelp.com/sql-injection-%E2%80%93-how-to-test-application-for-sql-injection-attacks/).
28. Verify application for [Cross-Site Scripting](https://www.softwaretestinghelp.com/security-testing-of-web-applications/).
29. Important input validations should be done at the server-side instead of JavaScript checks at the client-side.
30. Critical resources in the system should be available to authorized persons and services only.
31. All access logs should be maintained with proper access permissions.
32. Verify user session ends upon log off.
33. Verify that directory browsing is disabled on the server.
34. Verify that all applications and database versions are up to date.
35. Verify URL manipulation to check if a web application is not showing any unwanted information.
36. Verify memory leak and buffer overflow.
37. Verify if incoming network traffic is scanned to find Trojan attacks.
38. Verify if the system is safe from Brute Force Attacks – a trial and error method to find sensitive information like passwords.
39. Verify if the system or network is secured from DoS (denial-of-service) attacks. Hacker can target network or a single computer with continuous requests due to which resources on the target system gets overloaded resulting in the denial of service for legit requests.
40. Verify application for HTML script injection attacks.
41. Verify against COM & ActiveX attacks.
42. Verify against spoofing attacks. Spoofing can be of multiple types – IP address spoofing, Email ID spoofing,
43. ARP spoofing, Referrer spoofing, Caller ID spoofing, Poisoning of file-sharing networks, GPS spoofing.
44. Check for an uncontrolled format string attack – a security attack that can cause the application to crash or execute the harmful script on it.
45. Verify XML injection attack – used to alter the intended logic of the application.
46. Verify against canonicalization attacks.
47. Verify if the error pages are displaying any information that can be helpful for a hacker to enter into the system.
48. Verify if any critical data like the password is stored in secret files on the system.
49. Verify if the application is returning more data than it is required.

These are just the basic test scenarios to get started with Pentest. There are hundreds of advanced penetration methods which can be done either manually or with the help of automation tools.

**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 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 result, expected result, 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.

### The Complete Checklist (Test Cases) For Most Common Components Of AUT

**Note: You can use these checklists as it is 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 components of AUT.  You can add or update test cases specific to your application and make it an even more comprehensive list.

#### Checklist #1: Mobile Testing Checklist

|  |
| --- |
| Module Name: |
| Module Functionality: |
| Module Impact over the application: |
| Module Flow: |
| Menu & Submenu: |
| Spellings and Order & Suitability: |
| Control for each submenu: |

#### Checklist #2: Forms/Screens Testing Checklist

|  |
| --- |
| Form Functionality: |
| Form Impact over the application: |
| Form Flow: |
| Designing: |
| Alignments: |
| Title: |
| Field Names: |
| Spellings: |
| Mandatory Marks: |
| Alerts to Mandatory fields: |
| Buttons: |
| Default Cursor Position: |
| Tab Sequence: |
| The page before entering any data: |
| Page after entering data: |

#### Checklist #3: Textbox Field Testing Checklist

***Text Box:***

|  |  |  |
| --- | --- | --- |
|  | **ADD (In add screen)** | **EDIT (in Edit screen)** |
| Characters |  |  |
| Special Characters |  |  |
| Numbers |  |  |
| Limit |  |  |
| Alert |  |  |
| Spelling & Grammar in Alert message: |  |  |

***BVA (Size) for Text Box:***

Min —>—> Pass  
Min-1 —> —> Fail  
Min+1 —> —> Pass  
Max-1 —> —> Pass  
Max+1 —> —> Fail  
Max —> —> Pass

***ECP for Text Box:***

|  |  |
| --- | --- |
| ***Valid*** | ***In Valid*** |
| – | – |
| – | – |

#### Checklist #4: List-box or Drop-down List Testing Checklist

***List Box/Dropdown:***

|  |  |  |
| --- | --- | --- |
|  | **ADD (In add screen)** | **EDIT (in Edit screen)** |
| Header |  |  |
| The correctness of Existed Data |  |  |
| Order of Data |  |  |
| Selection and Deselection |  |  |
| Alert: |  |  |
| Spelling and Grammar of Alert message |  |  |
| Cursor after alert |  |  |
| Reflection of Selection and Deselection in remaining fields |  |  |

#### Checklist #5: Checkbox Field Testing Checklist

***CheckBox:***

|  |  |  |
| --- | --- | --- |
|  | **ADD (In add screen)** | **EDIT (in Edit screen)** |
| Default Selection |  |  |
| Action after selection |  |  |
| Action after de-selection |  |  |
| Selection and Deselection |  |  |
| Alert: |  |  |
| Spelling and Grammar of Alert message |  |  |
| Cursor after alert |  |  |
| Reflection of Selection and Deselection in remaining fields |  |  |

#### Checklist #6: Radio Button Testing Checklist

***Radio button:***

|  |  |  |
| --- | --- | --- |
|  | **ADD (In add screen)** | **EDIT (in Edit screen)** |
| Default Selection |  |  |
| Action after selection |  |  |
| Action after de-selection |  |  |
| Selection and Deselection |  |  |
| Alert: |  |  |
| Spelling and Grammar of Alert message |  |  |
| Cursor after alert |  |  |
| Reflection of Selection and Deselection in remaining fields |  |  |

#### Checklist #7: Date Field Test Scenarios

***Date field:***

|  |  |  |
| --- | --- | --- |
|  | **ADD (In add screen)** | **EDIT (in Edit screen)** |
| Default date display |  |  |
| Design of calendar |  |  |
| Navigation for different months and years in date control |  |  |
| Manual Entry in date text box |  |  |
| Date format and uniformity with the overall application |  |  |
| Alert: |  |  |
| Spelling and Grammar of Alert message |  |  |
| Cursor after alert |  |  |
| Reflection of Selection and Deselection in remaining fields |  |  |

#### Checklist #8: Save Button Testing Scenarios

***Save/update:***

|  |  |  |
| --- | --- | --- |
|  | **ADD (In add screen)** | **EDIT (in Edit screen)** |
| Without giving any data: |  |  |
| With only mandatory fields: |  |  |
| With All fields: |  |  |
| With Max limit: |  |  |
| With min limit |  |  |
| Spelling & Grammar in Confirmation  Alert message: |  |  |
| Cursor |  |  |
| Duplication of Unique fields: |  |  |
| Spelling & Grammar in duplication Alert message: |  |  |
| Cursor |  |  |

#### Checklist #9: Cancel Button Test Scenarios

***Cancel:***

|  |  |  |
| --- | --- | --- |
| With data in all fields |  |  |
| With only mandatory fields: |  |  |
| With all fields: |  |  |

#### Checklist #10: Delete Button Testing Points

***Delete:***

|  |  |
| --- | --- |
|  | **EDIT (in Edit screen)** |
| Delete the record which is not used anywhere in the application |  |
| Delete the record which has a dependency |  |
| Add the new record with same deleted details again |  |

#### Checklist #11: To Verify Impacted Areas after Save or Update

***After Saving/updating:***

|  |  |
| --- | --- |
| Display in View |  |
| Reflection in impacted forms in the application |  |

#### Checklist #12: Data Grid Testing List

***Data Grid:***

|  |  |
| --- | --- |
| Grid Title and spelling |  |
| Form Before giving any data |  |
| Message Before giving any data |  |
| Spellings |  |
| Alignments |  |
| S No |  |
| Field Names & Order |  |
| The correctness of Existed data |  |
| Order of Existed data |  |
| Alignment of Existed data |  |
| Page navigators |  |
| Data when navigating with different pages |  |

**Edit Link Functionality**

|  |  |
| --- | --- |
| Page after Edit: |  |
| Title and spellings |  |
| Existed data of the Selected record in each field |  |
| Buttons |  |

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Test Scenarios Login Page**

1. Verify that the login screen is having option to enter username and password with submit button and option of forgot password
2. Verify that user is able to login with valid username and password
3. Verify that user is not able to login with invalid username and password
4. Verify that validation message gets displayed in case user leaves username or password field as blank
5. Verify that validation message is displayed in case user exceeds the character limit of the user name and password fields
6. Verify that there is reset button to clear the field's text
7. Verify if there is checkbox with label "remember password" in the login page
8. Verify that the password is in encrypted form when entered
9. Verify that there is limit on the total number of unsuccessful attempts
10. For security point of view, in case of in correct credentials user is displayed the message like "incorrect username or password" instead of exact message pointing at the field that is incorrect. As message like "incorrect username" will aid hacker in bruteforcing the fields one by one
11. Verify the timeout of the login session
12. Verify if the password can be copy-pasted or not
13. Verify that once logged in, clicking back button doesn't logout user
14. Verify if SQL Injection attacks works on login page
15. Verify if XSS vulnerability work on login page

**Test scenarios for Registration Page**

1. Verify that all the specified fields are present on the registration page
2. Verify that the required/mandatory fields are marked with \* against the field
3. Verify that for better user interface dropdowns, radio buttons and checkboxes etc fields are displayed wherever possible instead of just textboxes
4. Verify the page has both submit and cancel/reset buttons at the end
5. Verify that clicking submit button after entering all the required fields, submits the data to the server
6. Verify that clicking cancel/reset button after entering all the required fields, cancels the submit request and resets all the fields
7. Verify that whenever possible validation should take place at client side
8. Verify that not filling the mandatory fields and clicking submit button will lead to validation error
9. Verify that not filling the optional fields and clicking submit button will still send data to server without any validation error
10. Check the upper limit of the textboxes
11. Check validation on date and email fields (only valid dates and valid email Ids should be allowed
12. Check validation on numeric fields by entering alphabets and special characters
13. Verify that leading and trailing spaces are trimmed
14. Verify that entering blank spaces on mandatory fields lead to validation error
15. Verify that after making a request to the server and then sending the same request again with the same unique key will lead to server side validation error

**Test Scenarios for Inbox Functionality(Receive Email)**

1. Verify that a newly received email is displayed as highlighted in the Inbox section.
2. Verify that a newly received email has correctly displayed sender emailId or name, mail subject and mail body(trimmed to single line).
3. Verify that on clicking the newly received email, user is navigated to email content.
4. Verify that the email contents are correctly displayed with the desired source formatting.
5. Verify that any attachments are attached to the email and is downloadable.
6. Verify that the attachments are scanned for viruses before download.
7. Verify that all the emails marked as read are not highlighted.
8. Verify that all the emails read as well as unread have a mail read time appended at the end on the email list displayed in the inbox section.
9. Verify that count of unread emails is displayed alongside 'Inbox' text in left sidebar of GMail.
10. Verify that unread email count increases by one on receiving a new email.
11. Verify that unread email count decreases by one on reading an email ( marking email as read).
12. Verify that email recipients in cc are visible to all user.
13. Verify that email recipients in bcc are not visible to user.
14. Verify that all received emails get piled up in the 'Inbox' section and gets deleted in cyclic fashion based on the size availability.
15. Verify that email can be received from non-gmail emailIds like - yahoo, hotmail etc.

**Test scenarios for Compose mail Functionality**

1. Verify that on clicking 'Compose' button, a frame to compose a mail gets displayed.
2. Verify that user can enter emailIds in 'To', 'cc' and 'bcc' sections and also user will get suggestions while typing the emailds based on the existing emailIds in user's email list.
3. Verify that user can enter multiple comma separated emailIds in 'To', 'cc' and 'bcc' sections.
4. Verify that user can type Subject line in the 'Subject' textbox.
5. Verify that user can type the email in email-body section.
6. Verify that user can format mail using editor-options provided like choosing font-family, font-size, bold-italic-underline etc.
7. Verify that user can user can attach file as an attachement to the email.
8. Verify that user can add images in the email and select the size for the same.
9. Verify that after entering emailIds in either of the 'To', 'cc' and 'bcc' sections, entering Subject line and mail body and clicking 'Send' button, mail gets delivered to intended receivers.
10. Verify that sent mails can be found in 'Sent Mail' sections of the sender.
11. Verify that mail can be sent to non-gmail emailIds also.
12. Verify that all sent emails get piled up in the 'Sent Mail' section and gets deleted in cyclic fashion based on the size availability.
13. Verify that the emails composed but not sent remain in the draft section.
14. Verify the maximum number of email recepients that can be entered in 'To', 'cc' and 'bcc' sections.
15. Verify the maximum length of text that can be entered in the 'Subject' textbox.
16. Verify the content limit of text/images that can be entered and successfully delivered as mail body.
17. Verify the maximum size and number of attachement that can be attached with an email.
18. Verify that only the allowed specifications of the attchement can be attached with an email/
19. Verify that if email is sent without Subject, a pop-up is generated warning user about no subject line. Also, verify that on accepting the pop-up message, user is able to send the email.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## General Test Cases

1. Verify that user is able to navigate through all the products across different categories
2. Verify that all the links and banners are redirecting to correct product/category pages and none of the links arr broken
3. Verify that the company logo is clearly visible
4. Verify that all the text – product, category name, price and product description are clearly visible
5. Verify that all the images – product and banner are clearly visible
6. Verify that category pages have relevant product listed specific to the category
7. Verify that correct count of total products are listed on the category pages
8. Search – Verify that on searching all the product satisfying the search criteria are visble on the search result page
9. Search – Verify the more relevant product for the search term are displayed on the top for a particular search term
10. Search – Verify that count of products is correctly displayed on the search result page for a particular search term
11. Filtering – Verify that filtering functionality correctly filters product based on the filter applied
12. Filtering – Verify that filtering works correctly on category pages
13. Filtering – Verify that filtering works correctly on the search result page
14. Filtering – Verify that correct count of total products is displayed after a filter is applied
15. Sorting – Verify that all the sort options work correctly – correctly sort the products based on the sort option chosen
16. Sorting – Verify that sorting works correctly on the category pages
17. Sorting – Verify that sorting works correctly on the search result page
18. Sorting – Verify that sorting works correctly on the pages containing filtered result, after applying filters
19. Sorting – Verify that product count remains intact irrespective of sorting option applied

## Product Buy Flow – Test cases

1. Verify that on the product page, user can select the desired attribute of the product e.g. size, color etc
2. Verify that user can add to cart one or more products
3. Verify that user can add products to wishlist
4. Verify that user can buy products added to cart after signing in to the application (or as per the functionality of the website)
5. Verify that user can successfully buy more than one products that were added to his/her cart
6. Verify that user cannot add more than available inventory of the product
7. Verify that the limit to the number of products a user can by is working correctly by displaying error message and preventing user from buying more than the limit
8. Verify that the delivery can be declined of for the places where shipping is not available
9. Verify that Cash on Delivery option of payment is working fine
10. Verify that the different pre-paid methods of payments are working fine
11. Verify that product return functionality works fine

## User(Buyer) Registration – Test cases

1. Verify that all the specified fields are present on the registration page
2. Verify that the required/mandatory fields are marked with \* against the field
3. Verify that for better user interface dropdowns, radio buttons and checkboxes etc fields are displayed wherever possible instead of just textboxes
4. Verify the page has both submit and cancel/reset buttons at the end
5. Verify that clicking submit button after entering all the required fields, submits the data to the server
6. Verify that clicking cancel/reset button after entering all the required fields, cancels the submit request and resets all the fields
7. Verify that whenever possible validation should take place at client side
8. Verify that not filling the mandatory fields and clicking submit button will lead to validation error
9. Verify that not filling the optional fields and clicking submit button will still send data to server without any validation error
10. Check the upper limit of the textboxes
11. Check validation on date and email fields (only valid dates and valid email Ids should be allowed
12. Check validation on numeric fields by entering alphabets and special characters
13. Verify that leading and trailing spaces are trimmed
14. Verify that entering blank spaces on mandatory fields lead to validation error
15. Verify that after making a request to the server and then sending the same request again with the same unique key will lead to server side validation error

## Seller – Product creation Test cases

1. Verify that authenticated sellers get access to product creation panel specific to the authorised categories
2. Verify that product creation panel is working fine for single product creation
3. Verify that product creation panel is working fine for multiple product creation
4. Verify that maximum product creation limit for seller is working fine, limiting seller to create more than the desired number of products
5. Verify panel validation for checking mandatory fields
6. Verify that duplicate product creation is restricted through panel
7. Verify that seller can update information and price of existing products
8. Verify that product created by seller get visible on the website after certain period of time
9. Verify that updation made by seller get visible on the website after certain period of time

## Test Scenarios Registration Form

Must Read: [Test Case Template With Detailed Explanation](https://www.softwaretestingmaterial.com/test-case-template-with-explanation/)

* User Name
* First Name
* Last Name
* Password
* Confirm Password
* Email Id
* Phone number
* Date of birth
* Gender
* Location
* Terms of use
* Submit
* Login (If you already have an account)

Test Scenarios of a Registration Form:

1. Verify that the Registration form contains Username, First Name, Last Name, Password, Confirm Password, Email Id, Phone number, Date of birth, Gender, Location, Terms of use, Submit, Login (If you already have an account)
2. Verify that tab functionality is working properly or not
3. Verify that Enter/Tab key works as a substitute for the Submit button
4. Verify that all the fields such as Username, First Name, Last Name, Password and other fields have a valid placeholder
5. Verify that the labels float upward when the text field is in focus or filled (In case of floating label)
6. Verify that all the required/mandatory fields are marked with \* against the field
7. Verify that clicking on submit button after entering all the mandatory fields, submits the data to the server
8. Verify that system generates a validation message when clicking on submit button without filling all the mandatory fields.
9. Verify that entering blank spaces on mandatory fields lead to validation error
10. Verify that clicking on submit button by leaving optional fields, submits the data to the server without any validation error
11. Verify that case sensitivity of Username (usually Username field should not follow case sensitivity – ‘rajkumar’ & ‘RAJKUMAR’ acts same)
12. Verify that system generates a validation message when entering existing username
13. Verify that the character limit in all the fields (mainly username and password) based on business requirement
14. Verify that the username validation as per business requirement (in some application, username should not allow numeric and special characters)
15. Verify that the validation of all the fields are as per business requirement
16. Verify that the date of birth field should not allow the dates greater than current date (some applications have age limit of 18 in that case you have to validate whether the age is greater than or equal to 18 or not)
17. Verify that the validation of email field by entering incorrect email id
18. Verify that the validation of numeric fields by entering alphabets and characters
19. Verify that leading and trailing spaces are trimmed after clicking on submit button
20. Verify that the “terms and conditions” checkbox is unselected by default (depends on business logic, it may be selected or unselected)
21. Verify that the validation message is displayed when clicking on submit button without selecting “terms and conditions” checkbox
22. Verify that the password is in encrypted form when entered
23. Verify whether the password and confirm password are same or not