A test case in applications development is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. When testing user interfaces, it is easy to overlook test cases that would be helpful for a more thoroughly tested solution. The following list identifies twenty test cases that should be considered when testing user interfaces.

**35 Useful Test Cases for Testing User Interfaces**

1. **Required Fields**
If the screen requires data entry on a specific field, designers should identify the required fields with a red asterisk and generate a friendly warning if the data is left blank.
2. **Data Type Errors**
If the screen contains dates, numeric, currency or other specific data types, ensure that only valid data can be entered.
3. **Field Widths**
If the screen contains text boxes that allow data entry, ensure that the width of data entered does not exceed the width of the table field (e.g. a title that is limited to 100 characters in the database should not allow more than 100 characters to be entered from the user interface).
4. **Onscreen Instructions**
Any screen that is not self-explanatory to the casual user should contain onscreen instructions that aid the user.
5. **Keep Onscreen Instructions Brief**
While onscreen instructions are great, keep the wording informative, in layman’s terms, but concise.
6. **Progress Bars**
If the screen takes more than 5 seconds to render results, it should contain a progress bar so that the user understands the processing is continuing.
7. **Same Document Opened Multiple Times**
If the application opens the same document multiple times, it should append a unique number to the open document to keep one document from overwriting another. For example, if the application opens a document named Minutes.txt and it opens the same document for the same user again, consider having it append the time to the document or sequentially number it (Minutes2.txt or Minutes\_032321.txt).
8. **Cosmetic Inconsistencies**
The screen look, feel, and design should match the other screens in your application. Creating and using a style guide is a great way to ensure consistency throughout your application.
9. **Abbreviation Inconsistencies**
If the screens contain abbreviations (e.g. Nbr for number, Amt for amount, etc), the abbreviations should be consistent for all screens in your application. Again, the style guide is key for ensuring this.
10. **Save Confirmations**
If the screen allows changing of data without saving, it should prompt users to save if they move to another record or screen.
11. Delete Confirmations
If a person deletes an item, it is a good idea to confirm the delete. However, if the user interface allows deleting several records in a row, in some cases developers should consider allowing them to ignore the confirmation as it might get frustrating to click the confirmation over and over again.
12. **Type Ahead**
If the user interface uses combo boxes (drop down lists), be sure to include type ahead (if there are hundreds of items in a list, users should be able to skip to the first item that begins with that letter when they type in the first letter).
13. **Grammar and Spelling**
Ensure the test cases look for grammar or spelling errors.
14. **Table Scrolling**
If the application lists information in table format and the data in the table extends past one page, the scrolling should scroll the data but leave the table headers in tact.
15. **Error Logging**
If fatal errors occur as users use your application, ensure that the applications writes those errors to a log file, event viewer, or a database table for later review. Log the routine the error was in, the person logged on, and the date/time of the error.
16. **Error Messages**
Ensure that error messages are informative, grammatically correct, and not condescending.
17. **Shortcuts**
If the application allows short cut keys (like CTRL+S to save), test each shortcut to ensure it works in all different browsers (if the application is web based).
18. **Invalid Choices**
Do not include instructions for choices not available at the time. For example, if a screen cannot be printed due to the state of the data, the screen should not have a Print button.
19. **Invalid Menu Items**
Do not show menu items that are not available for the context users are currently in.
20. **Dialog Box Consistency**
Use a style guide to document what choices are available for dialog boxes. Designers should not have Save/Cancel dialog on one screen and an OK/Cancel on another. This is inconsistent.
21. **Screen Font Type**
Ensure that the screen font family matches from screen to screen. Mismatching fonts within the same sentence and overuse of different fonts can detract from the professionalism of your software user interface.
22. **Screen Font Sizes**
Ensure that the screen font sizes match from screen to screen. A good user interface will have an accompanying style guide that explicitly defines the font type and size for headers, body text, footers, etc.
23. **Colors**
Ensure that screens do not use different color sets as to cause an inconsistent and poorly thought-out user interface design. Your style guide should define header colors, body background colors, footer colors, etc.
24. **Icons**
Ensure that icons are consistent throughout your application by using a common icon set. For example, a BACK link that contains an icon next to it should not have a different icon on one screen versus another. Avoid free clip-art icons, opt for professionally designed icons that complement the overall look and feel of your screen design.
25. **Narrative Text**
Having narrative text (screen instructions) is a great way to communicate how to use a specific screen. Ensure that narrative text appears at the same location on the screen on all screens.
26. **Brevity**
Ensure that narrative text, error messages and other instructions are presented in laymen’s terms but are brief and to-the-point.
27. **Dialog Box Consistency**
Use a style guide to document what choices are available for dialog boxes. You should have not have Save/Cancel dialog on one screen and an OK/Cancel on another, this is inconsistent.
28. **Links**
If your application has links on the screen (e.g. Save as Spreadsheet, Export, Print, Email, etc.), ensure that the links have consistent spacing between them and other links, that the links appear in the same order from screen to screen, and that the color of the links are consistent.
29. **Menus**
If your application has menu items, ensure that menu items that are not applicable for the specific screen are disabled and the order in which each menu item appears is consistent from screen to screen.
30. **Buttons**
If your application has buttons (e.g. Submit, OK, Cancel, etc), ensure that the buttons appear in a consistent order from screen to screen (e.g. Submit then Cancel).
31. **Abbreviation Inconsistencies**
If your screens contain abbreviations (e.g. Nbr for number, Amt for amount, etc), the abbreviations should be consistent for all screens in your application. Again, the style guide is key for ensuring this.
32. **Delete Confirmations**
It is a good practice to ask the user to confirm before deleting an item. Create test cases to ensure that all delete operations require the confirmation. Taking this a step further, it would also be great to allow clients to turn off specific confirmations if they decide to do this.
33. **Save Confirmations**
It is good practice to ask the user to confirm an update if updates are made and they navigate to another item before explicitly saving. Create test cases to ensure that all record movement operations require the confirmation when updates are made. Taking this a step further, it would also be great to allow clients to turn off specific confirmations if they decide to do this.
34. **Grammar and Spelling**
Ensure that you have test cases that look for grammar or spelling errors.
35. **Shortcuts**
If your application allows short cut keys (like CTRL+S to save), ensure that all screens allow using of the consistent shortcuts.