# Exercise: Test Levels and Test Types

<https://quiz.softuni.bg>

Тества 1:00ч ->

JS. Със MongoDB -> съ здва потебители и вкарва ги в базата после ги показава !!!

Пуска Get I Post заявки !!!

Problems for exercises and homework for the ["QA Fundamentals" course @ Software University](https://softuni.bg/courses/qa-fundamentals-internal).

## Levels and Types

Arrange the labels in the appropriate boxes.

UnitTesting/ComponentTestring

Integration Testing

SystemTesting

AcceptanceTesting

LEVELS

Functional Testing

NonFunctional

RiskBased Testing

ChangeRelatingTesting

TYPES

Acceptance Testing

Risk-Based Testing

Non-functional Testing

Integration Testing

System Testing

Functional Testing

Component Testing

Change-Related Testing

///

Functional Testing >

testing against the funcationalities given as requirements by clients requirements.

NonFunctional 🡪 testing answering the question How , What is the behavior of our SoftWare->Secure, Performance(производителност), Usability(потребилитлите могат с лекота да го ползват), Riliability(дали работи 24ч, 7 дни в седмицата, 365 дни в годината , тестване за надежност, за разчитаемост. Да не прекъсва сайта, интернета ....)

////-----------------------------------------------------------------------------------------------------------

Софтурени дефекти : Виж следвата лекция и го пейстни това ::

Инцитенти, грешки, рералично поведение на софтуерът от отчакването,

Очакваме Юзерът да се регистрира, а той не може да се ригистрива .

Грешките. Резултират в дефекти, Фаултс, Бъгс/,

Грешки може да им и в спецификацията също.

Ако клиентът се натъкне на дефект -> това е Failure /

Errors -> people made errors, .human mistakes are errors,

This error is defect into the program.

But this defect is not afailure because the client does not encounter /meet him.

**ERROR(made by people)**

**- DEFECT(error existing into the software)-**

**>Failure(found defect into the software)**

**Our aim as programmer is to find all defects to azvoid the failure.**

Camouflage bugs-> unfixed not foud bugs,

Static testreing-> against the requirements, specifications.

DesignSpecification is made by the

businessAnalyst from the clients requremsnts.

4 неща срещу което можем да логваме дефекти са :

1.Код, Системата, програмта,

2.Рекуайръмънтс,

3.ДизайнСпецифиакацията

4. Насоки, Упътвания към Юзерът заработа.

Дефекти можем да логваме и срещу дадените тестовае к.сме наравили,

Инцидентен доклад: 20та минута !

------------------------------------------------------------------------------------------------🡪

Итеративни модели за работа -> Скръм, Канган, Ъджайл, Лиин Модел->Това което е излишно не го прави.,

Тестерът-Оценява качеството на Софтуерът.

ТестПроцес :

ТестАнлиз и Планинг,

ТестДезйнИмплемантация и Екзекюшън,

ТестКложър Активитис, ТЕстРепортинг

///

Девелъпмънт-> цикли, и

Тестови Нива : 4 ;

UnitTesting/ComponentTestring

Integration Testing

SystemTesting

AcceptanceTesting

**Тестерите взимат най-голямо участие във Системното Тестване !!!**

ЮнитТестинг-тестване на най-базово ниво, на метод, най-малката единица ,к можем да тестваме, от програмистите, и се прави в изолация, ->грешна логика, код, проблем с данните, .Тестване и на класове.

Трислойна Архитектура

ФРонтЕнт, Бекенд, БазиДанни,

ФРонта-менюъл Тестеи,

На Бекенда и на Дроната са АутомейшънитеТЕстери,

Интеграционно тестване се прави и от дев и от Тестерите /

Идеята на интегррационното тестване е , изтесвали сме еното юнитче, изтествали сме и другото юнитче и ги тестване сега заедно дасли мога да раборят свързано в интеграция . Че всички компоненти фуньионират правилно !

Бекенд –бизнес Логика и Базата. Тестване интеграционно .

Integration testing int the small -> test two methods

In tergration testing in the large -> two systems ; beckend and DB //or our System Appication to communicate with another System .

Вкараме си Крединта катрта в нашето приложение, иданните се изпращт до друга ситема която пеоверява дали картата е активана и товага разрепава плащането.

Automatic tests we have on Unit, Integration System Level,

PostMan се използва за интерграционно API тестване .

Изикванията на Клиента да ги направи на Функционални Изиквания,

АксептансТестинг –Се праваи от Клиентът / Тества основните бизнесЦвенарии,

Може да се прави и от ПродуктОунерът .

ПродуктОунер -> човек който е връзка межву Дев Екипът и Клинетът Поръчител на проектът

ProductOwner -resemles BusinessAnalyst -> Меджинно ниво между Клиентът и Девовете.

АктептансТестинг може да се прави от Краините потребители -<>БетаТестване .

Може да се прави от друг екип във фирмата .

АксептанТестинг -> за кукита, правни изисквания,

ЮзерАксептансТестване -> Бета Тестване .(Аксептанс тестване правено от Юзерите, крайноте потребители

Аксептанс Берта тестването е БЕТАТестване, БлекБоксТестване !)

ВернаКутия.

ЮнитеТЕстването е WhateBoxTesting- > Ние Виждаме Кодът.

ТестПирамида !! >

Най-много имаме ЮнитТестове, По-Малко Интеграционни,

И най-малко ЮАйТестове.

Continuous Delivery /Deployemnt,

Каот правиш нещо ново, автоматично да се изпълняват тестовете

Operational Acceptance Testing ->дали докумнтацията която имаме е достатъчна за да функционира софтуеерът ок . Упътване за функционаране на софтуерът ни. Как да гпо инсталираме и ползваме .По дукоментация за софтуерът да проерим дали работи както е записано в докумнетацията .

ТестТипове:

**Фунционално, НеФунционално и Свързно с Промени !**

**//Example BankSoftware Testing**

**Check functionally ->if can You transefer money from user 1 to user 2 /**

**The decreased amount from one user is correct to the increased amount to the other user.**

**Check if automatic bill payments are made on time, paymt on water, electricity, on due Date…**

**Check if interest are calculated as it must be;**

**FunctonalTesting -> if the given feature is working -> functionality of payment , BinaryTesting !**

**Functional testing is binaty -> is true or false, if it works or no, no middle situation is possible.**

**The money is transferred or the money is not transferred !**

**The functional testing answers the question what should to our system !!!!**

**Functional Testing includes BlackBoxTesting, // We check the functional requirements .**

**Non-FunctionalTesting, testing that is pointed to security of the system**

**How reliable, how fast is working our system. How good is made ?**

**Testing shoul show some error conditions, -> to throwe some errors for the users..**

**Да имаме поне 1 Тест Кейс за всяко изикване,**

**1 Requrement-> one testCase at Least !!**

**Функцинално тестване на кола -> да ме превози от точка А до Точка Б. /Какво прави, воази ни.**

**Нефукнционално тотестванве-> добре ли вози, Сигурен ли е, ако се блъсна ще оживея ли, колко арчи, колко е бърз, колко ми е удобен. Как, колко добре,**

**Нефункционално тестване : видове -> Юзабилити, Ефишънси, Портабилитъ оф тхе продукт.,**

**Тестване за подържане на софтуерът, Тестване за преместване от една система .**

**Перформанце тестинг-> колко бързо става дадена заявка.**

**ЛоудТестинг-? Когато увеличавам броя на потребители които ползавт напият сайт, Тестване на натоварване.**

**Стрес Тестване -> акво ще стане ако нашата сиастема бъде натоварена изключително много над очакваното. Примерно нотмалното натоварване 1000 потребителя,**

**Тествеаме с 1000000 потребителя,. Поведението, й?**

**Много голямо натоварване, крашва ли, некрап ли**

**Волюм Тестване -> когаот 1000 потребителя вкарват данни в базата->**

**Тестване за Сигурност- >SQL Injection, special symbols,**

**Stability testing/Reliability testing -> интернетът в къщи ->ще работи ли без прекъсване всеки ден по 24 Ч.**

**Надеждно тестване.**

**Тестване за здравина > доколко системата приема грешни входни данни и колко бързо се възстановява .**

**Компитабилити Тестване-> може ли приложението ни да работи еднакво донре на различни браузъюри и операционни системи,**

**Нефунционалните изисквания не са описани в изискванията !!!!**

**Бързина ? за еди едно е бързо за друг е бавно .**

**Как тестваме Обено и Лоуд Тестване -> Тестовият софтуер има опция за симилиране на натоварване със 100, 10000 , 1 00000 потреебителя /Позлваме вуртуални Юзери –Симулираме много потребители!**

**JMeter -> tool за ПерформансТестване !**

**Retesting- > когато сме открили дефект и когато дефектът след това е поправен, тестваме отново да видим дали дефектът е премахнат. /Потвърдително тестване /Ретестването е потвърдително тестване/**

**Regression testing -> При промененеа добавена функционалност тестваме изцяло поведенеито за да видим в резултат на промяната сали всичко работи, и дали нещо не е счупено в резултат на добавката промяната, нещо друго не е засеганто и дали всичко работи.**

**Наново изпълняване на всички тестове за да се установи дали промяната не троши кодът.**

**Автоматизират се регресионните тестове !**

**Отново и отново тестСуйтата да пускаме. За да сме убенедни че след каквито и промени да са добавени кодът работи. Добавяме нова фукционалност, добавя ме и нов тест кейс, и пукаме отново тест суитата за да се уверим че всичко работи и всички тестове написани до моментът се изпълняват .**

**Преди работи кода, добавя се нова укционалност и регресинното тестване го пускаме за да видим след добавката сдали пак работи. Регресионното тестване не откриа нов бъг. Регресионото тестваен е ретестване на предишно работеща програма.Регресионното тестване може да се пари на всички нива .**

**Регресионното тестване –Техники -> РитестВсичко имаме 1000 теста, пускаме ги,**

**Правиме модификация -> пускаме пак 1000 теста.**

**Или Пускаме само спечифични тестове за този модул.**

**Или пускаме само приоритизрани тестове за всичките функционалности,**

**Обем на тестването.**

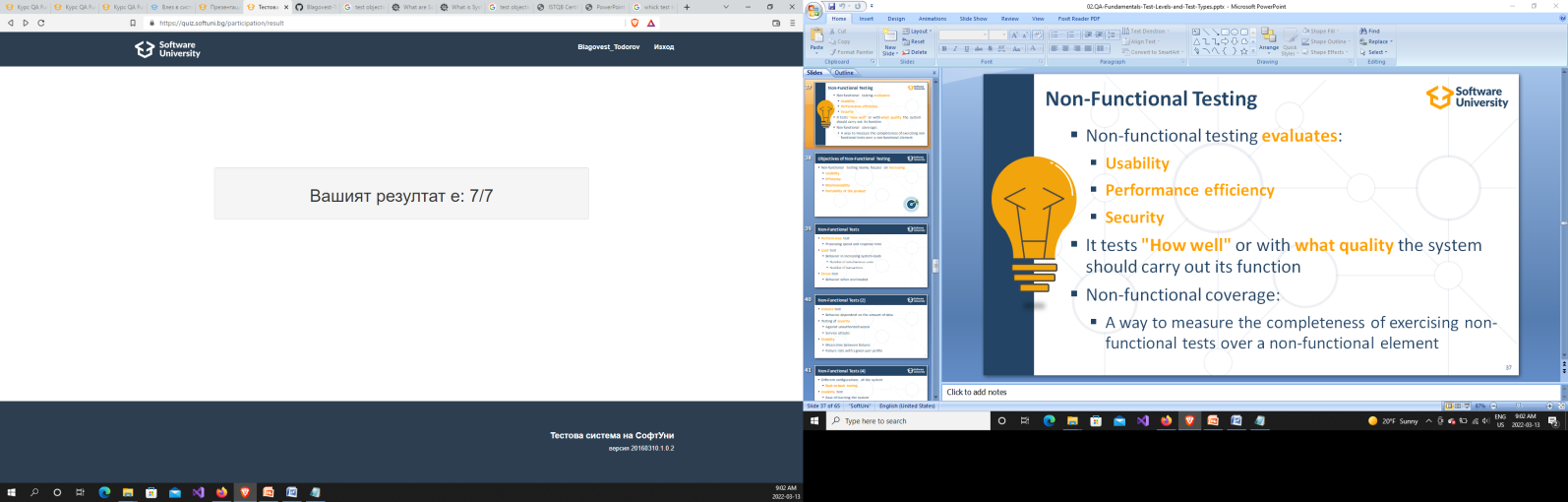
**МейтенансТестване -> // Тестване по подръжка, всяко ново или променено нещо се тества, в основа та мейтенанс Тестването стои регресионното тестване.**

**Предали сме на клиентът софтуерът, той, казва тук не работи тази финкционалност- Оправям ея, или изтриваме тази функционалност, или добавяме нова .**

**Тогава извършваме цялостно тестване.**

**Дори и софтуерът ни да не е променен, а е променена средата, трябва да направеим тестване !**

**H\**



<https://www.softwaretestinggenius.com/istqb-certification-exam-sample-papers-q-641-to-650/>

///

The main intent of smoke testing is not to perform deep testing but to verify that the core or main functionalities of the program or the software are working fine. Smoke testing aims to reject a badly broken build in the initial stage so that the testing team does not waste time in installing & testing the software application.

Smoke testing is also called as **Build Verification Test**.

Let’s see a simple example where you are given an email application to test. The important functions would be logging in to the email application, composing an email and sending it, right? And, in case the email is not sent, does it make any sense to test other functionalities like drafts, deleted messages, archives, etc? This means you will have to drop the build without further validation. This is called Smoke testing.

The main focus of smoke testing is to test the critical areas and not the complete application.

## ****When to perform Smoke Testing****

* When developers provide a fresh build to the QA team. A fresh build here means when the build has new changes made by the developers.
* When a new module is added to the existing functionality.

<https://testsigma.com/blog/smoke-testing-vs-sanity-testing-vs-regression-testing-explained/?fbclid=IwAR2XCQVgmCLE8Sc01ogc1wOCd_UfRFJVRgVIw-fP4bnubGwHVzFad8JqLO0>

///

## ****Differences Between Smoke vs Sanity vs Regression Testing****

|  |  |  |
| --- | --- | --- |
| **Smoke Testing** | **Sanity Testing** | **Regression Testing** |
| Performed on initial builds | Performed on stable builds | Performed on stable builds |
| To test the stability of new build | To test the stability of new functionality or code changes in the existing build | To test the functionality of all affected areas after  new functionality/code changes in the existing build |
| Covers end to end basic functionalities | Covers certain modules, in which code changes have been made | Covers detailed testing targeting all the  affected areas after new functionalities are added |
| Executed by testers & sometimes also by developers | Executed by testers | Executed by testers, mostly via automation |
| A part of basic testing | A part of regression testing | Regression Testing is a super set of Smoke and Sanity Testing |
| Done usually every time there is a new build | Planned when there is not enough time for in-depth testing | Usually performed, when testers have enough time |

///

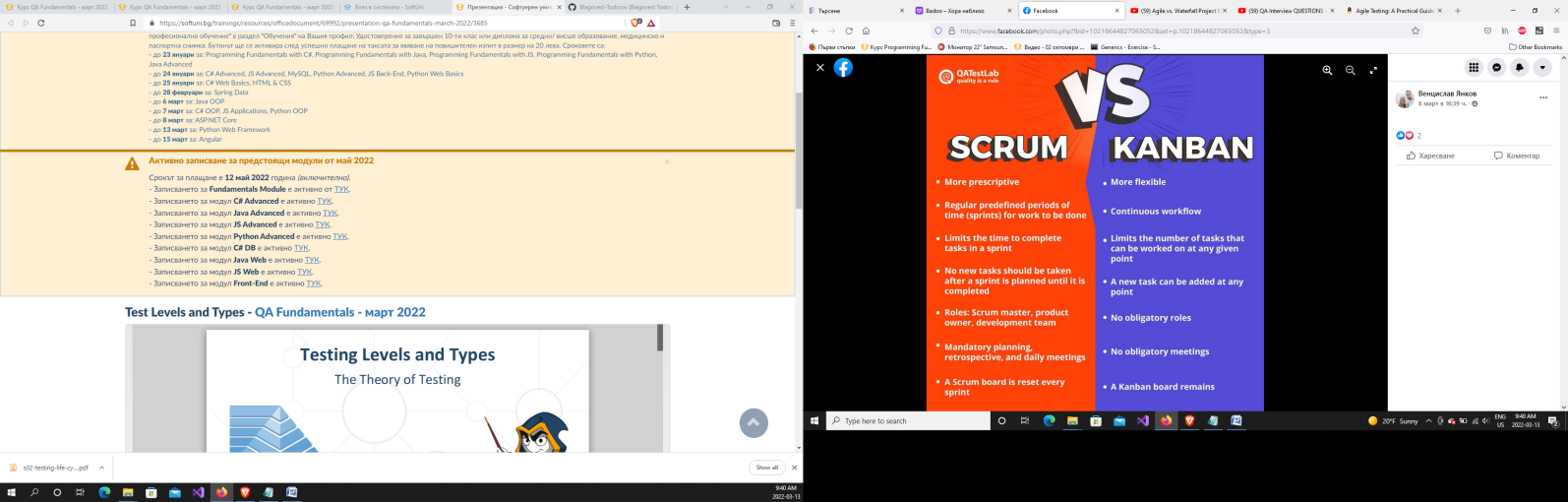
QA Tesstring questions :

<https://www.youtube.com/watch?v=P87IIE5ds_w>

Agile :

<https://www.youtube.com/watch?v=x-nX23hCwRE>

<https://www.youtube.com/watch?v=x-nX23hCwRE>



//

<https://www.youtube.com/watch?v=Z9QbYZh1YXY>

Agile –set of values and principle ->

<https://www.youtube.com/watch?v=9TycLR0TqFA>