Test Cases

Trainer: Atanas Radkov
Email: atanasvr@gmail.com
Facebook: facebook.com/atanasvr

PRAGMATIC
www.pragmatic.bg

About me:

Software Engineer @Skyscanner
Testing since 2005

atanasvr[at]gmail.com

in/atanasradkov/

@atanasvr

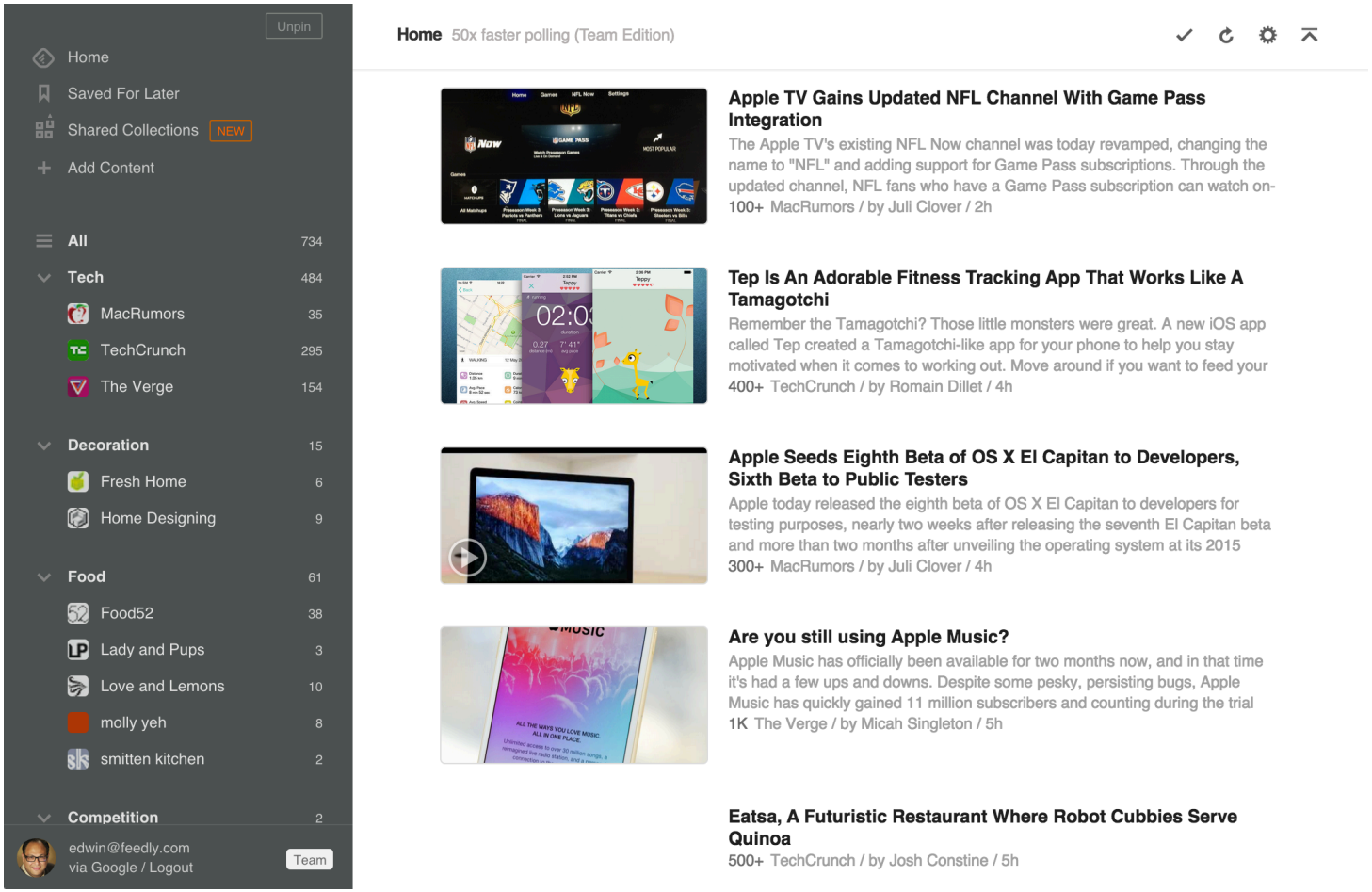
facebook.com/atanasvr

What we'll teach you

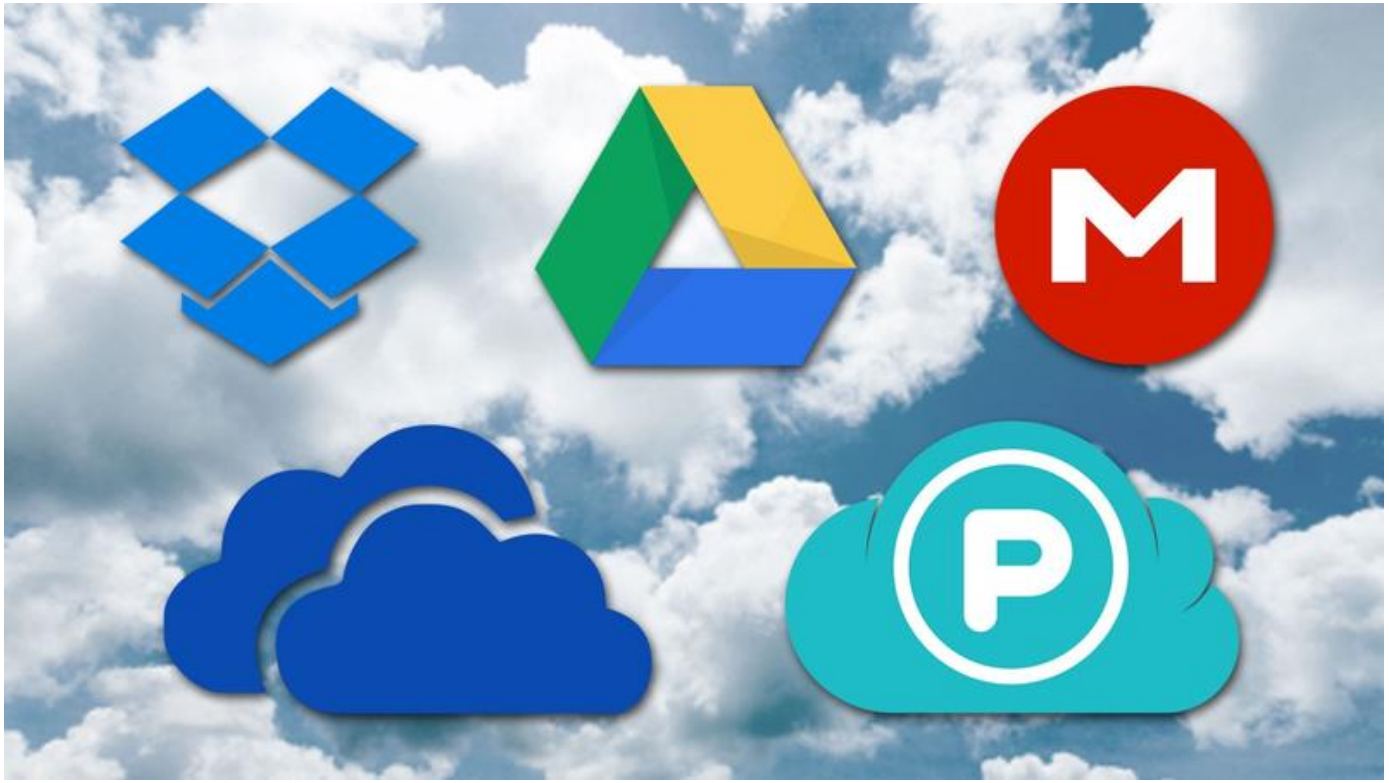
Be like a child

RSS Reader

Tip: You can use Feedly https://feedly.com/ to follow blogs and websites.



Use cloud storage



Read it later

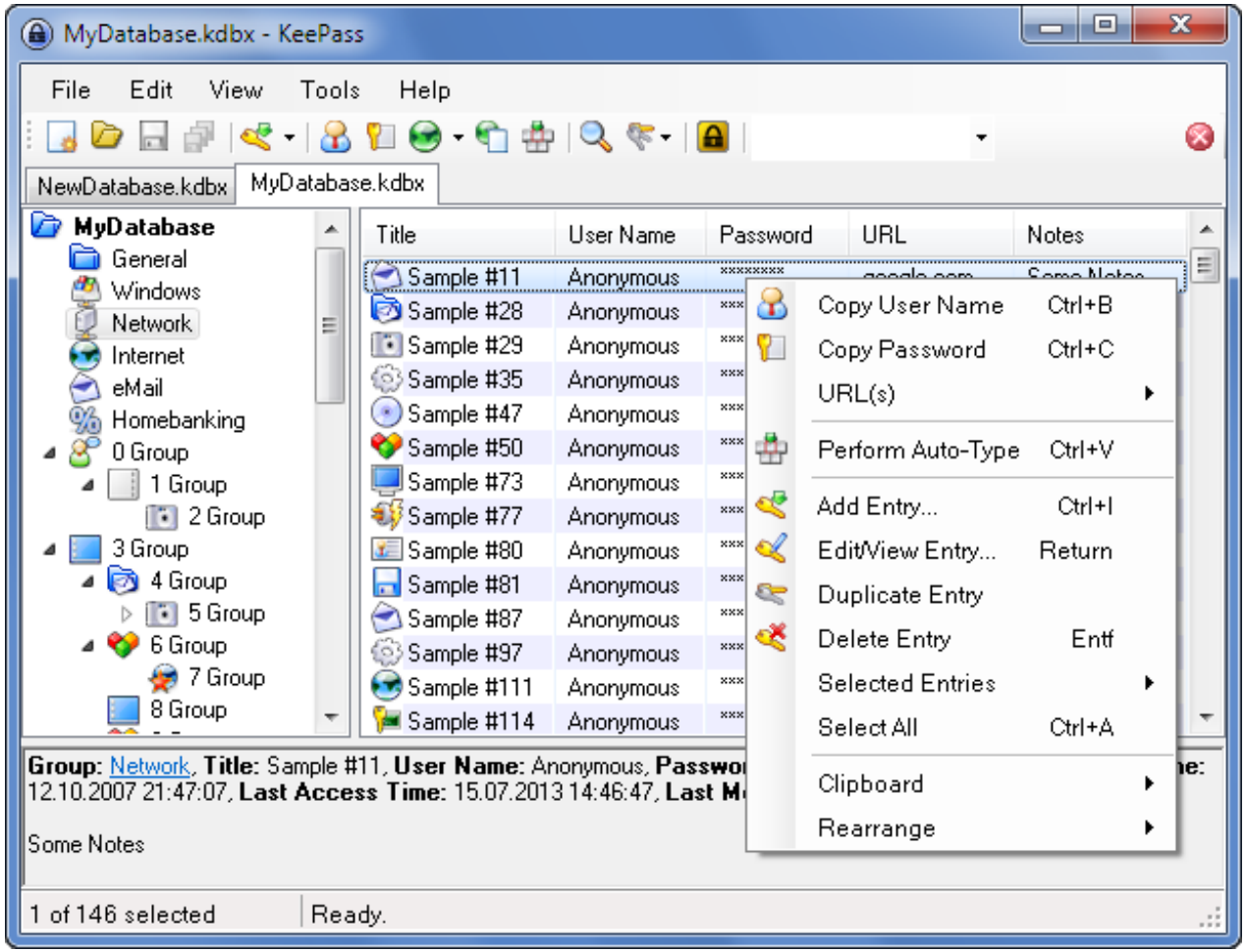
Tip: You can use Pocket www.getpocket.com to save and organize links and articles you want to read later or offline.

Listen To Podcasts

- History: https://www.bghistorypodcast.com/
- Language: The English We Speak, Business English Podcast
  - https://www.bbc.co.uk/programmes/p02pc9zn/episodes/downloads
  - https://www.businessenglishpod.com/about/free-podcast-subscription-options/
- Business: https://hbr.org/2018/01/podcast-ideacast
- Testing ☺
  - https://blog.gurock.com/5-software-testing-podcasts-to-entertain-educate-inform-and-inspire-you/
  - https://player.fm/podcasts/Software-Testing

Use Password Manager

https://lifehacker.com/5529133/five-best-password-managers



Lorem Ipsum

https://www.lipsum.com/

Lorem Ipsum

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..."

"There is no one who loves pain itself, who seeks after it and wants to have it, simply because it is pain..."

What is Lorem Ipsum?

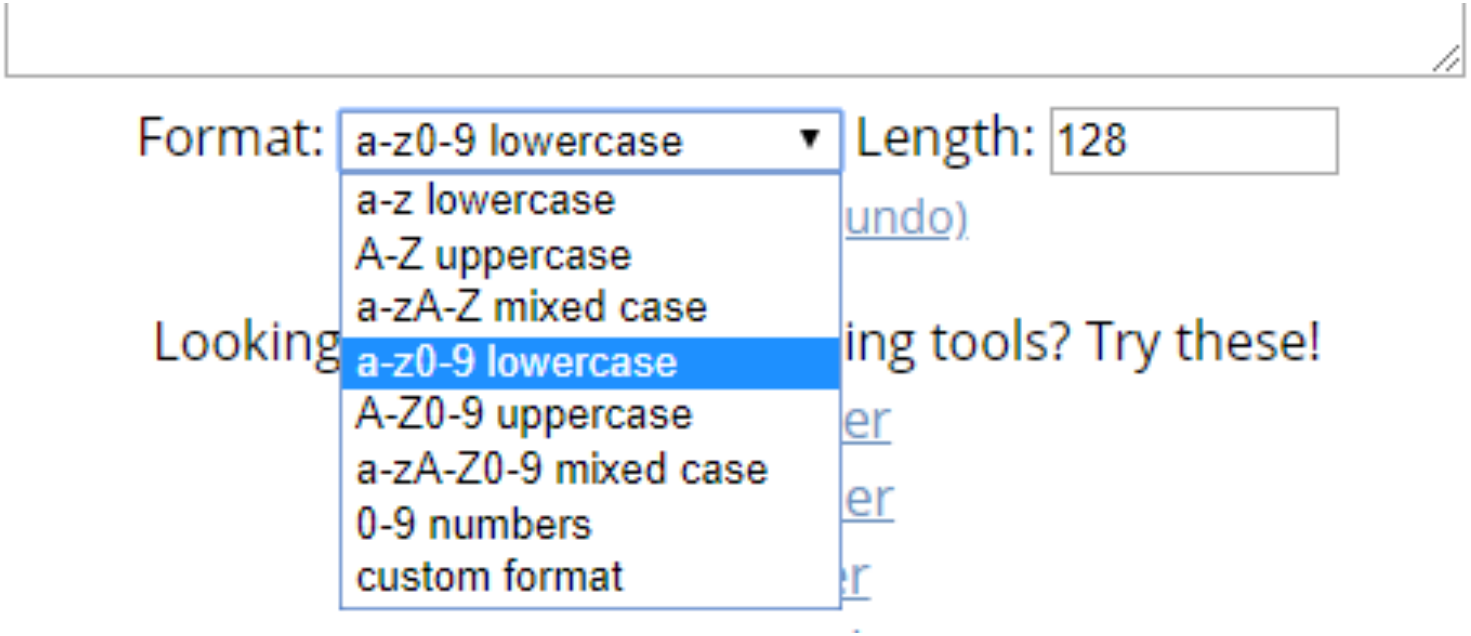
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it?

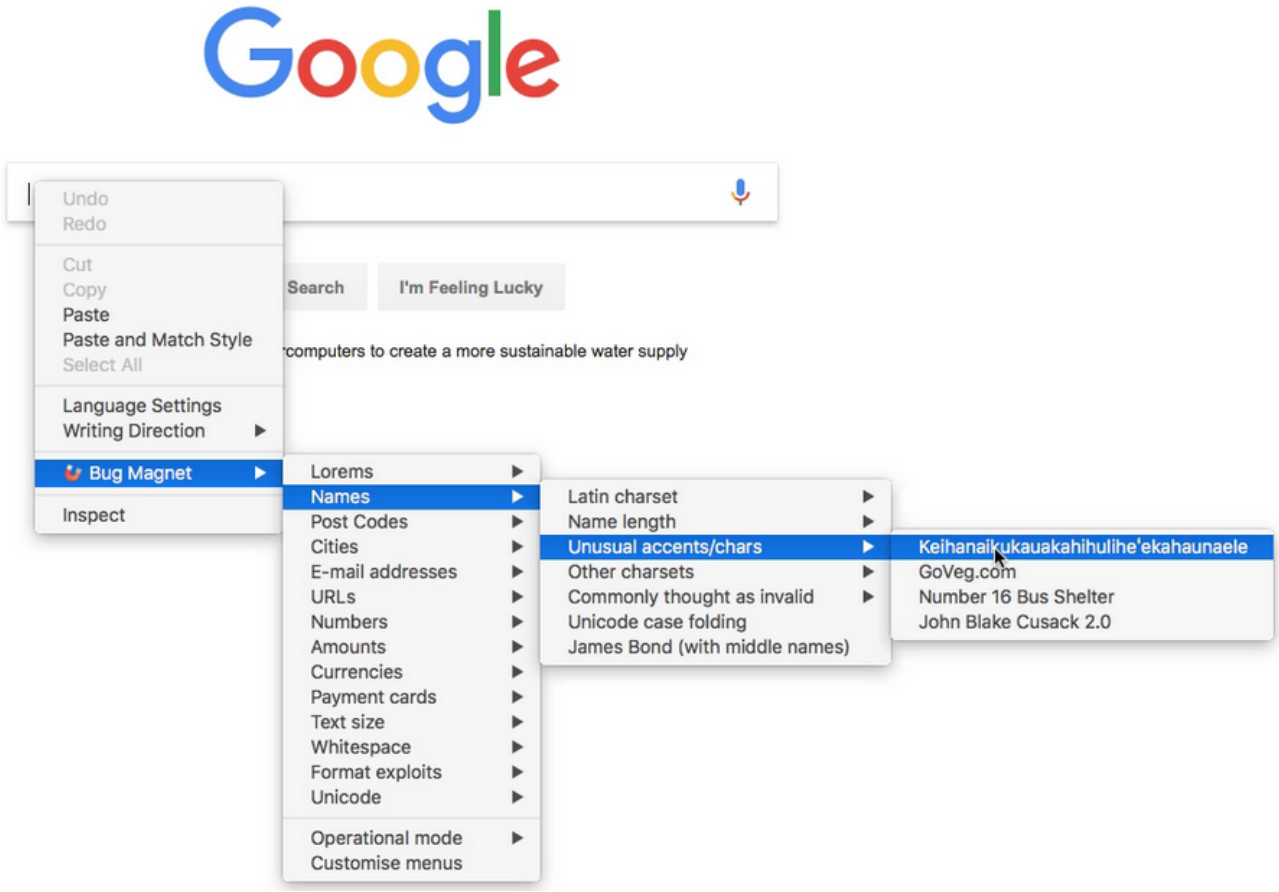
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Use String Generators

https://www.browserling.com/tools/random-string
http://www.unit-conversion.info/texttools/random-number-generator/



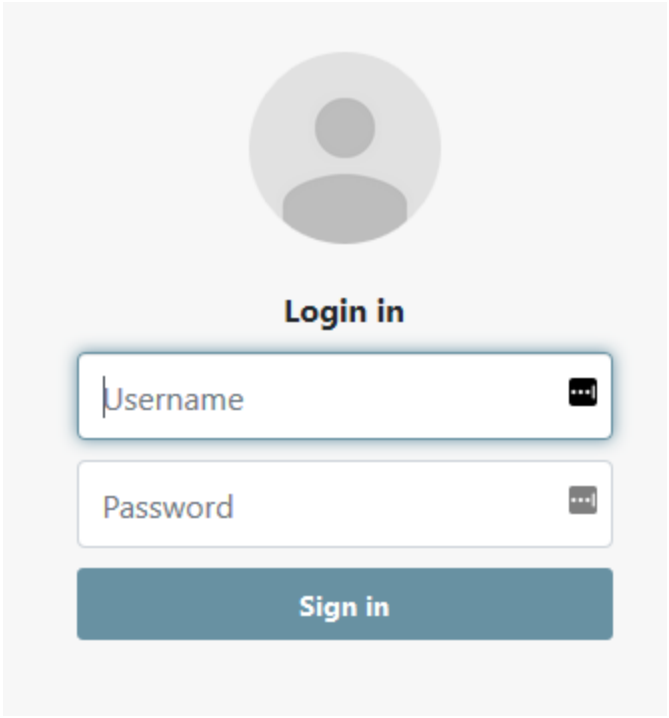
Tool of the day: Bug Magnet



https://bugmagnet.org/

BUG MAGNET

Login Form



Registration Form

Create User

Username

Email

Password

Confirm Password

Display Name

Name First Last

Nickname

Website

Bio

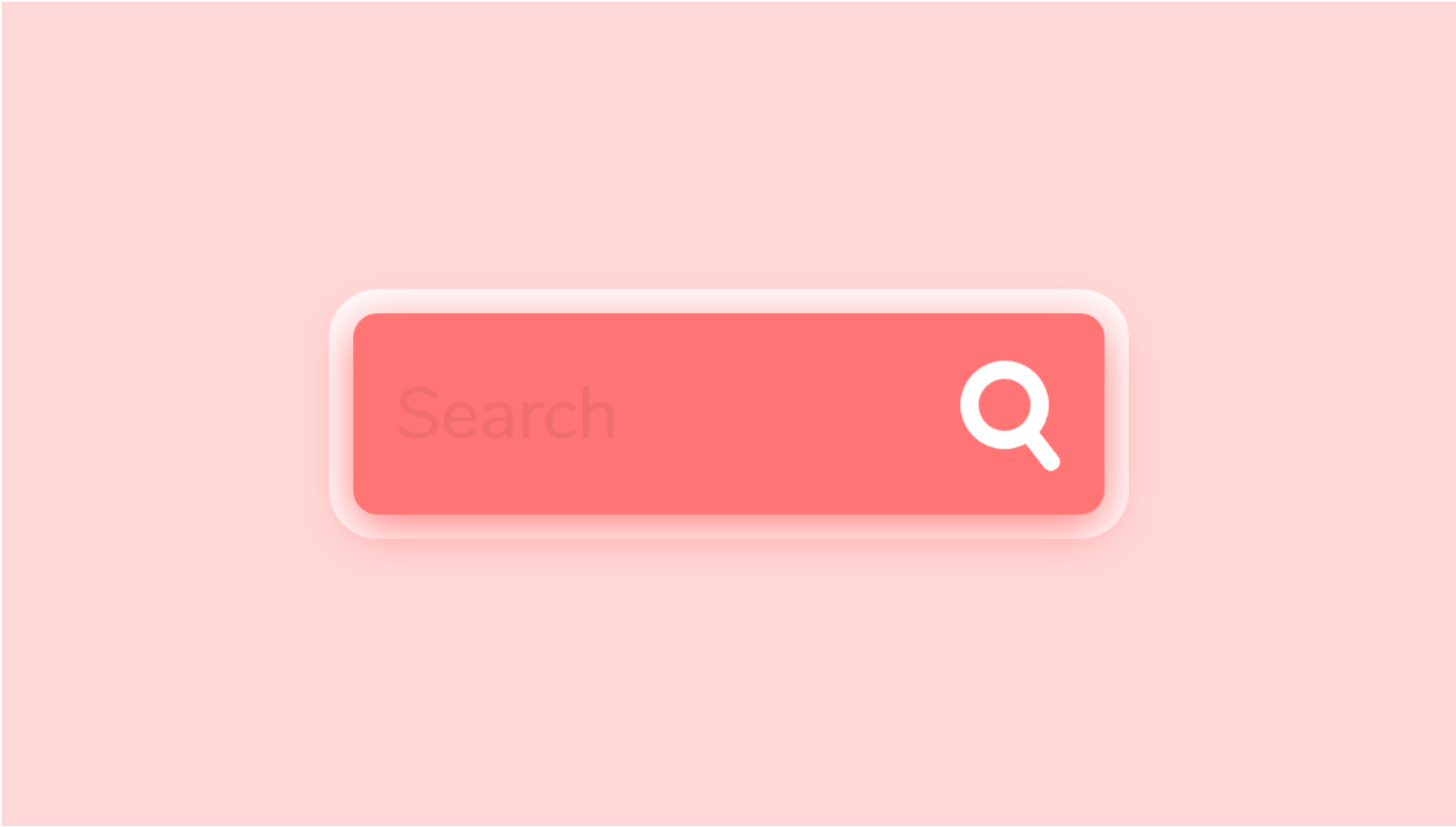
Jabber

AOL IM

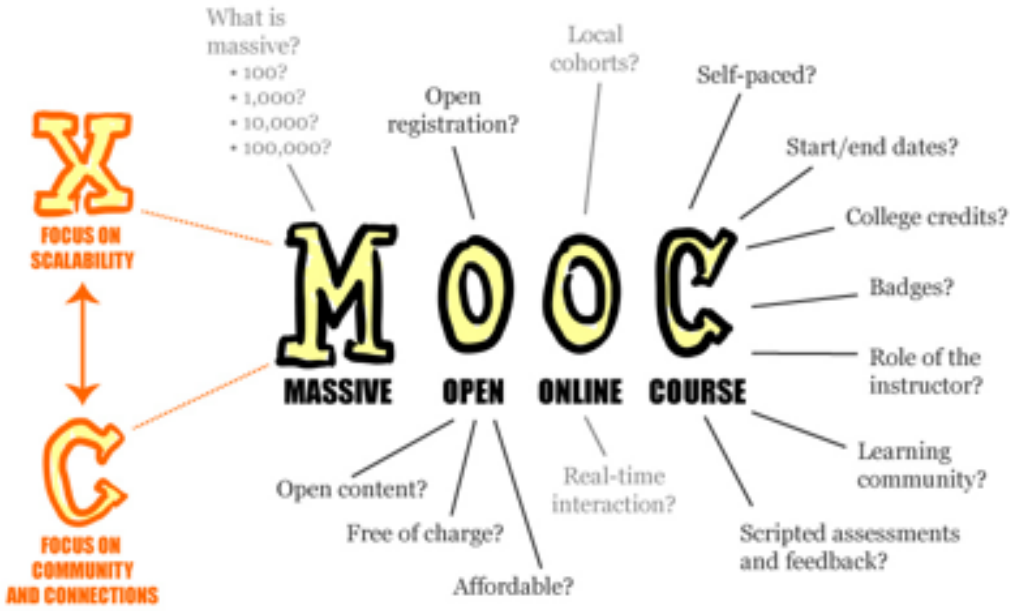
Yahoo IM

CREATE AND ACCESS

Well...



MOOCS


What is massive?
• 100?
• 1,000?
• 10,000?
• 100,000?
Open registration?
Local cohorts?
Self-paced?
Start/end dates?
College credits?
Badges?
Role of the instructor?
Learning community?
Scripted assessments and feedback?
Real-time interaction?
Affordable?
Free of charge?
Open content?
X
FOCUS ON SCALABILITY
C
FOCUS ON COMMUNITY AND CONNECTIONS
MOOC
MASSIVE OPEN ONLINE COURSE

MOOCS

Coursera
edX
Udacity
Udemy
Stanford Lagunita

https://www.class-central.com/report/mooc-providers-list/
https://knowledgelover.com/best-mooc-massive-open-online-course-providers-list/

Table of Contents

- Test Cases
- Requirements
- Use case
- Test plan
- Test reports
- Test suite
- Test scenario

On Testing

It’s not the client’s idea that goes to production, it’s the developer’s assumption that does..

Requirements

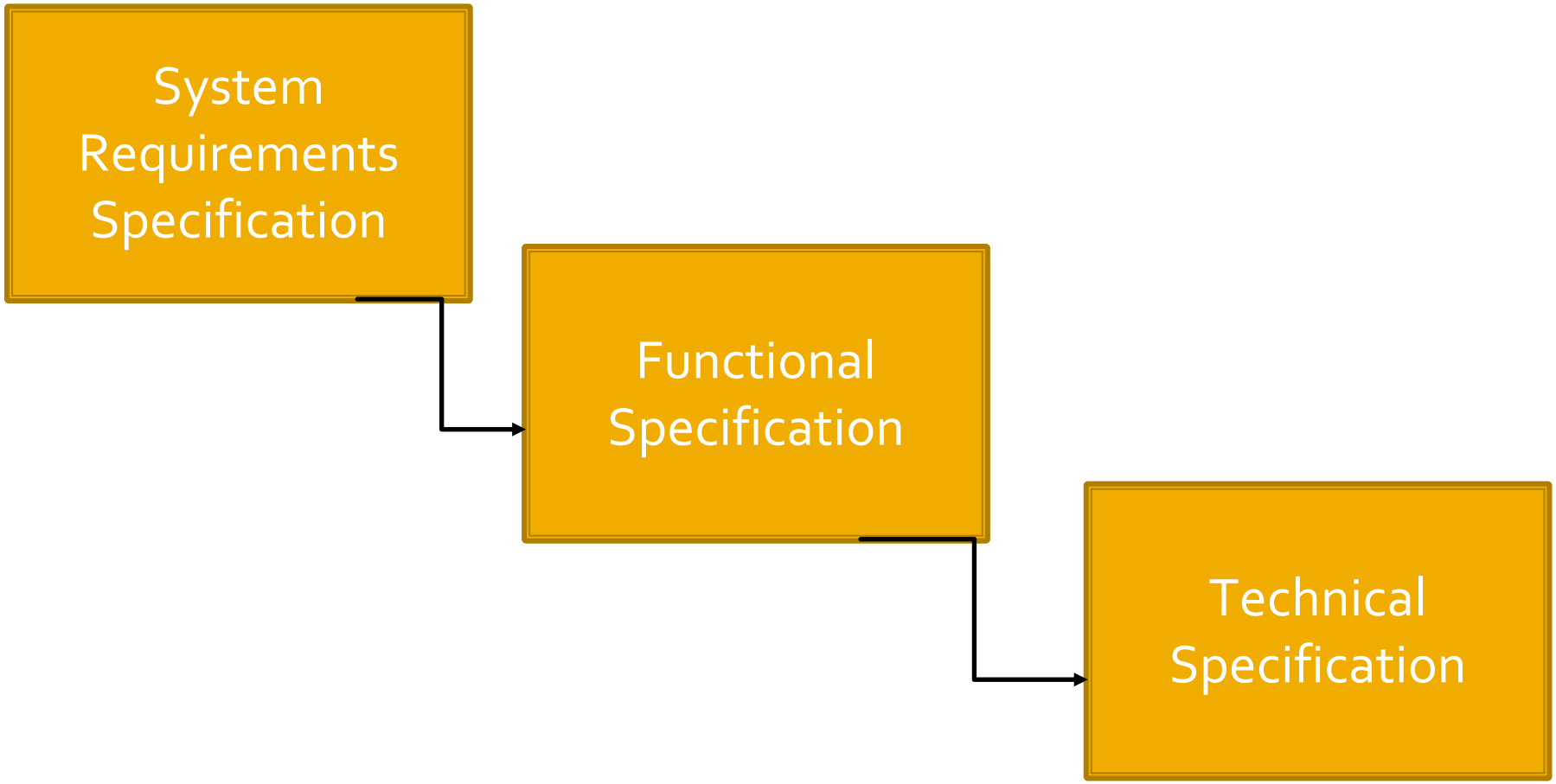
- The software requirements are description of features and functionalities of the target system.
- A **software requirements specification** (SRS) is a description of a software system to be developed.
- It lays out **functional** and **non-functional** requirements, and may include a set of use cases that describe user interactions that the software must provide.

- https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database
- https://www.bmc.com/blogs/software-requirements-specification-how-to-write-srs-with-examples/

Requirements Engineering

- The process of gathering the software requirements from the customer, analyse and document them, is known as requirement engineering.
- The result from this process is System Requirements Specification (SRS)
- https://medium.com/omarelgabrys-blog/requirements-engineering-introduction-part-1-6d49001526d3

Requirements documents



Requirement documents audience

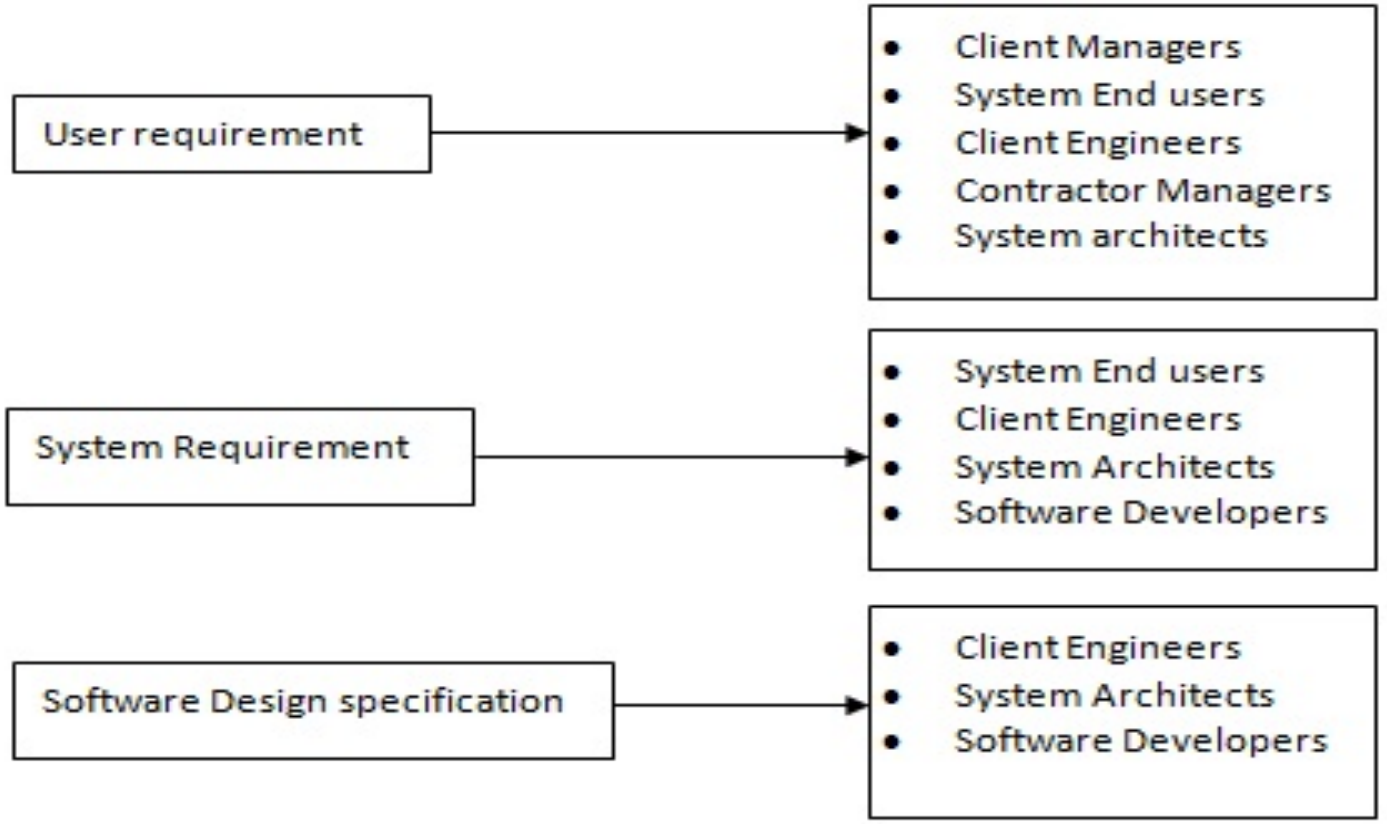


Fig. 3.1 Readers of Different Types of Specifications

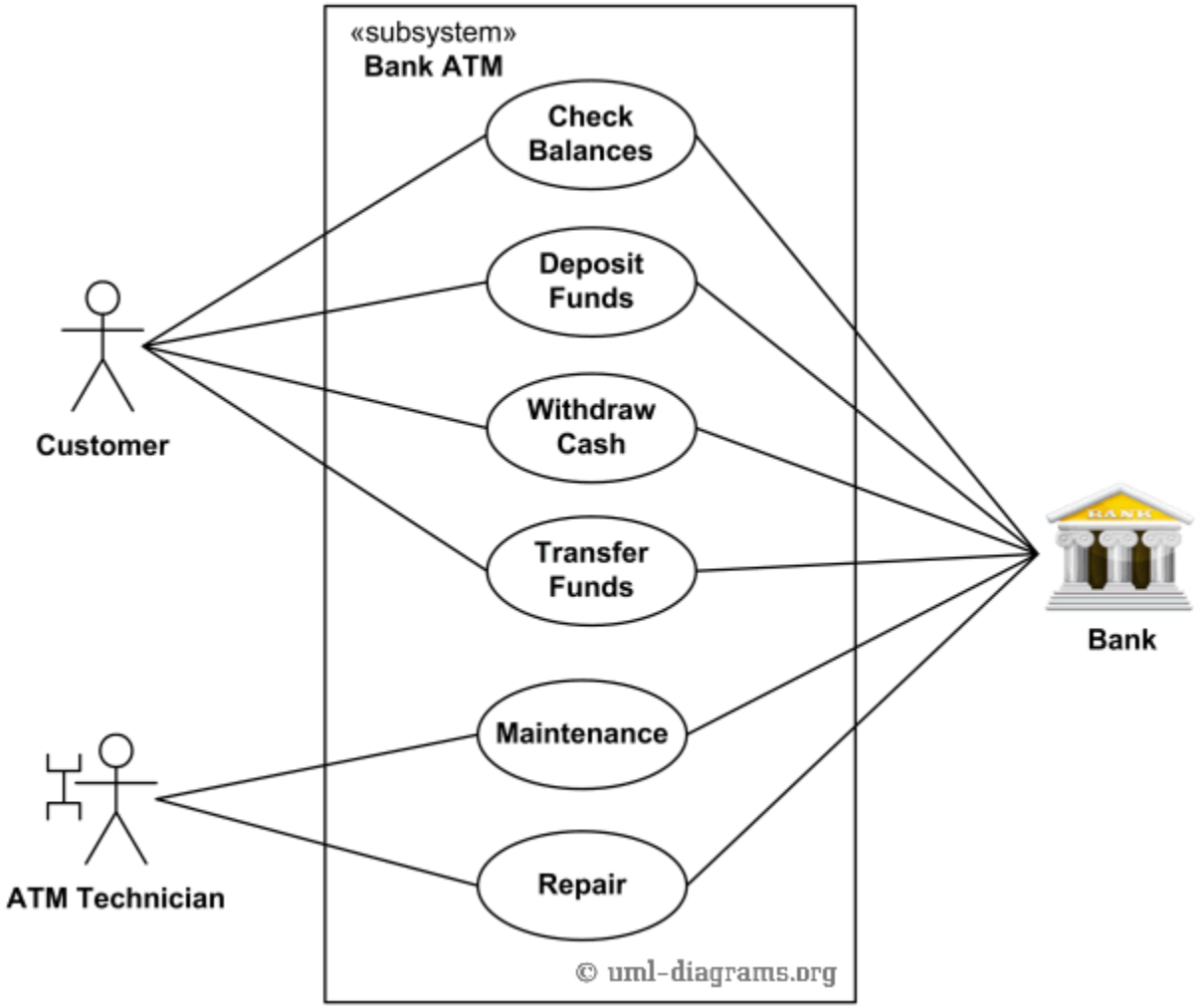
Use case

Use cases are a sequence of steps that describe the interactions between the actor and the system.

They serve as the foundation for developing test cases mostly at the system and acceptance testing levels.

https://www.projectconnections.com/knowhow/burning-questions/difference-between-use-case-and-test-case.html
http://tynerblain.com/blog/2007/04/12/use-case-vs-test-case/
https://www.softwaretestingclass.com/difference-between-use-case-and-test-case/

Use case 2



https://www.uml-diagrams.org/examples/online-shopping-use-case-diagram-example.html?context=uc-examples

Test Design Specification

The test design is the first stage in developing the tests for software testing projects.
It records what needs to be tested, and is derived from the requirements, specifications and use cases.
It specifies the test conditions for a test item, the detailed test approach, and identifies the associated high level of test cases. The main objective of this document is to specify which the test suites and test cases to run and which to skip.

http://www.professionalqa.com/test-design-specification

Test Plan

A test plan details:

- Software to test
- What and how to test
- Infrastructure required for the software to run
- Skills required to execute the test
- When to test
- To whom and what to report
- Total efforts required to complete the test
- When to stop testing and release the software.

A Software Test Plan is a document describing the testing scope, approach, resources, schedule, deliverable, communication, entry and exit criteria. It is the basis of formally testing any software/product in a project.

Test Plan 2

Test plans can be of different levels and types, depending upon the scope of testing.

Master Test Plan – Planning at organization / product level. This is usually high-level document setting standards for multiple levels.

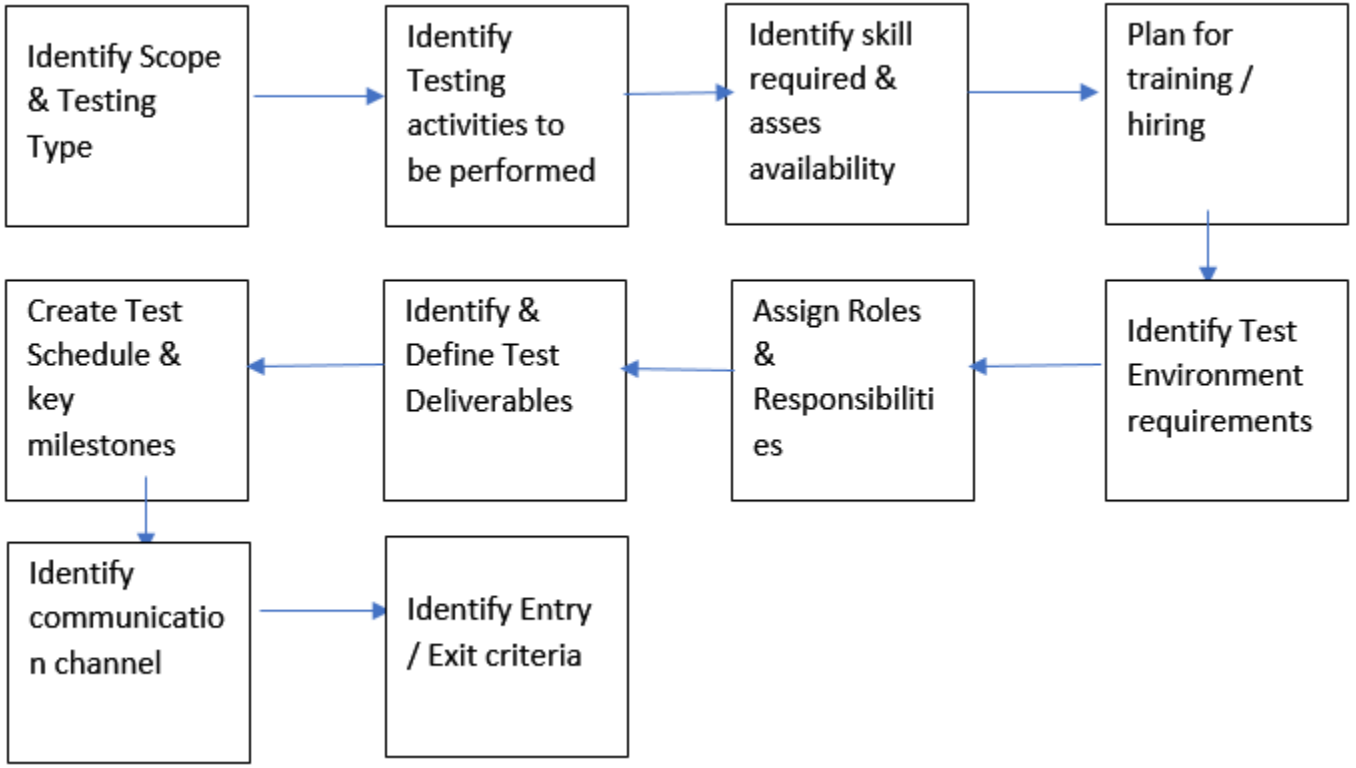
Phase Test Plan – Test plan for specific feature / interim release / phase. This plan adheres to guidelines set by the master test plan. Also with the Phase test plan, there can be test plans for specific type of testing as applicable. e.g.

Functional Test Plan – Covers functional testing of a software / phase

Security Test Plan – Covers functional testing of a software / phase

Performance – Covers performance testing of a software / phase

Test planning steps



https://www.softwaretestinghelp.com/test-plan-sample-softwaretesting-and-quality-assurance-templates/
https://www.opencodez.com/software-testing/test-planning.htm
https://www.softwaretestinghelp.com/test-plan-template/

Test Incident Report

- The Test Incident Report documents all issues (bugs) found during different test phases.
- It's the detail of the actual versus expected results of a test, when a test has failed, and anything indicating why the test failed.
- http://toolsqa.com/software-testing/what-is-an-incident-in-software-testing/

Test Summary Report

A detail of all the important information to come out of the testing procedure, including:

- an assessment of how well the testing was performed,
- an assessment of the quality of the system,
- any incidents that occurred,
- and a record of what testing was done and how long it took to be used in future test planning.

Test cases

Test cases help guide the tester through a sequence of steps to validate whether a software application is free of bugs, and working as required by the end user.

A well-written test case should allow any tester to understand and execute the test.

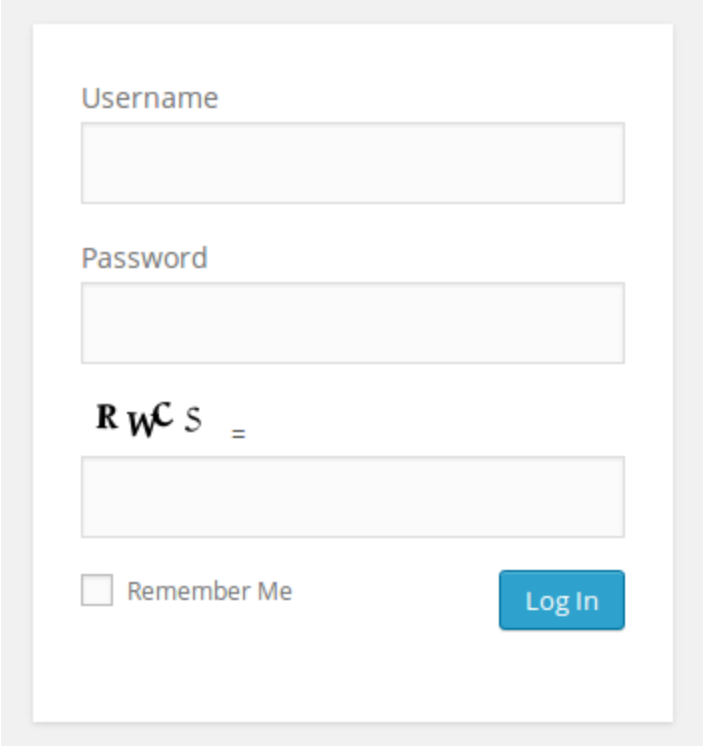
When writing test cases, it’s important to put yourself in the user’s shoes and to include all the necessary details.

Test cases 2

- A test case has components that describe an input, action/event and an expected response, in order to determine if a feature of an application is working correctly.
- Test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behaviour of the system is satisfied or not.

Test cases 3

- Go to the internal portal web site and log in to see your locker number in the employee details section.
- ???



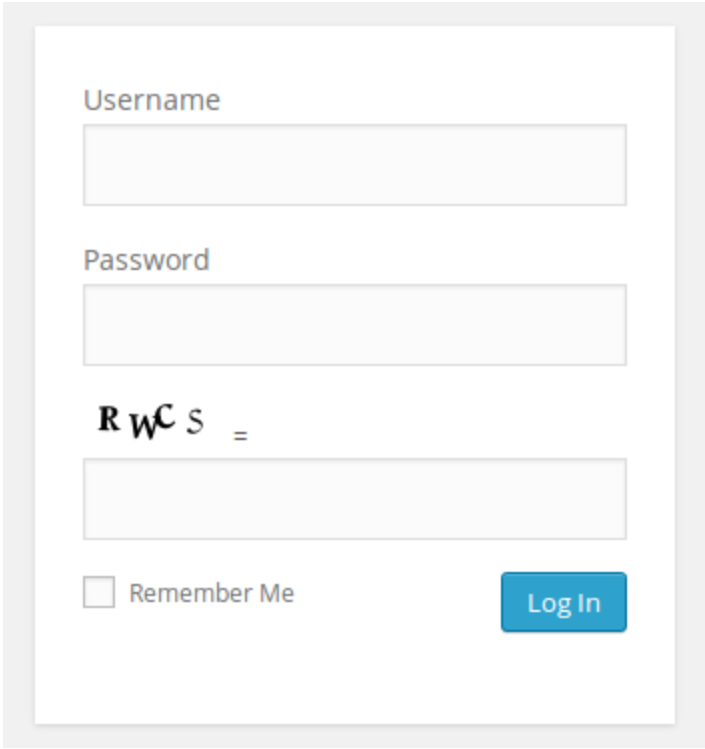
What is the address of the internal portal?

What username and password should I use?

Where is the employee details section?

Test cases 4

1. Go to portal.mycompany.net
2. Click ‘I’m registered user’ link
3. Enter your company email and domain password
4. Enter captcha data
5. Click on the Log In button
6. Go to ‘My Work Details’ link in the left section
7. Click my desk and locker number



Write test cases for the program

The specification:
1.0. Program froggy.py accepts user input.
1.1. Text of prompt for input: “What animal do you like more: frog or cat?”
1.2. If input is either “frog” or “cat”, the program must print on screen: “<user input> is a great animal”.
1.3. If user enters nothing and just presses “Enter” the program should print message: “You did not type anything”.
1.4. In all other cases the message should be “You did not enter the expected word”.

```
user_input = raw_input('What animal do you like more: frog or cat?')
#Display a prompt for text input.
animal_list = ['frog','cat'] #this is a list of 2 words one of which
is expected to be entered
if user_input in animal_list: #if user entered word that matches any
element inside animal_list
    print user_input + ' is a great animal'
elif user_input == '': #if user entered nothing and just pressed
Enter
    print 'You did not type anything'
else: #in all other cases
    print 'You did not enter the expected word'
```

Test cases 4

Test case: Payment can be made by Visa.

- "How can I do testing if I don't have a Visa card and don't know where to get one?"
- "How can I be sure that a payment was really made even if I had that Visa card?"

Improved test case:

1. Go to http://main.sharelane.com.
2. Click link "Test Portal".
3. Click link "Account Creator".
4. Press button "Create new user account".
5. Copy email to the clipboard.
6. Go to http://main.sharelane.com.
7. Paste user email into textbox "Email".
8. Enter "1111" into textbox "Password".
9. Press button "Login".
10. Enter “expectations” into textbox "Search".
11. Press button "Search".
12. Press button "Add to Cart".
13. Click link "Test Portal".
14. Click link "Credit Card Generator".
15. Select "Visa" from drop-down menu.
16. Press button "Generate Credit Card".
17. Copy card number to the clipboard.
18. Go to http://main.sharelane.com.
19. Click link "Shopping cart".
20. Press button "Proceed to Checkout".
21. Select "Visa" from drop down menu "Card Type".
22. Paste card number into text box "Card Number".
23. Press button "Make Payment".
24. Write down order id: ___.
25. Click link "Test Portal".
26. Click link "DB Connect Utility".
27. Make database query:
select result from cc_transactions where id = <order id>;

Expected result: "10"

Test cases 5

- In the credit card test case example the expected result (ER) was preceded by steps that were supposed to give the test case executor an actual result (AR). These steps can be also called the "procedure“.
- - The procedure is the instruction about input.
- - The execution of the procedure is the act of input.
- - The expected result is the expected output.
- - The actual result is the actual output.
- The test case execution is finished once we have compared the actual and expected results. This comparison can give us one of two results:
  1. PASS—i.e., AR equals ER
  2. FAIL—i.e., AR does not equal ER

Test Case Attributes

- Unique ID
- Title/Name/Summary
- Priority
- Setup(Prerequisites) and Additional Info
- Definition of expected results
- Attachments
- Revision History

Test Case Attributes 2

- **Unique ID** - The test case ID should be unique, not only within the concrete test suite which contains that test case,
but also within all test suites inside the company. The rationale for this is that, as time goes by, it will be necessary to keep test case statistics; update, remove, or move test cases between test suites, etc.

- **Title/Name/Summary** – short and clear description of the test case, so everyone who must execute that test case will immediately understand what is being tested.

- **Priority** - The test case priority reflects how important this test case is. The priority is graded from 1 to n, with 1 being the highest priority. P1 – Successful login. P4 – Background colour on the home page navigation is blue. Why we need priority? When we don’t have time we execute P1s only.

Test Case Attributes 3

- **Setup(Prerequisites) and Additional Info:**
- Information about existing user accounts or instruction on how to create a new user account.
- Data used in the test case; e.g. default password for testing.
- SQL queries (also called “SQL statements”); i.e., commands used to interact with the database (DB). SQL (pronounced "sequel") stands for Structured Query Language.
- Comments to help the tester in this specific setup and test execution

Revision History - a journal of any changes where we record: When, Who, Why and What.

- Created (date/name) – when the first version of this test case was created and who created it.
- Modified (date/name) – when any change to the test case took place and who was responsible for the change.
- Reason – why the test case was changed and what was changed.

Test Case Attributes 4

- **Expected Result:** The result that is expected after the test execution.
- **Attachments:** If any files/documents are needed for the execution of the test case, they might be attached (in case we use test case management system).

Test Case Attributes 5

Summary: description of test case idea	TC ID	
	Priority	
SETUP and ADDITIONAL INFO		
Created (date/name):	Reason:	
Modified (date/name):	Reason:	
1. Step 1 2. Step 2	- Expected result	

Test Cases - Summary

Summary: Payment can be made by Visa	TC ID	PMNT001A
	Priority	1

SETUP and ADDITIONAL INFO
Password: "1111" (this is default unchangeable password for all user accounts)
Search keyword: "expectations“ /// !!!Go to Test Portal>Helpers>DB Connect Utility to run SQL query below.
SQL1: select result from cc_transactions where order_id = <order id>;

Created (date/name): 25 Aug 2018/Pesho	Reason: New way to check if payment with card works.
Modified (date/name):	Reason:

1. Go to http://main.sharelane.com.
2. Click link "Test Portal".
3. Click link "Account Creator".
4. Press button "Create new user account".
5. Copy email to the clipboard.
6. Go to http://main.sharelane.com.
7. Paste user email into textbox "Email".
8. Enter password into textbox "Password".
9. Press button "Login".
10. Enter search keyword into textbox
11. Press button "Search".
12. Press button "Add to Cart".
13. Click link "Test Portal".
14. Click link "Credit Card Generator".
15. Select needed card from drop-down menu.
16. Press button "Generate Credit Card".
17. Copy card number to the clipboard.
……

Expected result
10

Test Case Example

Test Case Template

Test Case ID: Fun_10

Test Priority (Low/Medium/High): Med

Module Name: Google login screen

Test Title: Verify login with valid username and password

Description: Test the Google login page

Test Designed by: <Name>

Test Designed date: <Date>

Test Executed by: <Name>

Test Execution date: <Date>

Pre-conditions: User has valid username and password
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User= example@gmail.com	User should be able to login	User is navigated to	Pass	
2	Provide valid username	Password: 1234		dashboard with successful		
3	Provide valid password			login		
4	Click on Login button					

Post-conditions:
User is validated with database and successfully login to account. The account session details are logged in database.

Test Case Example

Your Company LOGO	Project Name:		Test Designed by:		
	Module Name:		Test Designed date:		
	Release Version:		Test Executed by:		
			Test Execution date:		

Pre-condition	
Dependencies:	
Test Priority	

Test Case#	Test Title	Test Summary	Test Steps	Test Data	Expected Result	Post-condition	Actual Result	Status	Notes

Automated Test Case Example

```
private FirefoxDriver wd;
/*Setup tthe driver and go the the website*/
@Before
public void makeWebDriverAndGotoSite() {
    wd = new FirefoxDriver();
    wd.get("http://etsy.com");
}
/*Teardown*/
@After
public void killWebDriver() {
    wd.quit();
}

@Test
public void adding_a_hat_to_the_cart() {
    /*Go to home page and search for "Hat"*/
    new Home(wd) {{
        searchFor().sendKeys("Hat");
        searchButton().click();
    }};
    /*We should expect a number of results*/
    new SearchResults(wd) {{
        numberMatchingSearchHeader().shouldMatch(".*\\d items.*");
        firstNonSponsoredListing().link().click();
    }};
    /*Add to card selected item*/
    new Listing(wd) {{
        addToCartButton().click();
    }};
    /*Verify that one item is in your cart*/
    new Cart(wd) {{
        numberOfItemsInCartHeader().shouldContain("1 Item in Your Cart");
    }};
}
```

Good Test Cases – Tips and Tricks

- Simple to execute (fast is good too)
- Repeatable
- Make failure obvious
- Distinguish different kinds of failures
- Easy to maintain
- Test exactly one thing

Good Test Cases – Tips and Tricks 2

- Organize test document in sections
- Cover negative cases
- Have atomic test steps
- Make sure tests are prioritized
- Make sure the steps sequence is right

Bad Practices

- Dependency between test cases
- Poor description of the steps(assumptions)
- Poor description of the expected results
- Misleading title/summary
- Wrong test priority
- Very long test procedure
- Composite steps

Bad Practices – Example

"Dependency" is the opposite of "independency." An independent test case is a test case that does not rely on any other test cases.

Test Case #1
The steps:
1. Enter living room.
2. Head for large leather chair.
3. Open **right** external pocket of backpack.

Expected result: huge beer mug

Test Case #2
The steps:
1. Enter living room.
2. Head for large leather chair.
3. Open **left** external pocket of backpack.

Expected result: potato chips

Bad Practices – Example 2

Test Case #1
The steps:
1. Enter living room.
2. Head for large leather chair.
3. Open **right** external pocket of backpack.
Expected result: huge beer mug.

Test Case #2
The steps:
1. See Steps 1 and 2 from Test Case #1
2. Open **left** external pocket of backpack
Expected result: potato chips

Do not do this! Test Case #1 may be deleted. Steps 1 and 2 of Test Case #1 might be changed… We might create new TC just for entering living room, but not rely on steps 1 and 2 in another TC.

Bad Practices – Example 3

Assumption that the previous TC has passed.
Test Case #1
1. Create a user with name “Gosho”
2. Add an email address “gosho@email.com”
Expected result: A user “Gosho” with email address “gosho@email.com” is created.
Test Case #2
The steps:
1. Change the email of user “Gosho” to “gosho1@email.com”
Expected result: the email of user “Gosho” is now “gosho1@email.com”
Test Case #2 was written with the assumption that before its execution:
- Test Case #1 had already been executed.
- The code tested in Test Case #1 doesn't have any bugs, so all data is generated in the correct way.

Bad Practices – Example 4

A composite step is the one that can be broken down into several individual steps.

Amazon.com – place an order for any product.
a. Launch Amazon.com
b. Search for a product by entering the product keyword/name into the “Search” field on the top of the screen.
c. From the search results displayed, choose the first one.
d. Click on Add to Cart on the product details page.
e. Checkout and pay.
f. Check the order confirmation page.

Which of these is a composite step?

Bad Practices – Example 5

Amazon.com – place an order for any product.

a. Launch Amazon.com
b. Search for a product by entering the product keyword/name into the “Search” field on the top of the screen.
c. From the search results displayed, choose the first one.
d. Click on Add to Cart on the product details page.
e. Click on Checkout in the shopping cart page.
f. Enter the CC information, shipping, and billing information.
g. Click Checkout.
h. Check the order confirmation page.

Bad Practices – Example 6

Poor description of the steps.

Directions:
1. Take US-101*
2. Take I-80
3. Take I-580
4. Take I-5
5. Take CA-170
6. Take US-101
7. Take exit 9A

**number of the freeway (high-speed road).*

Bad Practices – Example 7

Poor description of the steps – improved:

Directions:
1. Take US-101 S
2. Take I-80 E
3. Take I-580 E
4. Take I-5 S
5. Take CA-170 S
6. Take US-101 S
7. Take Exit 9A

This set of directions is much better, because once you approach the freeway it's clear what direction (North, South, East, or West) to take.

Bad Practices – Example 8

Poor description of the steps – this is how it should be:
1. Head north on Divisadero St toward Page St
2. Turn right at Oak St
3. Turn right at Octavia Blvd
4. Slight right at Central Fwy/US-101 S (signs for US-101 S)
5. Take the exit on the left onto I-80 E toward Bay Bridge/Oakland
6. Take the exit onto I-580 E toward CA-24/Downtown Oakland/Hayward-Stockton
7. Merge onto I-5 S
8. Slight right at CA-170 S (signs for Hollywood/CA-170 S)
9. Merge onto US-101 S
10. Take Exit 9A for Vine St
11. Turn right at Vine St

Test Cases Maintainability

Back to our credit card TC. “Login “ button can be renamed to “Sign In”. We have to change this in each TC. We can have too many small changes like this one.

Maintainability is the simplicity and ease of changing a test case to reflect changes in the software.

LOGICAL MODULE	TEST CASE STEPS
1. Create new user.	1. Go to http://main.sharelane.com. 2. Click link "Test Portal". 3. Click link "Account Creator". 4. Press button "Create new user account". 5. Copy email to the clipboard.
2. Login.	6. Go to http://main.sharelane.com. 7. Paste user email into textbox "Email". 8. Enter **password** into textbox "Password". 9. Press button "Login".
3. Find a product.	10. Enter **search keyword** into textbox "Search". 11. Press button "Search".
4. Add product to Shopping Cart.	12. Press button "Add to Cart".
5. Generate Credit Card	13. Click link "Test Portal". 14. Click link "Credit Card Generator". 15. Select needed card from drop-down menu. 16. Press button "Generate Credit Card". 17. Copy card number to the clipboard.
6. Do Checkout.	18. Go to http://main.sharelane.com. 19. Click link "Shopping cart". 20. Press button "Proceed to Checkout". 21. Select appropriate value from drop down menu "Card Type". 22. Paste card number into text box "Card Number". 23. Press button "Make Payment".
7. Get Order ID.	24. Write down order id: ___.
8. Query DB.	25. Run **SQL1**

Test Cases Maintainability 3

1. Create new user
We do have to tell tester how to create new user account using test tool Account Creator (Test Portal>Helpers>Account Creator), but why shall we put the same steps over and over again every time when new account needs to be created? Instead, what if we:
- select blocks of the steps which will be repeated in many test cases,
- put those blocks into an external document and
- put a reference to that external document in those test cases where our steps used to be.
For example: Create New Account
Create New Account is a link to the corresponding Web page located on our Test Portal. See Test Portal>More Stuff>QA KnowledgeBase>Create New Account.

Test Cases Maintainability 3

Summary: Payment can be made by Visa	TC ID	PMNT001A
	Priority	1

SETUP and ADDITIONAL INFO

Go to Test Portal>More Stuff>Qa KnowledgeBase to "See QA KB"

!!!Go to Test Portal>Helpers>DB Connect Utility to run SQL query below.

SQL1: select result from cc_transactions where order_id = <order id>;

Created (date/name): 01 Aug 2018/Pesho	Reason: New way to check if payment with card works.
Modified (date/name): 25 Aug 2018/Pesho	Reason: Modified steps to improve TC maintainability

1. Create New Account (See QA KB) 2. Login 3. Find Search Keyword (See QA KB) 4. Do search 5. Add found book to the Shopping cart. 6. Generate Credit Card (See QA KB) 7. Do Checkout (See QA KB) 8. Write down order id: _____ 9. Make DB query: SQL1	Expected result 10

Number of Expected Results in one TC

Our test case verifies only one concrete thing: "**Payment can be made by Visa**"
Let's assume that according to the document we can say that a payment with Visa is successful if—and only if—two conditions are met:
1. In the DB, the column result of the table cc_transactions has a value of "10" for the record with our Visa transaction.
2. The credit card balance is reduced by the amount equal to the amount of the payment.
So, in order to verify only one concrete thing, we have to check to see if the reality meets two expected results. We have two ways:
1. Split the test case IDEA into two IDEAs and create two test cases OR
2. Don't change the test case IDEA and have two ERs in one test case. The test case would pass if, and only if, BOTH ARs match the corresponding ERs. In all other cases, the test case would fail.

IDEA: Payment can be made by ***Visa***	TC ID	CCPG0001
	Priority	**1**

SETUP and ADDITIONAL INFO

Go to Test Portal>More Stuff>QA KnowledgeBase to "See QA KB"

In order to run SQL1 go to Test Portal>Helpers>DB Connect Utility.
SQL1:
select result from cc_transactions where order_id = <order id>;

How to get credit card balance:
Test Portal>Helpers>Credit Card Balance Viewer

Created (date/name): 11/17/2004/O.Ferguson	Reason: new way to verify that credit card transaction was successful.
Modified (date/name): 02/03/2005/I.Newsome	Reason: modified steps to improve test case maintainability
Modified (date/name): 02/15/2005/I.Newsome	Reason: modified steps and added second expected result to verify that card balance was reduced.
SL 1. *Generate Credit Card (See QA KB)* and copy card number into the temporary text file. 2. Get credit card balance and write it down: ____ 3. *Create New Account (See QA KB)* 4. Login 5. *Find Search Keyword (See QA KB)* 6. Do search 7. Add found book to the Shopping cart. 8. Copy card number from the temporary text file and *Do Checkout (See QA KB)* !!! While doing it, write down amount of payment: ____ 9. Write down order id: ____ 10. Make DB query: SQL1	✓ ***10***
11. Get credit card balance and write it down: ____	✓ Step 2 - Step 8

Number of Expected Results in one TC 2

Number of Expected Results in one TC 3

- Execute first 10. Compare the AR and ER.
- Execute Step 11. Compare the AR and ER.

The test case would pass if, and only if, both conditions are true:
1. The AR after Step 10 equals "10" AND
2. The AR after Step 11 equals "Step 2 - Step 8"—i.e., the result of subtracting the amount of payment (Step 8) from the card balance before payment (Step 2).

In theory, it would be cleaner to split the test case into two parts and create two test cases:
1. IDEA: "The correct success code is inserted into the DB if Visa is used"
2. IDEA: "The correct amount is debited from the Visa balance during Checkout"

If possible, it's better to create two test cases, BUT in the vast majority of situations, it's more practical to combine two or more ERs into one test case.

Positive Test Cases

- Positive testing checks situation where
  - The software is used in a normal, error-free way and/or
  - **The system is assumed to be sound**
- “Happy path testing” is generally the first form of testing that a tester would perform on an application. It is the process of running test scenarios that an end user would run for his use. Hence as implied, positive testing entails running a test scenario with only correct and valid data.

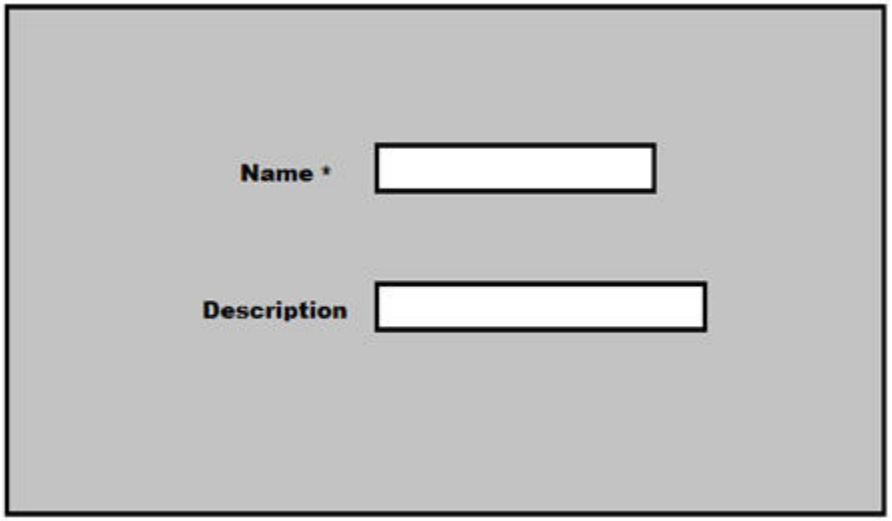
Negative Test Cases

- Negative testing checks situations that involve:
  - **User error** and/or
  - **System failure**
- 1.As a rule, **negative testing finds more bugs**. Two main reasons for this:
  a. Errors and failures can take many shapes and forms, so the PM and the programmer might not predict some of them, and thus the code will not be ready to handle certain abnormal situations.
- b. When writing and developing features, it is natural to concentrate on the normal usage and normal functioning of the software because that's how we provide value to our users.
- 2. Negative testing involves more creativity and puzzle-solving than positive testing. This happens for the same reason as above: errors and failures can take many shapes and forms.

- 3. Both negative and positive tests must be performed as a part of functional testing, but the rule is that we must execute positive tests first. Why? If functionality doesn't work during normal usage, it doesn’t really make sense to check if it works with abnormal usage.

Examples

- Enter text values in a dialog
- **Requirements:**
- The name text box is a mandatory parameter.
- The description is not mandatory.
- The name box can have only a-z and A-Z characters. No numbers, special characters are allowed.
- The name can be maximum 10 characters long.

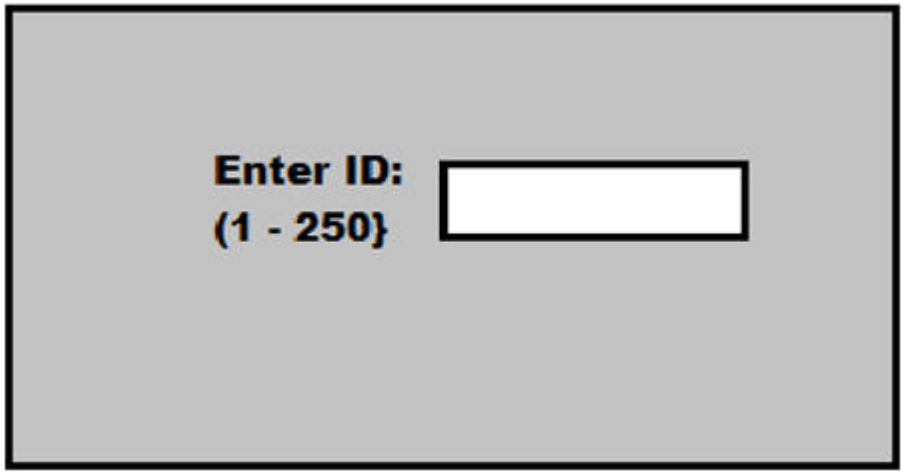


Examples 2

- **Positive test cases:** Below are some positive testing scenarios for this particular dialog.
- ABCDEFGH (upper case validation within character limit)
- abcdefgh lower case validation within character limit)
- aabbccddmn (character limit validation)
- aDBcefz (upper case combined with lower case validation within character limit)
- …
- **Negative test cases**: Below are some negative testing scenarios for this particular dialog.
- ABCDEFGHJKIOOOOOKIsns (name exceeding 10 characters)
- abcd1234 (name having numerical values)
- No name supplied
- snddwwww_ (the name containing special characters)
- …

Examples 3

- Enter numerical values in a dialog
- **Requirements:**
- The ID has to be a number between 1-250
- The ID is mandatory.



Examples 4

- **Positive test scenarios**: Below are some positive testing scenarios for this particular pane.
- 12 (Entering a valid value between the range specified)
- 1,250 (Entering the boundary value of the range specified)
- **Negative test scenarios**: Below are some negative testing scenarios for this particular pane.
- Ab (Entering text instead of numbers)
- 0, 252 (Entering out of boundary values)
- Null input
- -2 (Entering out of range values)
- +56 (Entering a valid value prefixed by a special character)

Exercise

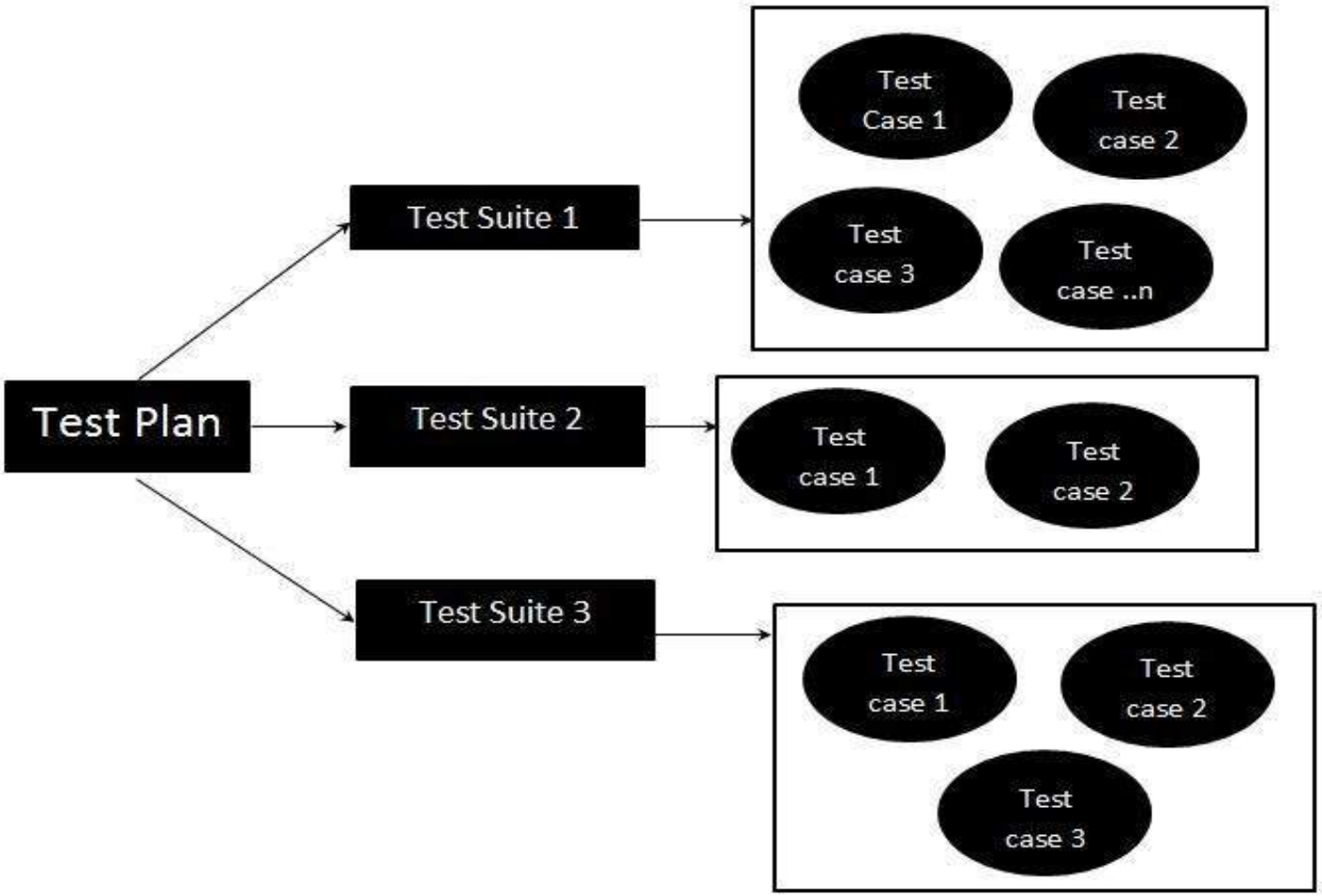
- Create test cases for login in your email and reading the last email received.
- Go to: Login page https://atanasvr.github.io/logintest/
- Spec: Login using username admin and password admin
- Take a look at the page above and check if it’s working as expected.

Test Suite /swi:t/

Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status.

A Test case can be added to multiple Test suites and test plans. A test suite allows you categorize test cases in such a way that they match your planning and analysis needs. Do you run functional and performance tests? Create two suites and label them accordingly.

Test Suite


Test Plan
Test Suite 1
Test Suite 2
Test Suite 3
Test Case 1
Test case 2
Test case 3
Test case ..n
Test case 1
Test case 2
Test case 1
Test case 2
Test case 3

Test Scenario

A set of test cases that ensure that the business process flows are tested from end to end.

The test cases in a scenario may be independent tests or series of tests that follow each other.

Example Test Scenario

Test Scenario 1: Validate the login page

Test Case 1: Enter a valid username and password
Test Case 2: Reset your password
Test Case 3: Enter invalid credentials
Test Case 4: Disable cookies and login again

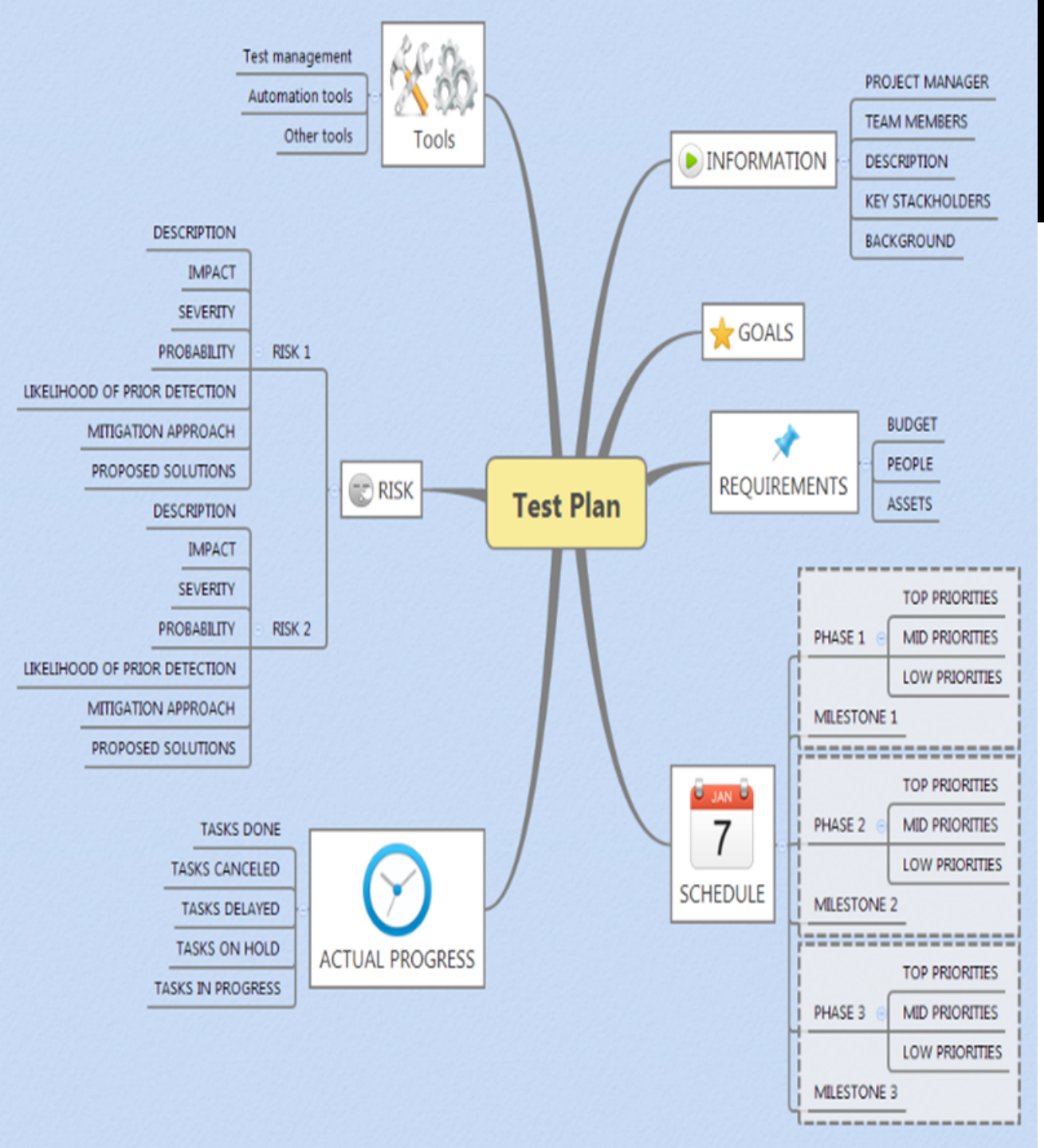
Mind Maps

- A mind map is a diagram used to visually organize information.
- A mind map is hierarchical and shows relationships among pieces of the whole.
- It is often created around a single concept, drawn as an image in the center of a blank page, to which associated representations of ideas such as images, words and parts of words are added.

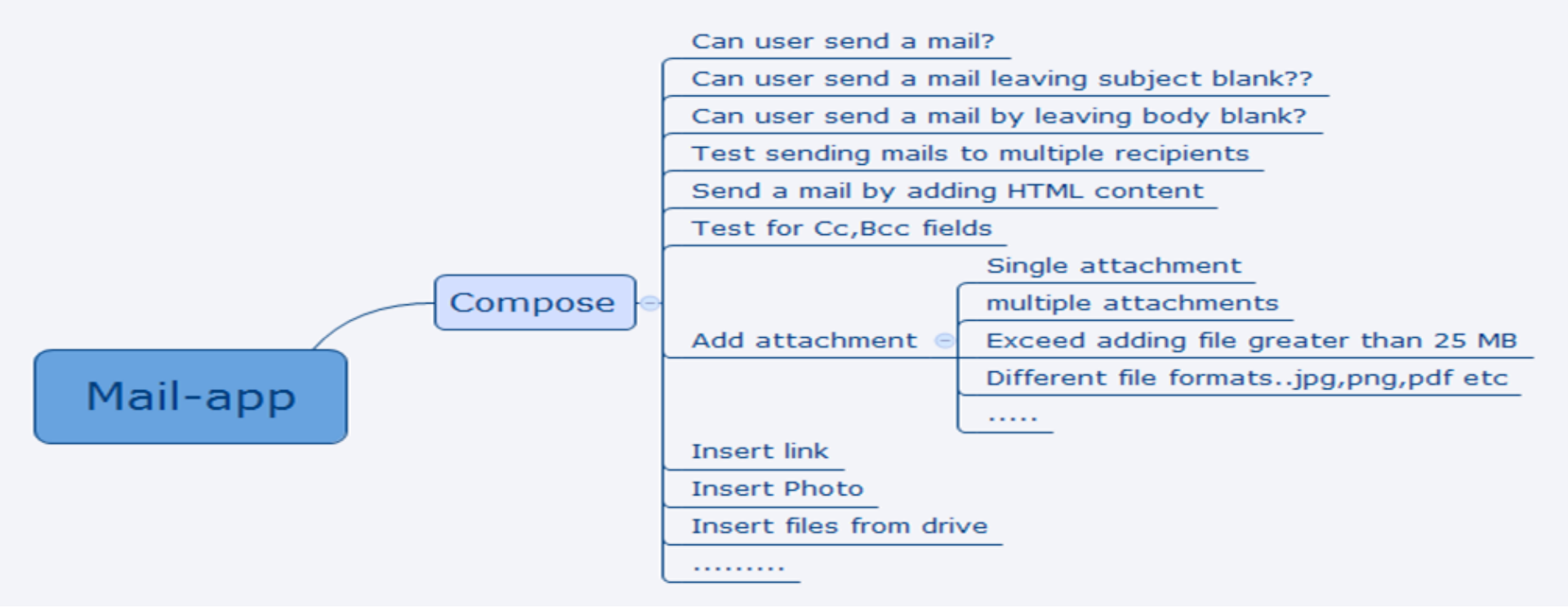
Why Using Mind Maps

- it is a creative way of structuring the data
- simple and convenient
- collects all the information in one place
- easy to change and highlight the necessary details

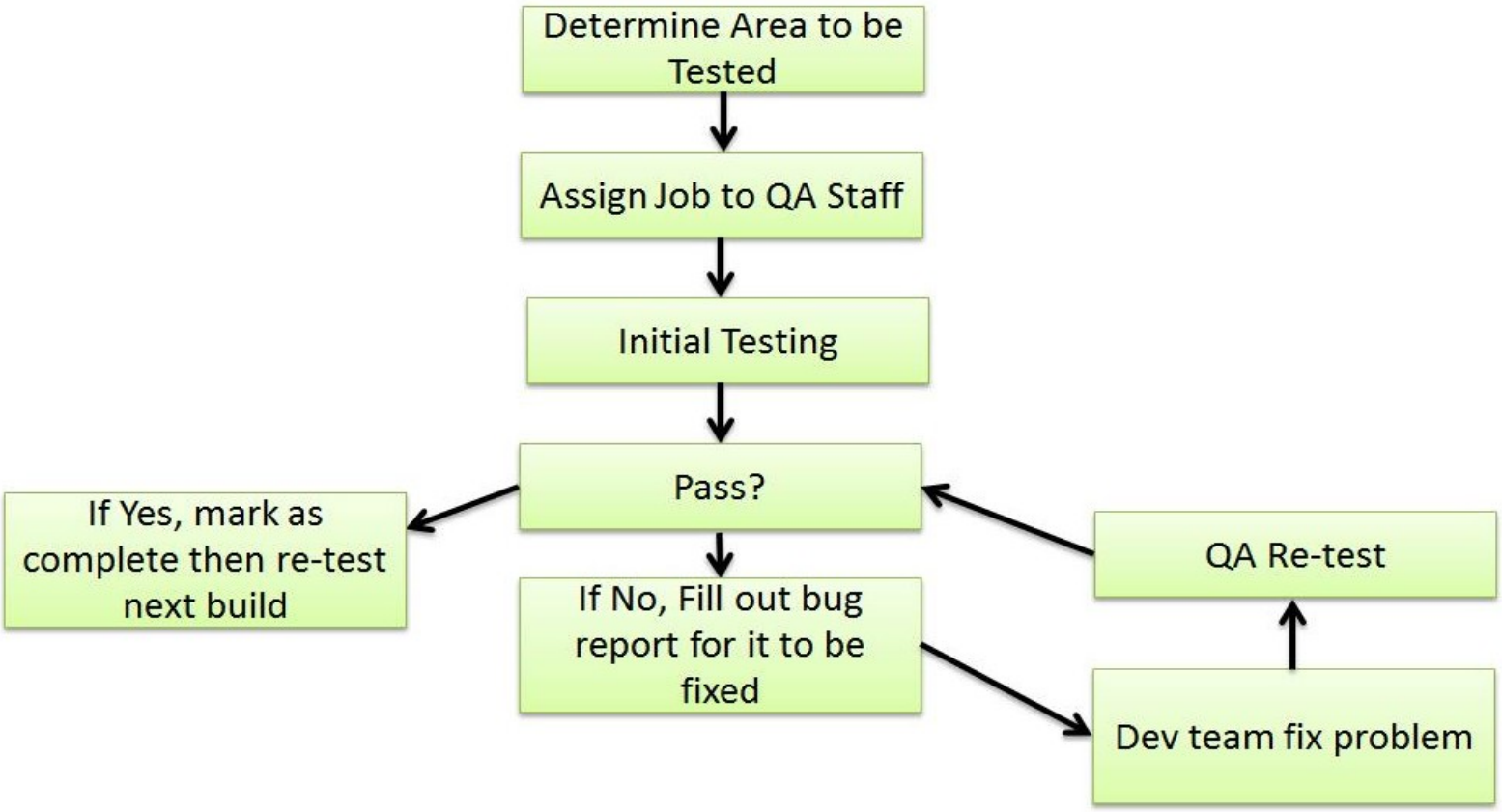
Mind Maps


Test Plan
Tools
Test management
Automation tools
Other tools
INFORMATION
PROJECT MANAGER
TEAM MEMBERS
DESCRIPTION
KEY STACKHOLDERS
BACKGROUND
GOALS
REQUIREMENTS
BUDGET
PEOPLE
ASSETS
RISK
RISK 1
DESCRIPTION
IMPACT
SEVERITY
PROBABILITY
LIKELIHOOD OF PRIOR DETECTION
MITIGATION APPROACH
PROPOSED SOLUTIONS
RISK 2
DESCRIPTION
IMPACT
SEVERITY
PROBABILITY
LIKELIHOOD OF PRIOR DETECTION
MITIGATION APPROACH
PROPOSED SOLUTIONS
SCHEDULE
PHASE 1
TOP PRIORITIES
MID PRIORITIES
LOW PRIORITIES
MILESTONE 1
PHASE 2
TOP PRIORITIES
MID PRIORITIES
LOW PRIORITIES
MILESTONE 2
PHASE 3
TOP PRIORITIES
MID PRIORITIES
LOW PRIORITIES
MILESTONE 3
ACTUAL PROGRESS
TASKS DONE
TASKS CANCELED
TASKS DELAYED
TASKS ON HOLD
TASKS IN PROGRESS

Mind Maps


Mail-app
Compose
Can user send a mail?
Can user send a mail leaving subject blank??
Can user send a mail by leaving body blank?
Test sending mails to multiple recipients
Send a mail by adding HTML content
Test for Cc,Bcc fields
Add attachment
Single attachment
multiple attachments
Exceed adding file greater than 25 MB
Different file formats..jpg,png,pdf etc
.....
Insert link
Insert Photo
Insert files from drive
.....

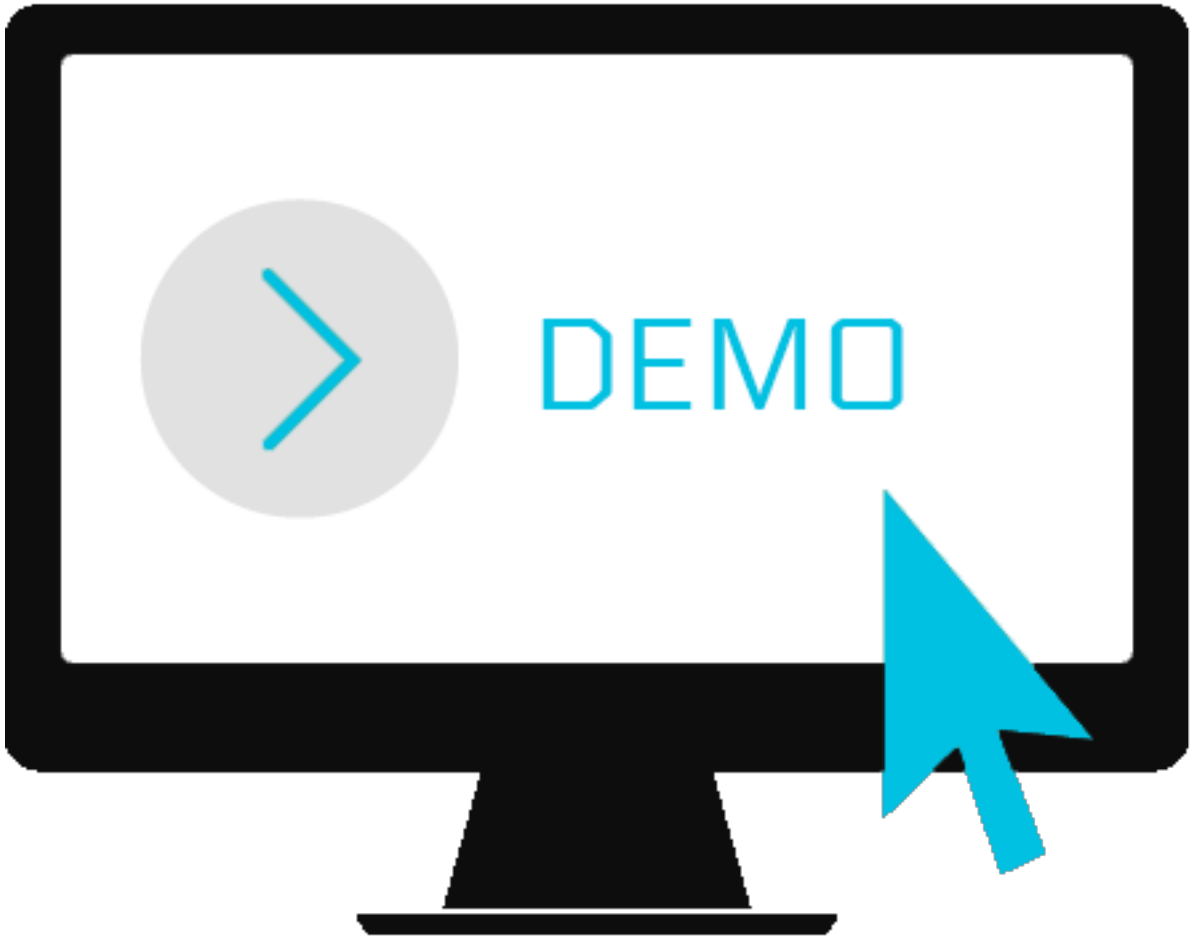
Test Case Planning And Execution Process



Test Management tools

1. Creating and maintaining project cycle/component information
2. Creating and maintaining the test artifacts specific to each release /cycle that we have- requirements, test cases, etc.
3. Establishing traceability and test coverage
4. Test execution support – test suite creation, test execution status capture, etc.
5. Metric collection/report generation for analysis
6. Bug tracking/defect management

Demo



Nonfunctional Testing

Reading

- https://www.allaboutrequirements.com/2011/10/test-cases-based-on-use-cases.html
- https://blog.testlodge.com/how-to-write-test-cases-for-software-with-sample/
- https://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/
- https://geteasyqa.com/qa/best-test-case-templates-examples/
- https://blog.qatestlab.com/2019/06/06/mind-maps-testing/
- https://lisacrispin.com/2011/02/28/using-mind-maps-for-test-planning/
- http://www.apps.testinsane.com/mindmaps/