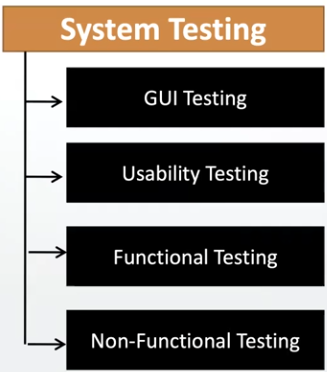
**System testing types**

**Системное тестирование** означает тестирование всей системы в целом, оно выполняется после интеграционного тестирования, чтобы проверить, работает ли вся система целиком должным образом. В основном это тестирование типа «черный ящик», которое оценивает работу системы с точки зрения пользователя с помощью документа спецификации и оно не требует каких-либо внутренних знаний о системе, таких как дизайн или структура кода.



System testing focuses on

**1-User interface testing (GUI).**

-Element Check: test if buttons, text boxes, drop downs, radio buttons and other web elements has the right size and in the correct place on the screen, resolution of the screen, image quality, spelling check, links (they must be in a special color and change the color when pointing the cursor to it)

**2-Usability testing** - насколько интуитивно понятно приложение, легко ли установить, пользоватся. User manuals should be easy to read.

**3-Functional testing** - is nothing but behavior of application. (**имеется в виду разная функциональность программы,например: как login работает, как send email работает,как проверка баланса работает и т.д. т.е. как работает тот или иной функционал)** проверяется каждая функция, путем ввода значений и проверки вывода: соответствует ли функция функциональным спецификациям, то есть совпадает ли фактический результат с ожидаемым от этой функции.

**Functional testing types:**

**a)-Object Properties testing**.

Object it is for example, an input field. property for this object might be - it can be enabled or disabled. Or for example, an object can be a drop down which contains multiple options, and the ability to choose one or multiple options is the property of that drop down. Or for example, an object is a link and the ability of that link to change the color when you direct the cursor to it is the property of that object and to redirect to another page is also the property of that object. Или например в анкете много input-ов , ты ввёл что-то нажал ентер и автоматически произошло переключение на след. инпут - это тоже properties.

**b)-Database testing (or backEnd testing)** - Мы используем UI программы для ввода какой то инф. и смотрим появилась ли эта инфа в б.д. или например в б.д. уже есть какая то запись и мы её апдейтим через UI, соответственно проверяем б.д., чтобы всё было изменено соответствующе, или удаляем через UI проверяем б.д. чтобы было удалено. или хотим получить (ретрив) инф. с б.д. запрашиваем через UI, проверяем бд что нам вернулось корректное значение.

**c)-Error handling testing**

Например ввёл не правильно имя юзера в этом случае должна появится ошибка говорящая причину что было сделано некорректно. И надо чекать что вообще такие error-message есть, а они должны быть и если их нет то это баг. И они должны быть понятными и легко читаемыми. Может быть :

-error message

-warning message

-info message

**d)-Calculations/Manipulations testing**

**e)-Links testing.** When testing links there might be different scenarios. One link may lead to the correct destination, another may lead to the wrong destination. If the link is broken what type of message should we expect and so on…There are a 3 types of links:

- Internal links - it is when the target of the link is located on the same web page For example you click on a link and it redirects you to the section at the bottom on the same page.

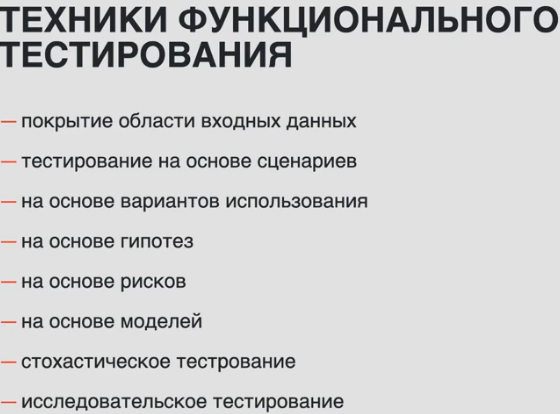
- External links - it is link that redirect you on a different web page.

- Broken links - when you click on the link you will get file not found exception or page not found error, forbidden page, page not available. When you see that type of message it means that target page is not available on the server.

**f)-Cookies & Sessions testing**

Cookies - temporary files created by Browser while browsing the pages through the internet and stored locally.

Session - for example you logged in but do not do any activity on the page, and then when you start do activity, the page may ask you to log in again.



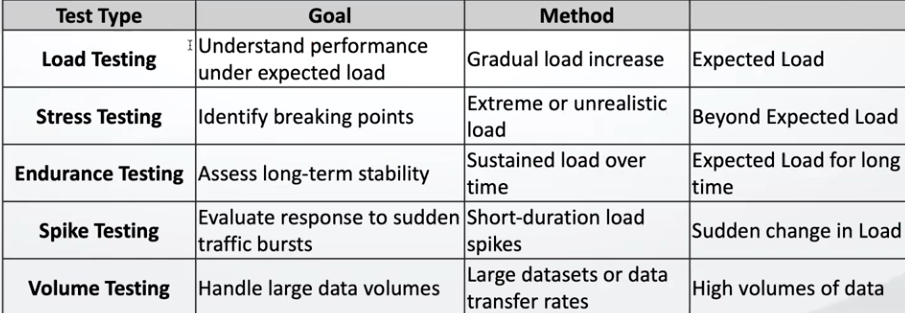
**4-Non-functional testing** (когда функциональное тестирование выполнено), - это тестирование не связанное напрямую с функционалом самого приложения. Например перформанс аппликейшна - как быстро приложение отвечает на те либо другие запросы, security of the application, application recovering from abnormal conditions or not, recovery testing, stress testing, compatibility testing…).

To perform some of the non-functional testing we need a special environment (performance,security) We can not perform functional and non functional testing in the same environment.

Most of the non - functional testing can be done manually except performance and security.

**1)Performance Testing** (**производительность**)- before doing performance testing we need to know the parameters: how many users access the application per day, how many users access the application simultaneously in one time. What are the days that will have the most traffic? How much data it is supposed to handle in a period of time (DB size).

**different types of performance testing:**

****

**-Load Testing** -how many concurrent users are accessing the application, for example one day it is 50 users, another day it is 500 users, another day it is 5000 users. The amount of users is the load for testing, we will create for example 50 virtual users and we will try to access the application with that expected load in one time. That is load testing, we need to simulate the users activity. The difficulties is that we need to create not only virtual users but all of them should have different environments (OS, browsers…). The tools for that: Load runner, J meters,...

So once we have got the needed amount of users what are we going to measure?

Response time - how much time it takes to get the response from server.

turnaround time - request time + response time (сколько времени пройдёт от начало запроса до ответа-наверно)

throughput - how many requests server processing in 1 second. Why? Because if an application uses 100 users in one time, the server can not process all of them at the same time, it will follow certain algorithm (for example - Fi Fo). But all of that goes with expected load - the amount of users that we expect to use the application.

**-Stress Testing** - test the performance of the application with unexpected load (lets say expected load for app is 300 users, so unexpected load we will perform with 400users), we need to identify where is the point that application is braking. And to find the breaking point we need to gradually increase the load, so we will see when it happened exactly. So stress is beyond the expected load. Technically it is done with the same tools as the Load testing

**-Endurance testing** - the time and stability are tested under expected load. Проводится на expected load, how much time our application is sustaining(поддерживающий), how much time the application is stable. Берут ожидаемое количество юзеров и смотрят как долго апликейшн может работать стабильно при такой нагрузке. Другими словами load testing - тест на силу приложения, сколько оно может потянуть юзеров, а endurance testing это тест на выносливость приложения.

**-Spike testing** - increasing the load suddenly, beyond expected lod.

**-Volume testing** - how much traffic (volume) of data, application is able to handle. Focus on a DB.

**2)Security Testing** - we will focus on a few aspects:

Authentication means users are valid or not valid. In ather words - (proving that it is you). For example, using a **username** and **password** to get into your email account. It's kind of proving who you are.

Authorization - you are the valid user, and it is ok, but what are the permissions you have? it means getting limited access to resources. For example theris 2 folders with photos. 1 folder is for public use and 2 folder is for private use only. So other people are only **authorized** to get your photos from public folder #1. They have limited access to resources.

Network security

Volunarability

Data Encryption & Decryption some sensitive data should not be visible, it should be masked. Data that is sent to the server or back should be encrypted. Because the only place that is not safe in a client - server architecture is the network. Server will decrypt data.

**3)Recovery testing**  - testing the ability of the system to recover after failure and it should save data the user have been workin on.

**4)Compatibility testing** (**совместимости**)-

OS - whether the application is compatible with os or not (it happened some time that some application is working only on windows, or Linux or both)

Browser compatibility - whether the application is compatible with specific type of browser (cross browser testing)

Hard Drive compatibility - как аппликейшн работает на разных машинах с разными процессорами, оперативкой и т.д.

**5)Configuration testing**

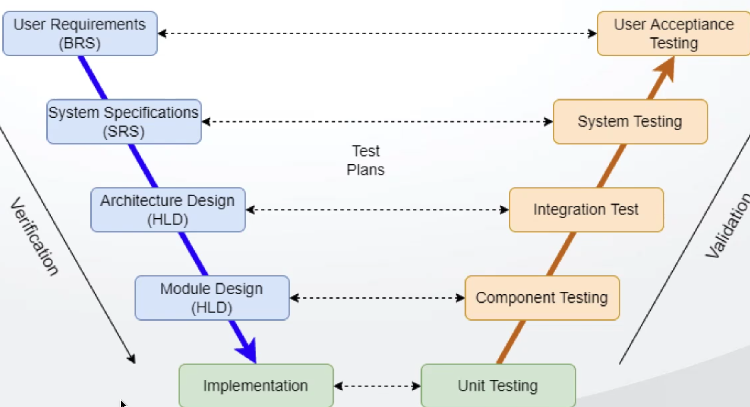
**6)Installation testing -** whether the application is installing properly or not. Screens should be clear to understand, the information for navigation should be easy and simple to understand. The un installation also should be easy and simple for a not experienced user, чтобы не было такого что мы установили затем удалили, затем снова устанавливаем а у нас ошибка что такие-то файлы уже существуют. Также апдейт с разрешения юзера.

**7)Sanitation / Garbage testing** - try to find extra features in application that have not been in the requirements and if such features are presented we need to consider them as bags.

**Поочерёдность тестирования согласно V - модели.**

Dynamic testing starts with unit testing and it is done by developers. Then component testing, (component consists of units) it tests components separately and similar to unit testing it is done by developers. Then comes the integration testing (when several modules/components are ready they need to be checked on how they work/communicate together) it is done by developers.

After that the entire software will be combined as one single unit/final software project. On that final project we will do **system** testing and this system testing is done by **testers**, the other phases are done by developers. Then we will deliver software to the customer and the software will be tested in the customer environment - user acceptance testing (UAT).

****

We do not need programming skills in order to do System testing, it is black box and we are testing the software.

All system testing we will do at the UI level (whatever we can see on the browser)

- Testing all functionality of the application with respective client requirements.

- It is black box testing technique.

- Conducted by a testing team (not developers).

- started after compilation of: unit testing, component testing, integration testing. Only then we start system testing

- We should know the customer requirements before starting system testing.