So You Think You Can Write a Test Case

Srilu Pinjala (Sridevi)

http://www.linkedin.com/in/SriluPinjala

Abstract

Does the **QUANTITY** of test cases parallel to **QUALITY** of the test cases and compliment testing?

Issues with test cases:

1. Most companies have vague test cases or no test cases.
2. Test Data, environment not stated in test cases, the tester has to guess these conditions.
3. Test steps are vague with multiple actions stated in one step.
4. Expected result description is vague or null with no GUI changes described.
5. No way to figure out which modules / flows / pages / components have test cases.
6. No way to measure the completeness of a test case.
7. Focusing only on the requirement and testing with a tunnel vision.
8. Redundant test cases costing more time and money but adding no value as per testing.
9. Adding more test cases to increase count for each requirement introduced.

We will look at typical test case examples. Identify what they lack. We will rewrite them as scenario based test cases with sufficient verification. We will work to understand and establish verification criteria for Software in general and for each product in particular. We will learn to write test cases that are robust, reusable, recyclable and ready for automation. We will learn to write QUALITY test cases.

Learning Objectives:

1. The test case belongs to the product.
2. Test steps are of two kinds – for Testing and for navigation
3. Test steps are user actions on a graphical user interface with test set up and re-setup
4. Expected result are the changes on the Graphical user interface not user perception.
5. Do one thing at a time, stop cramming.
6. Increase Scenario based test cases with complete flows
7. Reduce redundant actions and increase actual tests
8. Organize test cases based on application components / modules.
9. Write it once, and Write it accurately.

This paper introduces the concept that the test case should belong to the product not the requirement. The method coerces the tester to think outside the box to make the product consistent and to maybe fix the requirement too.

Biography

Sridevi Pinjala (Srilu) has been a QA professional for 10 + years. She worked at Porch, Amazon, IBM, etc. She came up with Green Lantern Automation Framework when the scripts are agnostic to the User Interface. She presented at PNSQC in 2011 and 2012. She is an expert at testing customer facing applications. More fun Stuff – http://qasrilu.blogspot.com

1 Introduction

In the age internet of all things, cloud computing, mobile communication, advanced extreme programming, exploratory testing, we still do not have clear definition of how to write a test case.

There are tons of literature online about how to write a test cases with types of test cases, review dates, created date, requirement it belongs to, approved by, etc. But no literature about how to actually approach the art of writing a test case.

Scenario	Test step	Expected Result
Verify cell phone charges	Charge the cell phone	Cell phone charged successfully

Now, that was not so hard. Any idiot can write a test case. Well, a couple of things –

Charge the cell phone –

- Did we test with the right cell phone?
- How do we verify we are using the right cell phone?
- How do we know it was plugged in correctly?
- Where do we get the cell phone under test?

Cell phone charges successfully –

- Define what successful means!! A blinking light?
- A straight light? (What if it a notification light for a different application was understood by a tester as the charging light and they have been marking it PASS all through the project cycle!!)
- What about scenarios like - How long it takes to charge the phone? How do we know if it failed?
- What is the obvious fail criteria for this test?

1.1 Who would benefit from this paper?

- You are new at a company where the test cases do not make sense or are incomplete or vague.
- Your product has been acquired, you have no test cases or requirements to work with.
- New testers are completely dependent on legacy testers for application testing and set up.
- Test cases are not helping to test the application accurately or completely.
- Existing test cases are missing several steps, data, environment, etc.
- Test cases written by one tester are not understood by the other testers.
- You are constantly adding test cases, but the testing needs do not seem to be meeting.

2 What is a Test Case?

As per IEEE Std. 610.12-1990 –

(A) "A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program or to verify compliance with a specific requirement.

(B) Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.

2.1 What is the purpose of a test case?

"A test case has two purposes: to expose an error, or to demonstrate correct execution." (Jorgensen, 2014)

1. Test case describes how to test a product or process.
2. Test case guides the user to use the product accurately.
3. Test case assists in identifying defects in the product.
4. Test case helps expose errors during regression.

2.2 What are the different kinds of test cases? Why are they different?

A test case has steps that describe the steps needed to test a feature (positive, negative, edge, boundary value, etc.) and steps that purely help navigate to the desired feature to test it.

2.2.1 Exploring / Testing

- These test cases explore the application, pages, and elements, the flows, the functions, to verify various criteria.
- These test cases have one user action at a time with one or more verification criteria.
- These test cases describe the testing methods or user action to the T.
- These test cases can have plenty of steps doing all kinds of testing on one specific feature.
- They take up more test steps.
- They are very specific.

2.2.2 Piloting

- These test cases provide adequate description to go through steps with adequate verification to get to the features that will need to be tested.
- These test cases can contain more than one user action usually a whole function with adequate verification (Ex: Create account, Login as user Joe, Logout, etc.)
- One function can be stated for execution per test step with adequate verification.
- Two or more piloting test functions cannot be crammed into one step.
- They need be specific but not detailed.
- It is important to make sure every pilot test case has adequate test coverage / test cases to test it.

2.3 What does the Test Case belong to? Requirement, Project, Product?

Many companies focus on showing the requirements, user stories, use cases, featurettes have been met through the test cases. So, the test cases are modeled to be traced to the requirements have been satisfied. This was the norm for a long time. (Merely making sure the requirement has been met does not complete the testing of the product.) But the requirements change, and sometimes they are no longer relevant.

The projects are born to fulfil certain requirements. But they sunset one day too. It does not make sense to keep tracking back to the project from 3 years ago.

Projects come and go. Requirements come, change and go. The core product is forever. The test case belongs to the Product. The test cases should target to test the product. Showing the requirements have been satisfied is part of the testing but not the only purpose.

2.4 Who are the stakeholders for the Test Cases?

Every product has stakeholders and target user base. Test cases also have a user base. Identify the users to be able to fully provide sufficient information to them.

a) Author of the test cases, Other testers in the company
b) User Acceptance testers, Beta testers, End users
c) Business Analysts, UX developers
d) Developers, Architects
e) Project managers, Scrum masters, Program managers
f) Upper management, etc.

Often the author and other testers are the users of the test case. Seldom other team members can and should be able to use the test cases to understand and test the product.

2.5 What is the ideal time to start writing Test Cases?

The norm is for the test cases be written as soon as the requirement is signed off on. But requirements talk about what an application should do and should not do. Their descriptions are vague and left to interpretation. Their implementation is kept neutral on purpose to be abstract like. So, requirements can give you an understanding of what to expect and be prepared for but not write exact steps for testing.

It is assumed the test cases should be written by the time the build is ready. But there is no way to know the exact steps and elements involved in the tests. Writing the test cases at this point would result with vague steps.

So, the best time to write a test case is – After the first round of testing is complete for a feature.

2.5.1 After the first round of testing

1. The interface is available to look at. Pages, elements, data, conditions become clear which makes describing accurate.
2. Working off of requirements limits the test step and expected result to assumptions
3. Assumptions will not enable the tester to test accurately and verify accurately.
4. Test case serves for many QA cycles all through the life of the product, not just the first cycle.
5. One gets a chance to explore the application before documenting the steps and verification criteria.

3 Key Components

As per many online sources – Test Suite ID, Test Case Id, Test Case Summary, Related Requirement, Prerequisites, Test Procedures, Test Data, Expected Result, Actual Result, Status, Remarks, Created By, Data of Creation, Executed By, Data of Execution, Test Environment, blah, blah. Depending upon the company's philosophy, some or all or more of these components are necessary components of a Test Case.

But the components that add value to the purpose of testing are the Test Step and Expected Result. The rest of the components are bells and whistles. Some of those components only serve administrative purposes. Today we are here only to talk about the **Test Step** and **Expected Result**.

3.1 Test Step

The Test Step is the user action. It is also the tester's actions to input data, set up application state, navigate to the feature in test.

3.1.1 Sample User actions and Tester steps

- Go to *URL* **something**
- Type in *field* **something**
- Type in *field* **username** – Excel.xlsx > tab1 > column1
- Type in *field* **password** – Excel.xlsx > tab1 > column2
- Select from *drop down list* **something** - data
- Click *button* **something**
- Click *link* **something**
- Check ON or OFF *check box* **something**
- Hover over *menu* **something**
- Go back one page, Refresh the page, Go forward one page.
- SQL - [`SELECT userID, userKey FROM Users WHERE userKey = 7`]
- SQL - [`SELECT TOP 5 * FROM Search WHERE searchType = 'service' AND searchResult LIKE '%ab%' ORDER BY searchResult ASC`]
- In DDL Search only, Check ON checkbox **People,** Type in *field* **Search –** data
- SQL – [`DELETE FROM Users WHERE firstName = 'john' AND secondName = 'jason';`] & [`SELECT * FROM Users WHERE firstName ='john';`]

3.1.2 Test Step – Dos

- Keep it clear, simple, specific (CSS).
- Do it right to test it right. So, explain it accurately.
- Avoid company jargons, abbreviation, etc. Treat the software elements as such.
- One step at a time / one function at a time / one action at a time.
- Instruct to use specific data sets for testing.
- Test steps should be present tense. The same tense the tests are being executed.
- Like talking to a Robot that does not understand logic and reasoning, only specific instructions.

3.1.3 Test Step – Don'ts

- Avoid instructions like – repeat the above steps or execute steps from 6 to 9. Tell the tester to execute specific functions instead.
- Don't cram multiple steps or functions into one test step. Verifications can be lost.
- Don't just mention a feature and expect the tester to know what to do with it.
- Don't just say use valid data or invalid data.

3.2 Pilot functions

It is best to execute one step at a time while exploring and testing a feature. But often we will need to execute certain functions to get to the feature we are going to test. These functions help navigate. They are crucial for this test case. But they do not need to be tested in this test case because they have been tested thoroughly in other test cases.

- Login to app – implies – go to URL, Log in and verify
- Log out – Click on header, click on log out link, verify logged out.
- Delete User – Delete a specific user from the data base using a SQL query.
- Register Account bankrupt – DQL for bankrupt Account, go to URL, and Register the account.

3.3 Expected Results

Expected Result is the changed state of the AUT. Typically involves GUI changes that are instantly visible, Data changes that need to be verified, other changes that have to be verified using tools.

3.3.1 Sample Verification Criteria

- Page appears with *title* **something**
- Header consists of *links* **something**, **something**
- *Error message* appears – "**something**"
- Dialog box **something** appears.
- SQL Verification **-** `userID = srilu.pnsqc@gmail.com`
- SQL Verification – 0 results.
- Logo - **International** (logo.png); Tabs - **View Saved Properties, Change Location, English (United States);** Text – **Global;** Field - **Search** with placeholder "Search"; Button – **Search;** Drop down list : ***(search only);*** Menus - **Research, Search Properties, Explore Our Services, Blog, About**
- "Page appears with Title – "Search | Global | Something International", CSS - (Home page elements), Header Search field is blank, Body - Search Results with field Search and Button Search, Links - Go to Property Search > Left Nav - Show Only"
- Auto-Complete appears with up to 5 matching suggestions with only Service names
- SQL Result - Top 5 rows match the search results

3.3.2 Expected Result – Dos

- Keep it clear, simple, specific (CSS)
- One step can have multiple verifications. List as much verification as possible.
- List and explain with IF and WHEN conditions where applicable.
- The description should be WYSWIG (What You See is What I Get)
- Verify page titles, logos, text, images, error messages, etc. verbatim.
- The results have to be tangible and verifiable, be it GUI or be it Data or both.
- Expected results should be in present tense. The same tense the tests are being executed.
- Like talking to a Robot that does not understand logic and reasoning, only specific instructions.

3.3.3 Expected Result – Don'ts

- Don't say "*it is successful*". Explain what defines successful.
- Don't say "*it has no errors*". Explain what no errors means.
- Don't say "system accepts". Explain how the system accepts, (new page comes up?)

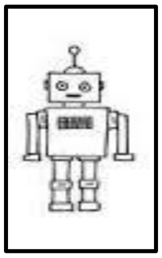
3.4 Verification for Pilot function

Pilot functions may have multiple steps and multiple verifications. It is not necessary to verify all the steps. So, it is ideal to verify at least one unique aspect that are crucial for this test case.

- Every function has at least one verification criteria.
- Verify page titles when on a new page.
- Verify login criteria that is unique to the credentials or account.
- Verify data on the GUI or in the Data base or both.
- The main reason to verify a pilot function is to make sure the application is in the desired state.

4 Go the Extra Mile

Test case is the place where the tester can be equipped with the tools and information they need, to get the testing done quickly and efficiently. Test cycle is not the place to test the tester's technical skills, do not assume the tester to know about the product automatically.



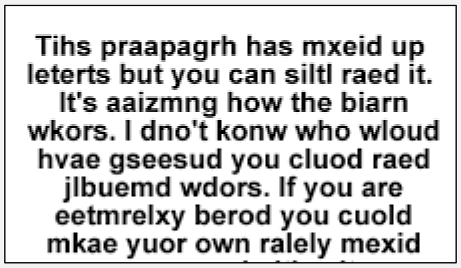
4.1.1 Like talking to a Robot

Enable the tester to execute the tests independently. The best way to achieve this goal by assuming the tester is a **ROBOT** who does not understand logic or assumptions. They only understand ***specific*** instructions.

Can you read the words in the Square?

If you can read them without issues, you have missed a lot of typos you could report!! (Human Error)

If only you copy pasted the text in a word document, you would have caught the typos easily. Suggest such ideas to the future test performer too.



Tihs praapagrh has mxeid up leterts but you can siltl raed it. It's aaizmng how the biarn wkors. I dno't konw who wloud hvae gseesud you cluod raed jlbuemd wdors. If you are eetmrelxy berod you cuold mkae yuor own ralely mexid

(Read the text in the square box?)

4.2 Wikis

Information gets lost over time. Create and maintain Wikis with information about various test components. Reference this information in the test cases where necessary. The Wikis can be updated periodically, the test cases need not be touched.

4.2.1 Environment

- Server operating systems, client operating systems, database servers, browsers, versions, etc.
- Don't just mention the OS, browser and its version, provide the location of these tools.
- Sometimes a browser needed for the test won't install on the standard operating systems. It will be helpful to provide information about which Virtual Machine to use and what credentials to use.
- If the test requires for connecting to a Database, provide information about names and credentials to be able to connect to the database.

4.2.2 Virtual Machines

Specify or list information about Virtual Machines with the necessary browser versions and other applications along with the credentials needed to use to log into them in the Wikis.

4.2.3 Databases

Specify or list the databases relevant for the product and testing in Wikis. Note down the credentials needed to access them. Also note down the owner for the database, along with where to get the credentials from, if needed.

4.2.4 Data Sheets

Create and maintain Excel spreadsheets (typically) that can be used for manual as well as with automated testing in an agreed location. Specify the location for each file in the wiki. If the location of the file name changes, the wiki needs to be updated but the test case does not. The data in the file can be updated without affecting the wikis or the test cases.

4.2.5 Test Data

- Test data may be listed in the test steps.
- It may be listed in a separate column too. That will make editing it a lot easier.
- “Valid account”, “invalid credentials”, “system admin”, etc. is not test data.
- Provide links to the Wiki. Provide the location of the Excel sheets with test data.
- Provide SQL queries with step by step instructions to locate the needed test data.

Provide steps and tools to create test data whenever necessary in the Wikis, which will make it easy to update the steps when needed.

4.2.6 Test Setup and Re-Setup

- Test set up is not a one-time thing before the execution of the test. It could be done during too.
- Environment may be set up and updated and modified several times during the test execution.
- Data may be set up and updated and modified several times during the test execution.
- Example – Register through SQL -> Login through GUI -> Update the registration date through SQL -> test password reset through GUI.

4.3 Pictures

Create and maintain Wikis with pictures of screen shots that will need to be verified. It will be easy to update the Wikis rather than the individual test cases.

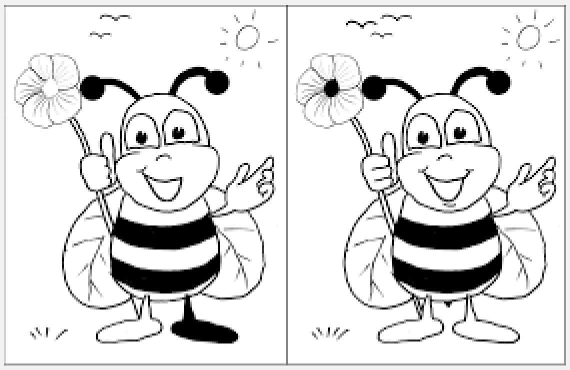
Providing a picture in the test case is an excellent way to describe the steps and expected results. But there are things one can do to make the process more efficient.

Find the differences

Comparing pictures is an exciting game. It is also mostly hit and miss. You do not want to take that chance. If you do, you might miss out on important verification.

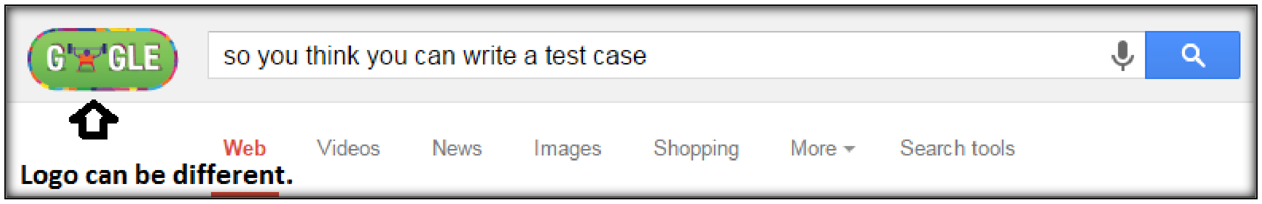
So, highlight the crucial elements that need verification. This will make the job of the tester easier and quicker.

There are a lot of new features to test all the time, so make sure the older features have been tested and verification criteria defined in the test cases.



(Can you find the differences in these pictures? I am not going to.)

4.3.1 Highlight



Posting a picture is not efficient enough. Highlight what to look for in a picture and what to ignore.

Unless the tester has a good understanding about the product and what changes to look for and ignore, you will end up with a bunch of missed bugs or more annoyingly a bunch of unnecessary bugs. That is a lot of resources' time and money and energy (and bruised ego).

5 Where to start

Where to start in the absence of any requirements or test cases or documentation. Companies go through acquisitions, mergers, reorgs and major changes. Companies let go of QA teams during hard times. During which documentations related to an application are lost. What are you supposed to do?

Revise and Study the product, Mind map the application, Explore each element, and Design test cases by naming and organizing them, Integrate test cases as scenarios with accurately fitting data and other information. Revise the test cases from time to time.

5.1 Revise the product

Use the product that is in front of you. Explore the product. Treat an element like an element. Treat a widget as a widget. Go further and understand what Databases, tables and columns the elements talk to. Most applications out there are GUI based, so this example will be for GUI based products. Study the application.

- Authentication roles
- No. of pages
- CSS Elements and Unique elements in the application.
- Search features
- Email and communication features
- Session time out and other time based functions.
- Other features, Etc.

5.1.1 Pick a style

Decide if you want to model your tests around the GUI features, Core functions or flows. The style is based on your best judgment. What works for one product won't work for others. But, failing to plan is a plan to fail. So, pick a style. The style can be changed at any point very easily.

5.2 Map the Application

Map the application by the pages, and its elements based on the hierarchy. This is your **visual mind map** that can give you a visual for the elements you know about, their hierarchy, etc.

You may list all the known characteristics for each element in the following cells, like DB it talks to, field validation, how it reacts, etc. This visual mind map may be used for your initial round of testing as an unofficial test case.

5.2.1 Example 1 (MS Excel MAP)

Level 01	Level 02	Level 03	Level 04	Level 05
menu Home				
	group Clipboard			
		icon cut		
		icon copy		
			Copy	
			Copy As a Picture	
		icon format painter		
		icon Paste		
			section Paste	
				icon Paste
				icon Formulas
				icon Formulas & Number Formatting
				icon Keep source formatting
				icon No borders
				icon keep source column width
				icon Transpose
			section Paste Value	
				icon Values
				Icon Values & Number Formatting
				Icon Values & Source Formatting
			section Other paste options	
				(recently used option appear)
			section Paste Special	
				Dialog box Paste Special

MS - EXCEL

5.2.2 Example 2 (Login page)

Level 01	Level 02	Level 03
page Login		
	field **Email or Phone**	
		EM Enter your Username
	field **Password**	
		EM Enter your Password
	checkbox **Keep me signed in**	
	checkbox **Remember my password**	
	button **Sign In**	
		EM The username or password you entered is incorrect
	Link Can't access your account?	

	link **Sign in with a single-use code**	
	logo **Outlook**	

Hotmail – Login section

This visual mind map may be used for your initial round of testing as unofficial test cases. Use this map to track your testing status and coverage just by marking them cleverly with color or *font* type.

5.3 Explore each element

Once we have mapped out the visible elements, explore each element and understand what they do.

- URL - invoke it in multiple browsers and operating systems.
- Page – refresh it, resize it, and go back and forth, Note down the elements on each page.
- Go through as many pages as possible and group pages based on similarities.
- Separate CSS elements from unique elements on a page.
- Look for spelling and grammar slip-ups, consistency in font, colors, images, and error messages.
- Authentication – understand the login flows, the data, login types, user roles, etc.
- Profile information – set up and edit and delete profile information and the many ways to do it.
- Search – when does autocomplete populate, what kind of results show up.
- Session time out – Look in the source code, configuration files, data base or just observe
- Text Field – fill it up. Understand the field validation. What kind of characters it accepts and how many and what it does not and under what conditions.
- Links – click them and understand what they open up to. Do they respond to hover, or click?
- Buttons – click them and understand what they process.
- Check boxes – check ON and OFF each option and understand what they take into account.
- Menus and Sub menus – Observe where they navigate to.
- Explore - Dialog boxes, tool tips, help files.
- Databases – note down which tables and columns are talking to the UI elements.

5.3.1 Exploratory test cases for the Hotmail Login page

Page	Attributes
Go to URL **Hotmail.com**	Page open with title – Sign In Left section – Image – www.wiki.com/mycompany/signimage (fake wiki) Field – enter your phone number and we'll send you the download link; Button – Send My Link Right section – Logo – Outlook Fields – Email or Phone, Password Checkbox – Keep me signed in Button – Sign in Links – What's this?, Can't access your account?, Sign in with a single-use code Footer section – Links – Contact Us, Terms of Use, Privacy & Cookies, Link Disclaimer ©Copyright

Element	Attributes
filed **Email or Phone**	1. DB – SELECT email FROM users; 2. Placeholder text – Email or phone 3. If registered email is typed – Page Inbox appears. 4. Field Validation – Accepts letters, numbers and characters (period, hyphen, underscore), 256 digits. 5. If field is left blank – login page persists with the typed in username and error messages in red font “Enter your Username” 6. If unregistered email ID is typed – error message “The username or password you entered is incorrect”.

Element	Attributes
filed **Password**	1. DB – SELECT email FROM password – (is encrypted) 2. Placeholder text – Password 3. If registered email is typed – Page Inbox appears. 4. Field Validation – Accepts letters, numbers and characters (period, hyphen, underscore), 256 digits. 5. If field is left blank – login page persists with the typed in username and error messages in red font “Enter your Password” 6. If unregistered email ID is typed – error message “The username or password you entered is incorrect”.

Element	Attributes
check box **Keep me signed in**	1. ON – the user stays logged on the particular browser when navigated to the Login URL. 2. OFF – the user will have to have to login when navigated to the Login URL

Element	Attributes
button **Sign In**	1. Unregistered email and password typed in – Error message “The username or password you entered is incorrect” 2. Gibberish email and password typed in – Error message “The username or password you entered is incorrect”. 3. Registered email and gibberish password typed in – Error message “The username or password you entered is incorrect” *4.* Registered email and registered password typed in – 5. *Page opens with title – **Inbox*** 6. Registered email and registered password typed in (using a new IP address) – Page open with title – verify your email

Element	Attributes
link **Sign in with a single-use code**	Same page persists (title – Sign In) with message – “A single-use code lets you sign in without entering your password. This helps protect your account when you're using someone else's PC” with link “Learn more” Fields – Microsoft account, Phone number, Links – What’s this? Learn more, Already have a code? Button – Text me the code

5.4 Design

Design test cases for all possible scenarios based on the findings. These test cases are simple. These test cases can be used individually or in test sets with multiple iterations and data too. Keep them ***CSS***.

As per the mapping, it is easy to remember each element’s hierarchy. These test cases should have an information as to what the pre-requisites are for using them. Some may have none, all will need the product to invoke. Post a picture and highlight what to click on and what to look for.

5.4.1 MS-Excel

Sample test cases for Excel.

5.4.1.1 Click Icon CUT

Test Step	Expected Result
Select any amount of text. Click icon CUT (Home > Clipboard > Cut)	The selected text is cut (disappears) from the page / cell.

5.4.1.2 Click Icon COPY

Test Step	Expected Result
Select any amount of text. Click icon Copy (Home > Clipboard > Copy)	The selected text persists on the page. It does not disappear from the page.

5.4.1.3 Click Icon COPY: Formula

Test Step	Expected Result
Type random numbers in any column’s 10 cells. Type in formula to add the numbers in the 11th cell [=SUM (A1:A10)] and press enter.	The numbers in all 10 columns are added into the cell 11 (no need to verify the calculation)
Select the 11th cell and Click icon Copy (Home > Clipboard > Copy)	The selected text persists on the page. It does not disappear from the page.

5.4.1.4 Click Icon PASTE

Test Step	Expected Result
(After using cut or copy with selected	The selected text is pasted (appears) where the cursor is

text) Click icon PASTE (Home > Clipboard > Paste)	placed.
Place the cursor in a new cell and Paste	The selected text is pasted (appears) where the cursor is placed.
Place the cursor in the middle of text in a cell and Paste	The selected text is pasted (appears) where the cursor is placed.

5.4.1.5 Click Icon PASTE: Formula

Test Step	Expected Result
Execute test case: Icon Copy: Formula	
Click in the 11th cell of the next column Click icon Paste Formula (Home > Clipboard > Paste > Formula)	Zero appears in cell.
Type random numbers in the first 10 cells of the same column	The Zero value is replaced with the value of added numbers in the 10 columns.

5.4.2 Naming

Naming and organizing the test cases is a crucial part of the design phase. The name of the test case must help in recognizing what product, flow or feature it belongs to and help organize it too.

Listing the project name in the test case name does not add any value. Naming the test case after the requirement creates a traceability, but does not ensure total test coverage. The name of a test case should be able to tell the audience what area of the product the test case belongs to.

Name the test case after a feature, element or function or flow of the application.

The test case name can be divided into parts. Each part indicates a certain aspect of the product.

First part - indicates the **product**

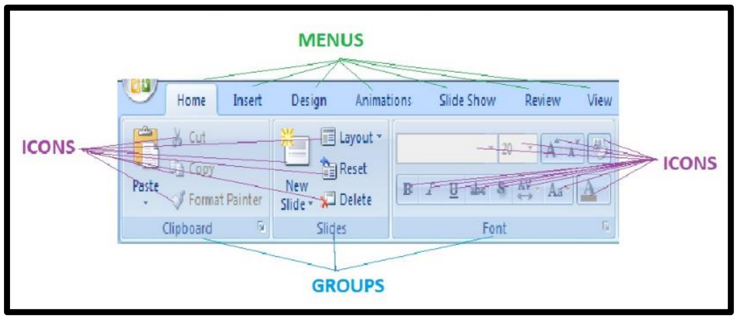
Second part - indicates the **module** or **page** or **flow**

Third part - indicates the **feature**

Fourth part - indicates the **function**

A test case can have more parts than 4

5.4.3 Test Case names for each features



- MS.PPT.mHome.iCut = Microsoft > Power Point > menu Home > icon Cut
- MS.PPT.mHome.iPaste.oPasteSpecial = Microsoft > Power Point > menu Home > icon Paste > option Paste Special
- Mail.fUsername = Mail > Username
- App.Search.AutoSuggest = App > Search > Auto Suggest

5.4.4 Test Cases / Test set names for Pilot functions

Pilot functions may also be written separately. This eliminates the redundant steps while navigating. Remember, there is little value in testing over and over. Test it once and test it right.

- Mail.Login = Username + Password + Sign In + Verify Login
- Mail.Register = lots of elements and fields + Verify Registration
- MS.PPT.Close.withoutsaving = exist + click no on dialog box save

5.4.5 Organize

Organize the test cases as per your plan (Flow or Feature or Functions or Element). What makes most sense to your team and what is best suitable for the product. It is all about being organized and being able to locate a test case easily for execution, editing, merging or deleting as needed.

5.4.5.1 Sample folders and test cases

Home – test cases for common elements on all pages.

Create Profile – Register through email, register through Facebook, register half way, etc.

Authentication – with email, Omni authentication, merge with Facebook, etc.

My Profile – Forgot password, change address, etc.

Upload images – Upload image files, data files, documents, delete, edit images, etc.

Communication – Send request, forward profile, announce, etc.

SQL – Select UserID7, Delete Registered users, Reset Password, etc.

5.4.5.2 Organizing test cases like this gives

- Visibility of coverage
- Minimizes execution time
- Eliminates redundancy
- Saves time when time comes for test sets.

5.5 Integrate

Test case/test set with all possible combinations as scenario based test cases are integrated in this phase. Typically the blank fields, valid credential and invalid credentials flows are separated into 3 test cases, there is no reason why the tests cannot be combined into one test case or test set. Combining tests has to be considered on a case to case basis. The obvious advantage of writing test cases like this – reduction in execution time.

This is where the test cases/set are updated a little more to tell a story with each scenario. These test case/sets can be linked to one or more requirements. They are basically user flows at this point with specific or various outcomes that have been defined. (Remember, it is not just about the product, enable the tester too, don't forget to provide them the specifics and the tools to get the testing done.)

Rule of thumb for organizing test cases as test sets maybe based on -

- Test execution order
- Priority
- Dependency

5.5.1 Example Login test case

Test Step	Expected Result
Go to URL **mail.com**	Page open with title – Sign In Left section – Image – www.wiki.com/mycompany/signimage (fake wiki) Field – enter your phone number and we'll send you the download link. Button – Send My Link Right section – Fields – Email or Phone, Password Checkbox – Keep me signed in Button – Sign in Links – What's this?, Can't access your account?, Sign in with a single-use code Footer section – Links – Contact Us, Terms of Use, Privacy & Cookies, Link Disclaimer
<u>(Blank fields)</u> Leave *fields* **Email** and **Password** blank. Click *button* **Sign In**	Error messages appear in Red Font – **Enter your email, Enter your password**
Refresh the page	Error messages do not show anymore.
Type anything in the *field* **Email** (asdgkdf) Leave the *field* **Password** blank. Click *button* **Sign In**	Error message appears in Red Font – **Enter your password**
Refresh the page	Error messages do not show anymore.
<u>(invalid credentials)</u> Type in the *field* **Email** (example1, example2, etc.)	Error message appear in Red Font – **The username or password you entered is incorrect**

Type in *field* **Password** (asdfggdg34@#$) Click *button* **Sign In**	
Refresh the page	Error messages do not show anymore.
<u>(Registered credentials)</u> Type in the *field* **Email** [SELECT userID, userKey FROM Users] Type in *field* **Password** [userKey = 12, use password **Pass1;** userKey = 13, Use password **Af14!!**] Click *button* **Sign In**	Page appears with Title - **Inbox *(number of emails, if any)* – *(email)* – Gmail** Header elements….. Left Nav….. Footer elements…. Body elements ……
SQL – SELECT * FROM accessTracker WHERE userID = '***email***' ORDER accessTime ASC	accessIN = '**current time**' (00:00:00)
Go back one page	Same page persists. Login page with login fields does not show up
Click *button* **Sign Out**	Page appears with *title* – **Sign In**
SQL – SELECT * FROM accessTracker WHERE userID = '***email***' ORDER accessTime ASC	accessOut = '**current time**' (00:00:00)
Click *link* **Can't access your account?**	Page open with *title* – **Why are you having trouble signing in**
Click *button* **Cancel**	Page appears with *title* – **Sign In**
Click *link* **Sign in with a single-use code**	Page persists (*title* – **Sign In**) with message – **A single-use code lets you sign in without entering your password. This helps protect your account when you're using someone else's PC** with *link* **Learn more** Page screenshot - www.wiki.com/my/signimage Fields – Microsoft account, Phone number, Links – What's this? Learn more, Already have a code? Button – Text me the code

More steps may be added to this test case for forgotten password, code, logging in with phone number, etc. (I am keeping this short for this paper)

Integration is NOT JUST ABOUT grouping with other test cases. This is the phase where the test data and test environment information is added to the tests. Links to Wikis, locations to Excel sheets, specific test data are added to the test sets.

5.5.1.1 Test Data

- List SQL queries where necessary to enable the tester to get to the Data needed.
- Maintain clean test data in Wikis or in Excel sheets that are accessible to all.
- Log in as Valid, registered, system admin means nothing.
- What qualifies as a valid, registered, etc. means everything.
- Suggest the sample test data. Provide SQL queries to help the test executer find the data needed for testing. Going the extra step saves time in the long run.

5.5.1.2 More

- Enable the tester to get back from the changed state (refresh the page, go back one page, etc.)
- If the tester needs to update things in Database for next steps, list the instructions to do so, step by step.

5.5.2 Pilot Test Case for Login

Test Step	Expected Result
Log in as System Administrator [SELECT userID, userKey FROM Users WHERE userKey = 7] or [Wiki – http://wiki.com/proj/pass] Or [Users.xlsx > tab – 2015]	Page appears with Title - **Inbox *(number of emails, if any) – (email)***

The pilot login function serves the purpose of navigation only. Verifying the page title suffices. These test cases do not test. They help navigate to the desired destination and the desired state for testing other features. The login pilot can be used in combination with other test cases.

5.5.3 Test case with Data Sheet

Search is a feature that can be used with multiple keywords. In such cases, use a data sheet as below. This way one test case can be used with multiple data values with obvious results.

Test Step	Data	Expected Result
`SQL - SELECT TOP 5 * FROM Search WHERE searchResult LIKE IN ('Data') ORDER BY mostRelevant ASC;`	Type in	0 to 5 records appear. (Note down the results and records from `searchResult` as Step 1)
`SQL - SELECT * FROM Search WHERE searchResult LIKE IN ('Data') ORDER BY mostRelevant ASC;`	Click On	(Note down the results and records from `searchResultLinks` as Step 2)
`SQL - SELECT COUNT * AS count FROM Search WHERE searchResultLink LIKE IN('Data');`	First Result	(Note down the count as Step 3)
Type in *field* **Search**	Type in	If Matching results exist – Auto-complete appears with up to 5 matching suggestions. Compare the results from SQL Step 1 If no Matching results exist – Auto-complete does not appear at all
If auto- populate appears Click on *search Result*	Click On	Auto-complete closes. Selected text appears in the *field* **Search.**
Click on *button* **Search**		Page appears with title – "Search". Typed in text persists in the *field* **Search** Body appears with – • Text– Search Results ***(count) [SQL step3]***

		• Footer text – Page 1 of ***(SQL step 3/10)*** • VCR buttons for pages appears if more than 10 results count from SQL step 3 • 10 links appear as search results.
Verify the first link in the results		Matches the ***First Result*** in the data sheet.

5.5.3.1 Test data sheet

Type in	Click on	First Result
sa	san Francisco	san Francisco jeans
bob	bob chodos	bob chodos profile
zzzzzz		
123	123 Studios	123 Studio Seattle
car	Car dealers	Volkswagen
Act	Acting	Mel Brooks
Sail	Sailing boats	Seattle boats

Using a data sheet in this scenario reduces the number of test cases and increases the iteration count.

Use If conditions in the Test cases.

Every element has multiple attributes and purposes and reacts differently based on some logic. If this logic is clearly defined in a test case, several tests can be run in a single flow.

5.5.4 Pilot Test Case for Search

Test Step	Data	Expected Result
Type in *field* **Search** Click on auto-complete **txt** Click *button* **Search**	Type in, Click on	Page appears with title – **Search Results** First link – **Something**

5.6 Repeat the cycle

Test cases are not a onetime thing. The test case will be documented and revised and refined several times through the life of the product.

- Set up time to go over the sets of test cases periodically. Review and update as necessary.
- New test cases may be added to keep up with the new requirements and new features.
- Two or more test cases may be combined. This depends on a case to case basis.
- Some test cases could be deleted due to redundancy. You don't need to test one thing twice.

5.6.1 Change in Features

An existing feature of the product has a new requirement it needs to satisfy.

- New test cases may not be necessary.
- Add or update steps to existing test cases as necessary.

5.6.2 New Feature

A new feature is being introduced in the product.

- New test is necessary.
- Add a new test case for testing and pilots.

- It may also be necessary to update other existing test cases to ensure consistency.

5.7 Owner

Test cases are part of knowledge management. Just like the knowledge management can be successful with an owner, the test cases can also be managed and kept up to date by an owner. It is necessary to have an owner for the Test Cases of any given product.

The test case author does not become the owner by default. It could be a Lead, or Manager or a product owner who is the owner. Typically it is the QA team that has the ownership.

The owner is in charge of the document's upkeep. They may or may not physically edit and update the documents. They will have trained testers in place to update and edit the test cases and concerning documents.

6 Typical Test Cases

6.1.1 Typical test case for Logging

Test Step	Expected Result
Login with valid user ID and password	Logged in successfully
Login with invalid User ID	Not logged in

In this case, what is a valid user ID and what is an invalid User ID? Unrequested? Invalid chars? What does "logged in successfully" mean? How can we be sure we verified log in correctly.

6.1.2 Typical test case for search

Test Step	Expected Result
Search for "San Diego"	Results for San Diego are found
Search for "!@#$$"	Results for !@#$$ are Not found

The test step is not specific but it somehow makes sense. But this is not enough when testing.

6.2 Revise a typical Test case

6.2.1 Typical test case for invoking URL

Test Step	Expected Result
Go to URL – https://www.google.com/	Page loads successfully Or Page throws no errors

This test case seems like a good enough test case. But what defines the criteria for being "successful"? What does "no errors" mean for the Application under Test? What is successful for Yahoo home page is not the success criteria for Google or some other home pages!! So, define the success criteria.

6.2.2 Revised test case

A simple Go To URL – (something) suffices as a test step.

Most companies have multiple environments for every application. It is a good idea to maintain one test case with all the environments listed, so that the test case uses one of those URLs based on the test cycle or test plan. All the information we need to test and verify in one location. Described simply.

Test Step	Expected Result

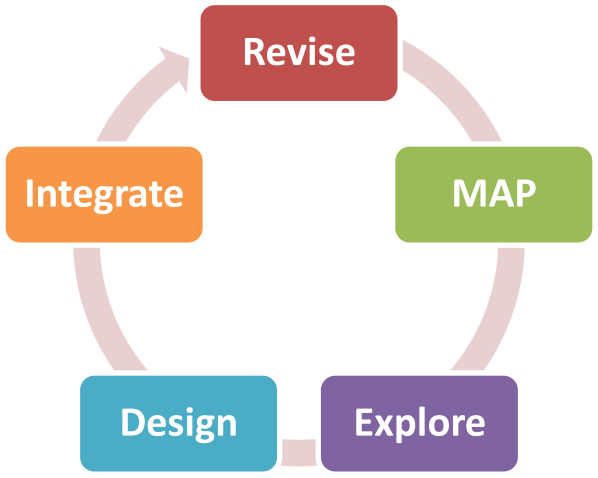
In Chrome browser incognito.
Go to URL – https://www.google.com/
Or
Go to URL –
QA1 = http://qa1.com
QA2 = http://qa2.com
Prod = http://prod.com
Stage = http://stage.com
	Page opens with title – Google
	Header consists of –
	• Links +You or +*(Username)*, Gmail, Images
	• Buttons – [app group buttons], Sign in (blue)
	Body consists of –
	• Logo – Google
	• Text Field – with placeholder text Say “OK Google”
	• Buttons – Google Search, I’m feeling Lucky
	Footer consists of –
	• Links – Advertising, Business, About, Privacy, Terms, Settings

6.2.3 Pilot Test case

Test Step	Expected Result
Go to URL – https://www.google.com/	Page opens with title – Google

We have verified the page elements in the main test case. We are using this step as a pilot. Hence, it is ideal to do some verification. Verifying the title of a page is the best and simplest verification.

7 Test Case Life Cycle



<u>Life Cycle of a Test Case</u>

Revise – the application and understand the best approach for testing it and managing test cases.

MAP - the application based on the pages, elements, flows, functions or features.

Explore – the code, interfaces, databases, environments, application, etc.

Design – the test cases with appropriate names and organize them in the appropriate category.

Integrate – the mini test cases with various test data and environment information, etc.

Note

- Test Case is mere ***data***. It is instructions and verification in its simplest form.
- Test step and Expected result is ***information*** about the test. It is like a journal with information about what was tested and how it was tested and what tools were used for testing it. If the data is not clear and complete the information is not complete for future test execution.
- Test Case execution status becomes the ***knowledge*** that helps understand the health of the application, product or feature.
- The knowledge helps, us the humans, make an informed ***decision*** about the release, sprint, project and most importantly the product.

8 Summary

Test case is not a document that assumes what could be tested when the product is developed. It is a document that describes how it was tested and how it works. If the document can be furnished with the tools necessary for testing, supporting the product through its life will become stress-free and doubt free.

Test case supports the product through its lifecycle. Not just the first release.

Test case **belongs** to the Product, not the requirement or the **project**.

The best time to write a test case is after the **first round of testing.**

Talk to the Robot. Robots do not understand much. They need specific instructions.

Test cases are two types – **Explore, Pilot**

Exploring test cases explore the feature thoroughly with specific verification.

Pilot test cases navigate through the steps they do some verification.

Specify **one** user action or pilot at a time.

One test case can have **more** than one verification criteria.

Don’t tell the tester what to do, tell them **how to** do it.

Provide the **tools** to get the job done accurately and rapidly. Go the extra mile to **equip the tester**.

Provide exact **SQL** queries for execution and verification in the test case.

Picture says a thousand words, so **highlight** what to look for in a picture.

Periodically **revise** the test cases to add, update, delete and merge as necessary.

Have an **owner** who will oversee the test creation process and standards for the test cases.

Following this method of documentation won’t give job security, but it will give job satisfaction!!

Works Cited

Jorgensen, P. C. (2014). *Software Testing: A Craftsman's Approach 4th Edition.* Boca Raton, Florida, USA: Taylor & Francis Group, LLC.