

5.

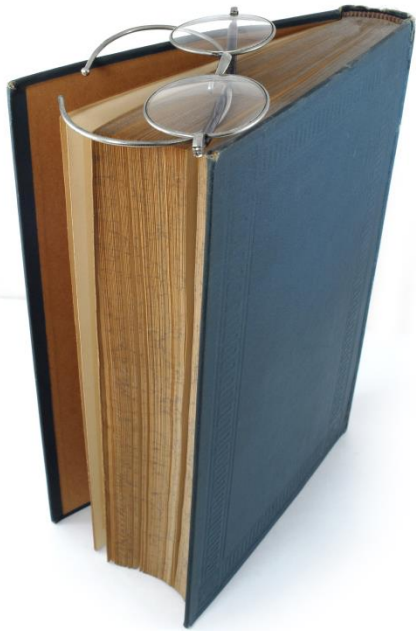


Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice

What is a Test Plan?

A **document** describing the scope, approach, resources and schedule of intended test activities. It identifies the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and any risks requiring contingency planning.





WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

[Interaction](#)

[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

[Tools](#)

[What links here](#)
[Related changes](#)

Not logged in [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

[Article](#) [Talk](#)

[Read](#)

[Edit](#)

[View history](#)



Software test documentation

From Wikipedia, the free encyclopedia

Software test documentation is the vital element that raises any experimental activities to the level of a **software test**.^[1] International organisations like IEEE and ISO have published standards for software test documentation.

Contents [hide]

- [1 Status of IEEE 829](#)
- [2 Background to IEEE 829](#)
- [3 Documents Required by IEEE 829](#)
- [4 Use of IEEE 829](#)
- [5 References](#)
- [6 External links](#)

Status of IEEE 829 [\[edit \]](#)

Note: **IEEE 829-2008** has been superseded by [ISO/IEC/IEEE 29119-3:2013](#).^[2]

IEEE software life cycle

SQA – Software quality assurance [IEEE 730](#)
SCM – Software configuration management [IEEE 828](#)
STD – **Software test documentation** [IEEE 829](#)
SRS – Software requirements specification [IEEE 830](#)
V&V – Software verification and validation [IEEE 1012](#)
SDD – Software design description [IEEE 1016](#)
SPM – Software project management [IEEE 1058](#)
SUD – Software user documentation [IEEE 1063](#)

[V](#) · [T](#) · [E](#)

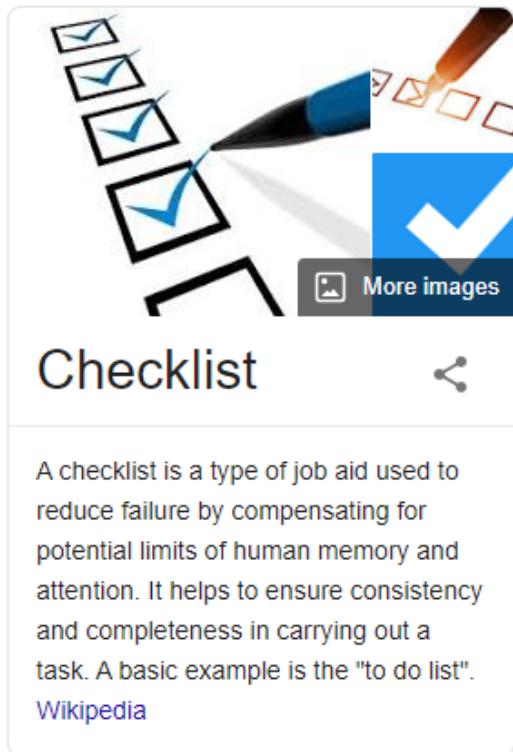
IEEE 829 test plan structure [\[edit \]](#)

[IEEE 829-2008](#), also known as the 829 Standard for Software Test Documentation, is an [IEEE](#) standard that specifies the form of a set of documents for use in defined stages of software testing, each stage potentially producing its own separate type of document.^[1] These stages are:

- Test plan identifier
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Suspension criteria and resumption requirements
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice



Regional Specification				
Date and time is properly formatted for target region				
Phone number formats are properly formatted for target region				
Colors are appropriate for the target market and express the desired message				
Licenses and product names obey country-specific regulations				
Provided phone numbers are accessible by the users in the target market				
Currency conversions and formats are handled properly				
	Pass	Fail	N/A	Notes
Language				
Terminology is consistent across the UI, help files and documentation				
Text is free of grammatical mistakes				
Text is properly translated				
Text is free of character corruption				
	Pass	Fail	N/A	Notes

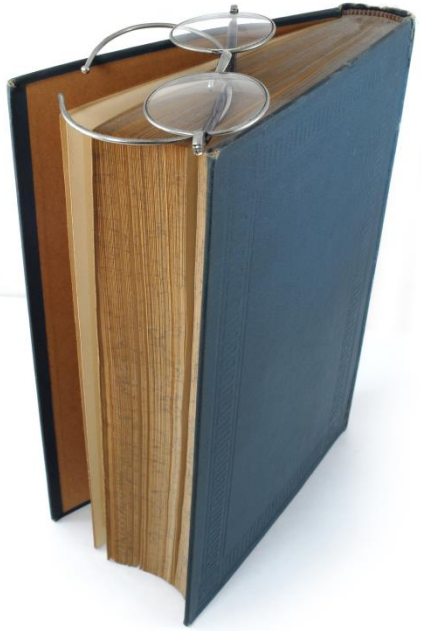
What is a Checklist?

- **Checklist** - is a list of tests which should be run in a definite procedure. It helps to understand if testing is fully run and how many failed. It also helps formalize testing separately taken functionality, putting tests in a list. Test order in the checklist may be strict as well as random.

Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice

What is a Test Case?



A **test case** is a document, which has a set of test data, preconditions, steps for reproduce, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

How to write a good test case 😊

<https://www.facebook.com/seen.everything/videos/1440544456077890/>

Test Case Template

Mandatory Fields:

- **Test case ID**: Unique ID is required for each test case. Follow some convention to indicate the types of the test. **For Example**, 'TC_UI_1' indicating 'user interface test case #1'.
- **Test Designed By** Name of the Tester.
- **Test Designed Date**: Date when it was written.

- **Test Title/Name**: Test case title. **For Example**, verify the login page with a valid username and password.
- **Test Steps**: List all the test execution steps in detail. Write test steps in the order in which they should be executed. Make sure to provide as many details as you can.
- **Expected Result**: What should be the system output after test execution? Describe the expected result in detail including message/error that should be displayed on the screen.

Test Case Template

Optional (recommended) Fields:

- **Test priority (Low/Medium/High)**: This is very useful while test execution. Test priority for business rules and functional test cases can be medium or higher whereas minor user interface cases can be of a low priority. Test priority should always be set by the reviewer.
- **Pre-conditions**: Any prerequisite that must be fulfilled before the execution of this test case. List all the pre-conditions in order to execute this test case successfully.
- **Test Summary/Description**: Describe the test objective in brief.
- **Test Data**: Use of test data as an input for this test case. You can provide different data sets with exact values to be used as an input.
- **Attachments/References**: This field is useful for complex test scenarios in order to explain the test steps or expected results using a Visio diagram as a reference. Provide the link or location to the actual path of the diagram or document.
- **Module Name**: Mention the name of the main module or the sub-module.
- **Post-condition**: What should be the state of the system after executing this test case?
- **Requirements Link**: Requirements for which this test case is being written for. Preferably the exact section number of the requirement doc.

Test Case examples (Zephyr for Jira)

The screenshot displays the Jira Zephyr Test Case interface for a project named 'SM Portfolio Project'. The top navigation bar includes links for Dashboards, Projects, Issues, Boards, Tests, and a 'Create' button. A search bar is located on the right. The left sidebar shows project navigation options: Backlog, Active sprints, Releases, Reports, Issues (selected), Components, Tests, and Custom Fields. Below these are project shortcuts and an 'Add link' button.

The main content area is titled 'Guru99 Example Test Case' and includes a breadcrumb 'SM Portfolio Project / TP-60'. Action buttons at the top include Edit, Comment, Admin, Clone, More Actions, and Execute. An 'Export' button is also present.

Details

Type:	Test	Status:	TO DO (View Workflow)
Priority:	Medium	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Component/s:	None		
Labels:	None		

Description

Click to add description

Test Details

Freeze test no. and steps

Add step Columns

	Test Step	Test Data	Test Result	Attachments	Actions
1	Guru99 Example Test Step	Any data that is required for the test step	The expected result from this test step	0 attached	
2					

Test Executions

Freeze version and status Columns

People

Assignee: Unassigned
Assign to me

Reporter: Test Manager

Votes: 0

Watchers: 1 Stop watching this issue

Dates

Created: Just now

Updated: Just now

Agile

View on Board

HipChat discussions

Do you want to discuss this issue? Connect to HipChat.

Connect Dismiss

Test Case examples (TestFlo for Jira)

Online Store Application / OSA-573 Version 1.3.6 - Build 2.1. - Internet Explorer / OSA-574

Change the order/shipping address

EditCommentAssignMoreRetestAdmin

PrintableExport

Details

Type:Test Case

Priority:Medium

Affects Version/s:None

Component/s:Account

Labels:None

Sprint:OSA Sprint 2

Preconditions:The customer is logged in to the account

Requirement:OSA-9

TC Status:Fail

TC Template:OSA-92

Fails:1

Steps Progress:100%

Status:FAIL (View Workflow)

Resolution:Unresolved

Fix Version/s:None

Issue Links

defectOSA-577 The changed order/shipping address in the customer account is n... DONE

Steps

	Action	Input	Expected result	Status
1	Go to the address data tab		View of address data tab	Passed
2	Make changes to your data and click save button		Changes have been saved	Failed

People

Assignee:Elizabeth Hoover
Assign to me

Reporter:Max Power

Votes:0 Vote for this issue

Watchers:1 Start watching this issue

Test information

PROGRESS

0Waiting2Executed1 Passed1 Failed

100%

ACTIONS

Create Defect for OSA-9

Link with Requirement

DEFECTS

OSA-577 DONE

Agile

Completed Sprint:OSA Sprint 2 ended 11/Jul/17

View on Board

Test Case examples (Test Link)

TestLink 1.9.16 (Moka pot)

ashish [admin]

Test Project: SP Sample TestLink Project

Test Suite: Child Suite - Create Test Case

Create Cancel ☐ check to create another test case after saving

Test Case Title

Sample first test case

Summary

test case sample

body p

Preconditions

1. TestLink should be installed

Test Case

gm-1:GmailLogin

Version 1

Summary

This Testcase is to test the login functionality of gmail Page.

Preconditions

1. User should be connected to the internet.
2. User should have valid gmail username and password.

Step actions	Expected Results	Execution
1 Open Gmail Website	The Website should be opened.	Manual
2 Enter username in the username textbox.	textbox should accept the entered data.	Manual
3 Enter password in the password textbox.	textbox should accept the entered data.	Manual
4 Click on "signin" button.	Login should success and navigate to the mail box page.	Manual

Create step Resequence Steps

Status: Draft Importance: Medium Execution type: Manual Estimated exec. (min): Save

Keywords: None

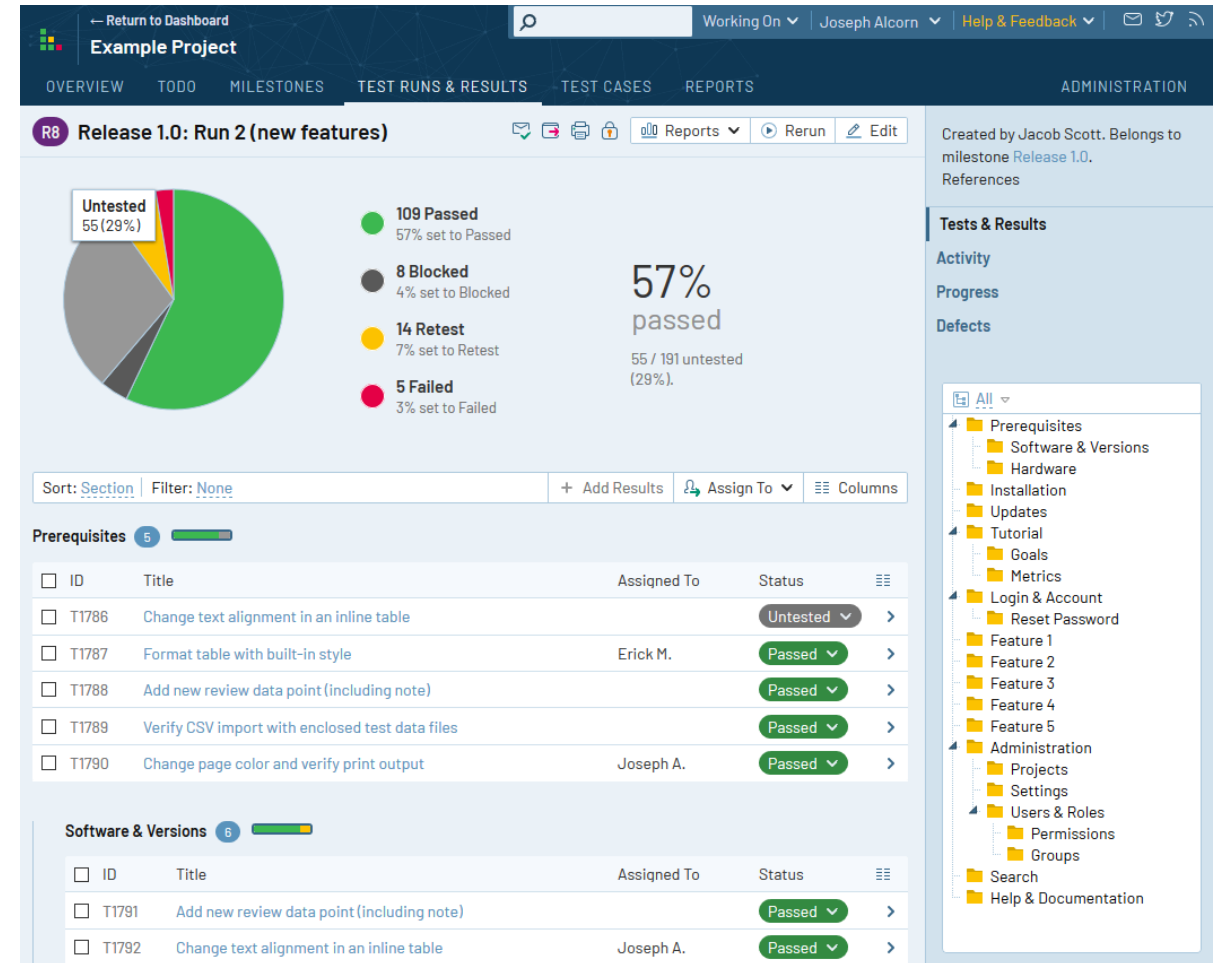
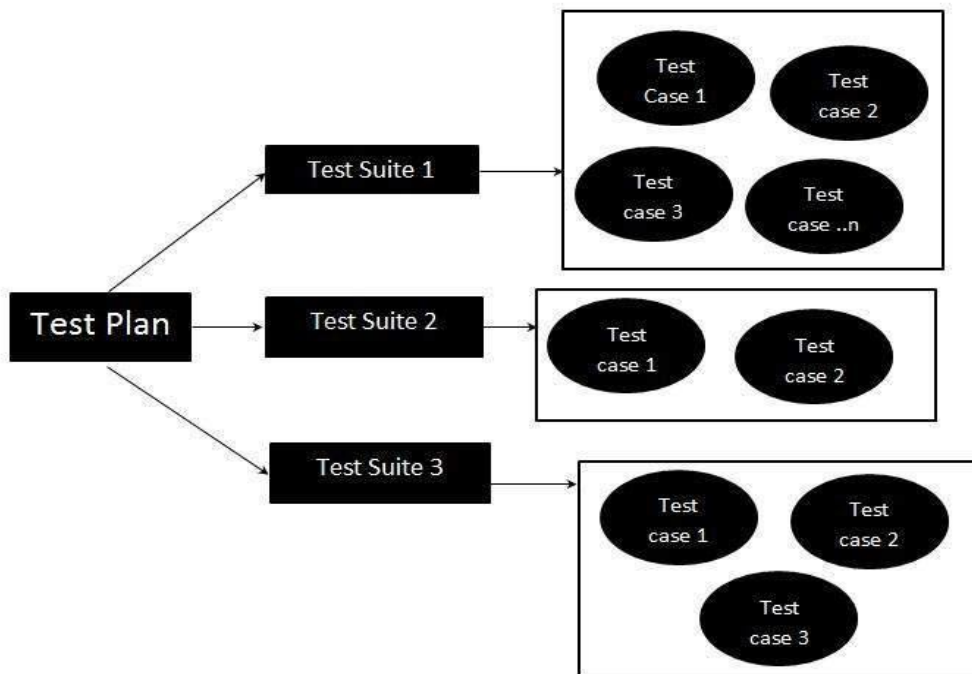
Requirements: None

Test Suite. Test Case planning and Execution.

Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, In progress and completed.

A Test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests.

Test suites are created based on the cycle or based on the scope. It can contain any type of tests, viz - functional or Non-Functional.



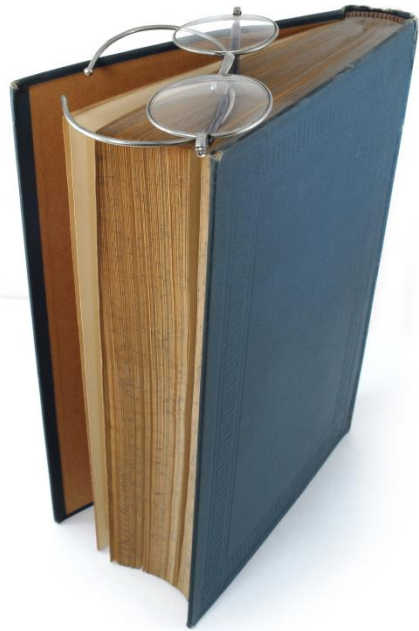
Test Case Example (notepad)

	A	B	C	D	E	F	G	H	I	J	K	L
1	Тест-кейсы для тестирования программы "Блокнот"											
2	ID	Author	Date creation	Test category	Priority	Summary	Preconditions	Steps to reproduce	Expected result	Pass/Fail	Environment	Note
3	OPEN-001	Fisrt Name Last name	16-03-15	Открытие программы "Блокнот"	High	Открыть программу "Блокнот" через ярлык на рабочем столе		1. Найти на рабочем столе ярлык "Блокнот" 2. Кликнуть по нему дважды.	Открывается окно "Безымянный -Блокнот"			
4	OPEN-002	Fisrt Name Last name	16-03-15	Открытие программы "Блокнот"	High	Открыть программу "Блокнот" через меню кнопки "Пуск"		1. Нажать кнопку "Пуск" 2. В меню выбрать пункт "Программы" и нажать 3. Выбрать в меню пункт "Стандартные" . 4. Выбрать в списке программ "Блокнот" и нажать	Открывается окно "Безымянный -Блокнот"			
5	OPEN-003	Fisrt Name Last name	16-03-15	Открытие программы "Блокнот"	High	Открыть программу "Блокнот" в форматах txt, html, doc		1. На необходимый файл в формате txt, html, doc наводим курсор мышки 2. Кликнуть правой кнопкой мышки 3.Выбираем в меню опцию "Открыть с помощью"Блокнот"	Открывается файл в окне "Блокнот"			
6	SCALING-001	Fisrt Name Last name	16-03-15	Действия с окном программы "Блокнот"	Medium	Свернуть окно программы "Блокнот"		1. В правом верхнем углу открытого окна "Блокнот" выбрать кнопку "Свернуть" 2. Нажать кнупку "Свернуть"	Окно "Безымянный - Блокнот" сворачивается			
7	SCALING-002	Fisrt Name Last name	16-03-15	Действия с окном программы "Блокнот"	Medium	Развернуть окно программы "Блокнот"		1. В правом нижнем углу экрана найти кнопку с надписью "Безымянный - Блокнот" 2. Нажать на кнопку	Открывается окно "Безымянный -Блокнот"			
8	SCALING-003	Fisrt Name Last name	16-03-15	Действия с окном программы "Блокнот"	Medium	Раширение окна программы "Блокнот"		1. В правом верхнем углу открытого окна "Блокнот" выбрать кнопку с изображением квадрата 2. Нажать кнопку с изображением квадрата	Окно "Безымянный - Блокнот" разворачивается во весь экран			
								1. В правом верхнем углу открытого окна				

Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice

What is a Test Management Tool?



A **tool** that provides a possibility to store and manage requirements, to create, edit and execute test cases, to track bugs, risks and issues, to track coverage, to link bugs to test steps during test cases execution, to report.

Test Management Tool examples

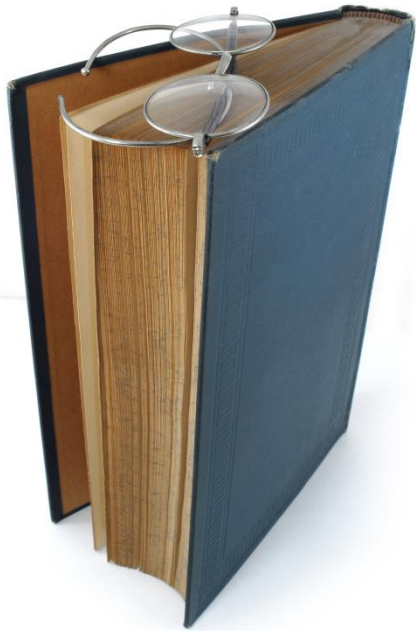


Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice

Coverage

Coverage is an amount of testing performed by a set of executed test cases.



Coverage categories

- ❑ **Program based coverage.** This category is concerned with coverage related to the software code.
- ❑ **Specification based coverage.** This category is concerned with coverage related to specification or requirements.

```
port class MainActivity extends  
@Override  
public void onCreate(Bundle savedInstanceState)  
super.onCreate(savedInstanceState)  
setContentView(R.layout.activity_main)  
public void onClick(View view)  
Intent i = new  
webView.setWebViewClient(new Call  
myRec webView.setWebViewClient(new Call
```

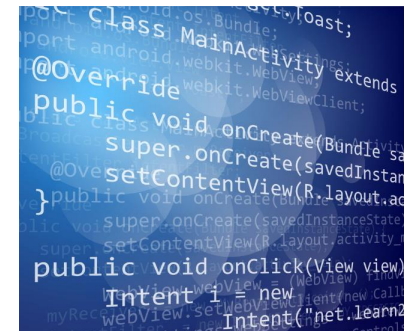


SPECIFICATIONS

Program based test coverage

For software code there are several types of test coverage:

- **Function coverage** – each function has been called at least once
- **Statement coverage** – each row has been executed at least once
- **Branch coverage** – each branch (for example, IF statement, loop) has been executed at least once
- **Condition coverage** – each condition (boolean “true” or “false”) has been executed at least once
- ...



```
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onClick(View view) {
        Intent i = new Intent("net.learn2code");
        myWebView.setWebViewClient(new Call
        myWebView.loadUrl("http://www.learn2code.com");
    }
}
```

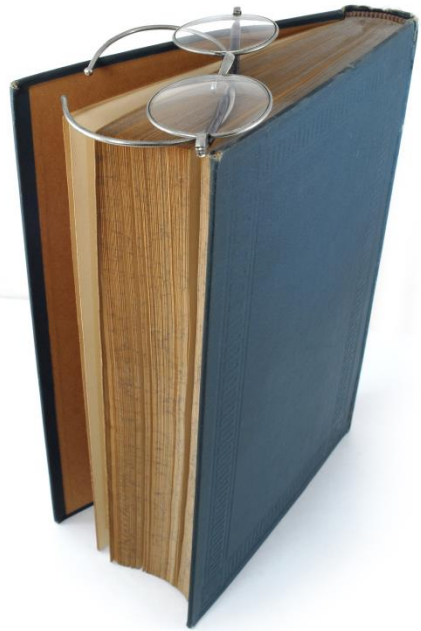

Specification/requirements based test coverage

- test coverage shows how many requirements are covered with test cases.
- test coverage depends on the level of details considered in the specification/requirements



SPECIFICATIONS

Traceability



Traceability in software testing is the ability to trace tests forward and backward through the development lifecycle.

Traceability matrix

REQUIREMENTS TRACEABILITY MATRIX					
Project Name: Online Flight Booking Application					
Business Requirements Document BRD		Functional Specification Document FSD		Test Case Document	
Business Requirement ID#	Business Requirement / Business Use case	Functional Requirement ID#	Functional Requirement / Use Case	Priority	Test Case ID#
BR_1	Reservation Module (It should allow user to book one or more tickets, one way or round way for future dates)	FR_1	One Way Ticket booking (It should allow user to book one way ticket)	High	TC#001 (Verify if user is able to book one way ticket) TC#002 (Verify if user is able to book multiple way tickets)
BR_1		FR_2	Round Way Ticket (It should allows user to book round way ticket)		TC#003 (Verify if user is able to book round way ticket) TC#004 (Verify if user is able to book multiple round way tickets)
BR_1		FR_3	Multiplicity Ticket booking (It should allows user to book one way or round way ticket for multiple cities)	High	TC#005 (Verify if user is able to book one way ticket for multiple cities) TC#006 (Verify if user is able to book round way ticket for multiple cities)
BR_2	Payment Module (User should able to make payment for booked tickets via Credit / Debit Card or through Reward Points)	FR_4	By Credit Card (It should allows user to make payment by Credit Cards)	High	TC#007 (Verify if user is able to pay by Maste Ccard) TC#008 (Verify if user is able to pay by Visa Card)
BR_2		FR_5	By Debit Card (It should allows user to make payment by Debit Cards)	High	TC#009 (Verify if user is able to pay by Debit Card)
BR_2		FR_6	By Reward Points (It should allows user to make payment by Reward Points)	Medium	TC#010 (Verify if user is able to pay fully by Reward Points) TC#011 (Verify if user is able to pay partially by Reward Points)

Traceability in software testing is often done using a traceability matrix. It might include:

- Requirements, user stories, or epics.
- Test cases for those requirements/user stories/epics.
- Test runs (and their results).
- Issues or defects (and whether they've been resolved).

Linkage in
Test/Issue
Management
Tool:



Requirement -> Test Case -> Test Run -> Issue
OR
Issue -> Test Run -> Test Case -> Requirement

Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice

Test cases to automate

- **Business critical paths** – the features or user flows that if they fail, cause a considerable damage to the business.
- Tests that need **to be run against every build/release** of the application, such as smoke test, sanity test and regression test.
- Tests that need to run against **multiple configurations** — different OS & Browser combinations.
- Tests that execute **the same workflow but use different data** for its inputs for each test run.
- Tests that involve inputting **large volumes of data**, such as filling up very long forms.
- Tests that can be used for **performance testing**, like stress and load tests.
- Tests that take a **long time to perform** and may need to be run during breaks or overnight.
- Tests during which **images must be captured** to prove that the application behaved as expected, or to check that a multitude of web pages looks the same on multiple browsers.



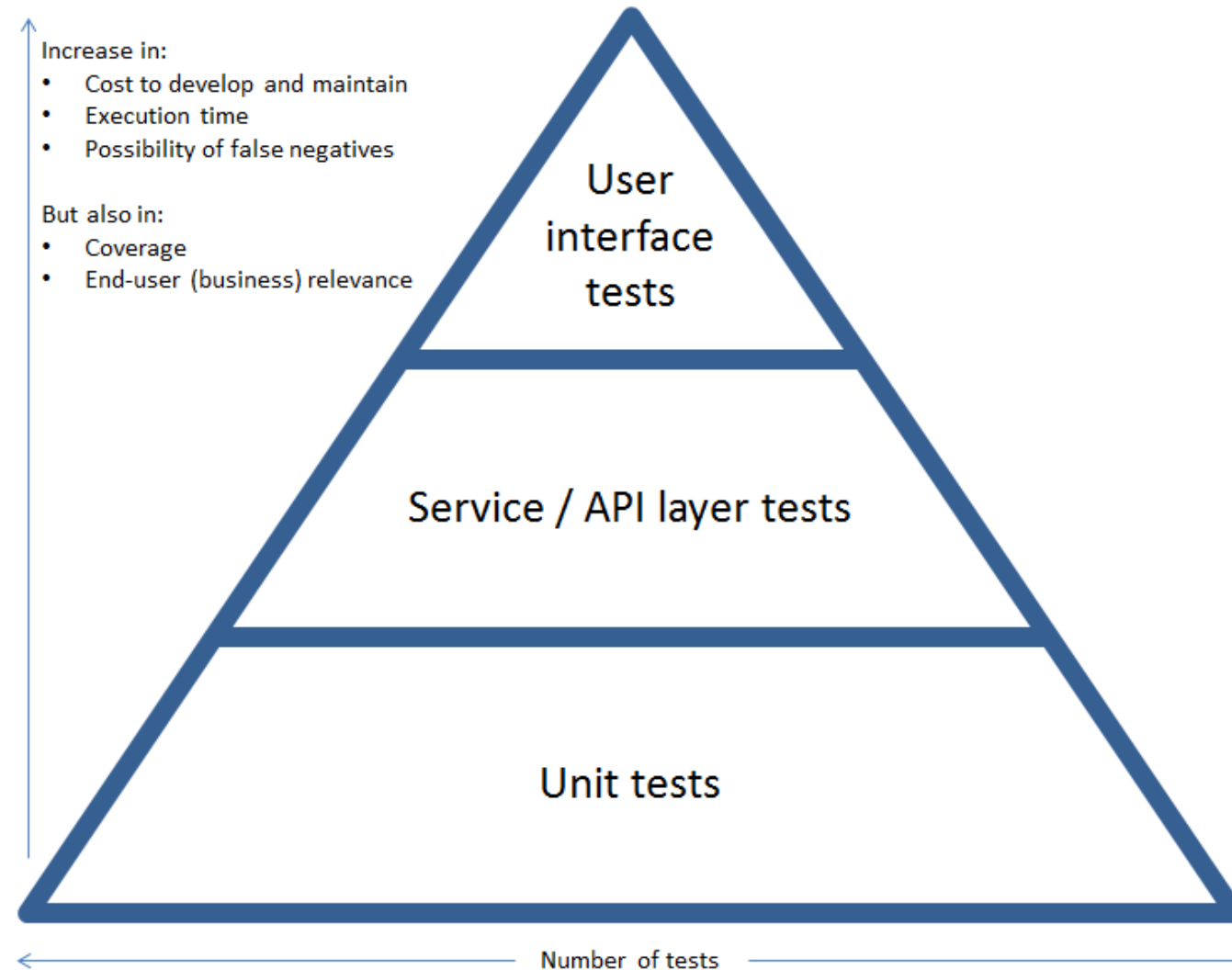
Test cases not to automate



NO

- Tests that you will only **run only once**. The only exception to this rule is that if you want to execute a test with a very large set of data, even if it's only once, then it makes sense to automate it.
- User experience tests for **usability** (tests that require a user to respond as to how easy the app is to use).
- Tests that need to be run **ASAP (as-soon-as-possible)**. Usually, a new feature which is developed requires a quick feedback so testing it manually at first.
- Tests that require **ad hoc/random testing** – Exploratory Testing.
- **Intermittent tests**. Tests without predictable results cause more noise than value. To get the best value out of automation the tests must produce predictable and reliable results in order to produce pass and fail conditions.
- Tests that require **visual confirmation**, however, we can capture page images during automated testing and then have a **manual check of the images**.
- Test that **cannot be 100% automated** should not be automated at all, unless doing so will save a considerable amount of time.

Test automation pyramid



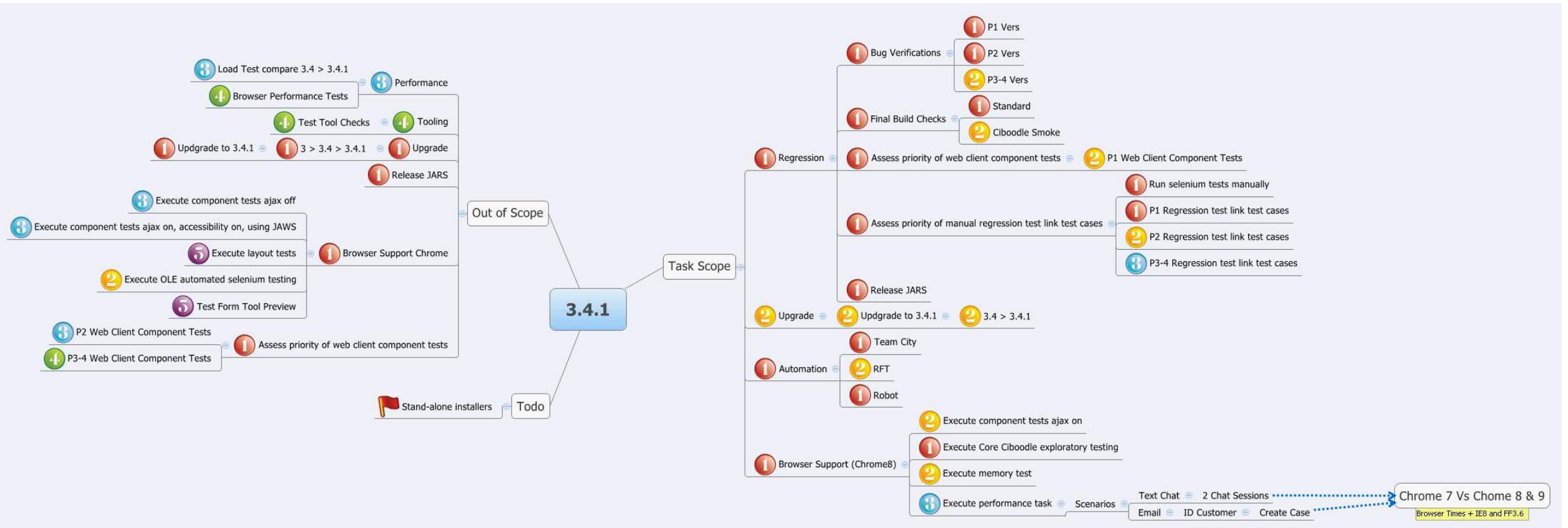
Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice

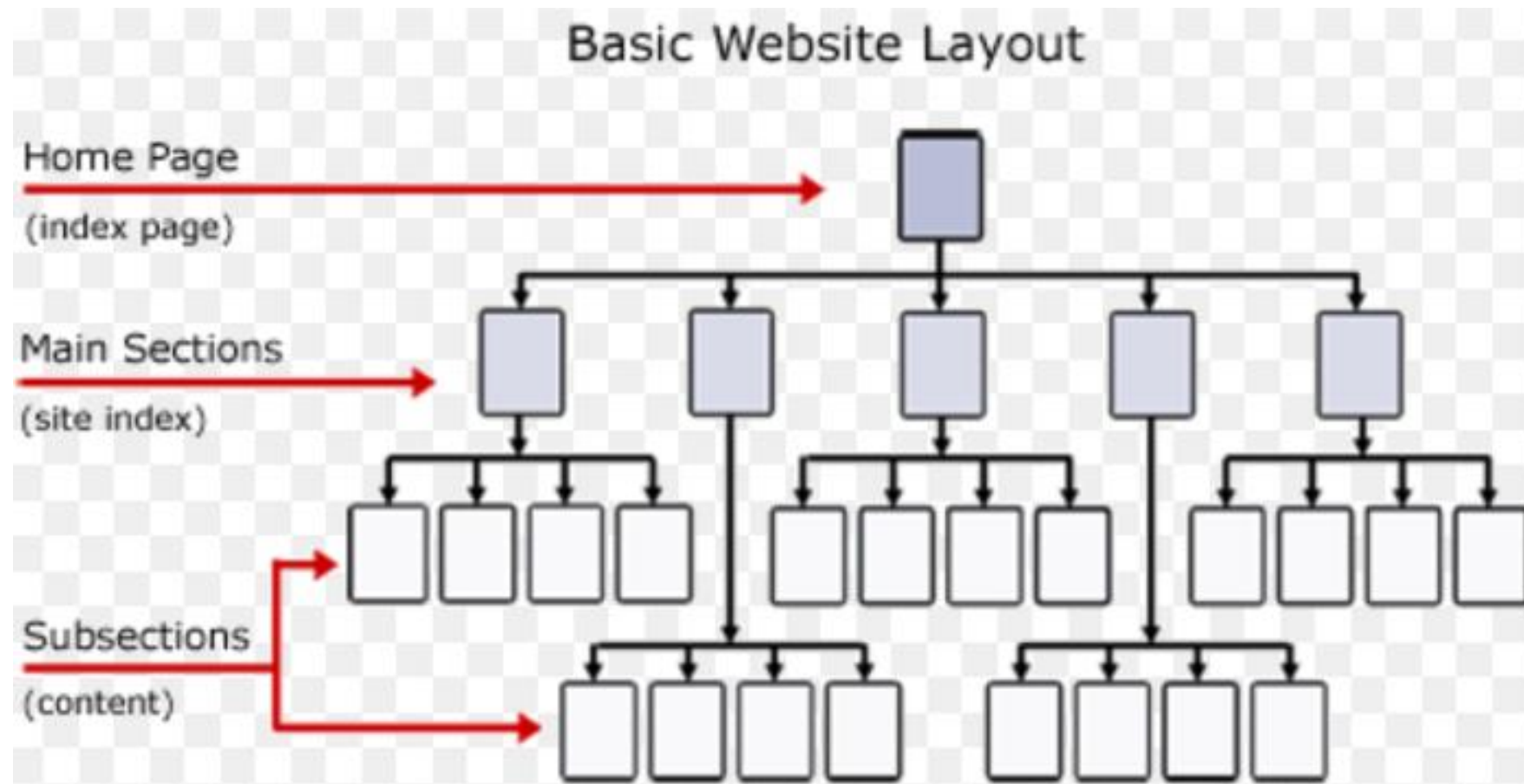
What is a Mind Map?

- A mind map is an easy way to brainstorm thoughts organically without worrying about order and structure. It allows you to visually structure your ideas to help with analysis and recall.
- A mind map is a diagram for representing tasks, words, concepts, or items linked to and arranged around a central concept or subject using a non-linear graphical layout that allows the user to build an intuitive framework around a central concept. A mind map can turn a long list of monotonous information into a colorful, memorable and highly organized diagram that works in line with your brain's natural way of doing things.

Mind Map example



Splitting website functionalities



How can we
Split?

1A.LV

TAVS VIEDVEIKALS

info@1a.lv

67428800

Par 1a

Meklēt preci

ienākt / Reģistrēties
Mans profils

Vēlmju
saraksts

0

0.00 €
Mans grozs

Telefoni, planšetdatori,
Apple veikals

Dator tehnika, preces
birojam

Datoru komponentes, tīkla
produkti

Sadzīves tehnika

TV, audio, video, spēļu
konsoles

Smaržas un kosmētika

Skaistumam un veselībai

Sports un fitness

Bērniem, mazuļiem

Zoo preces

Dārzam, remontam un mājai

Auto un moto preces

Tūrisma preces

Mājsaimniecības preces

Santehnika un apkure

Foto tehnika, optika

Apavi, apģērbs un aksesuāri

MAZLIETOTAS UN
ATJAUNOTAS PRECES

DĀVANU KARTES

Visas kategorijas

Kāds ir mana
pasūtījuma statuss?

0%
LĪZINGS*

ATJAUNOTI
STACIONĀRIE
PORTATĪVIE DATORI*

LIELISKA IESPĒJA IEGĀDĀTIES LĒTĀK!

IESKATIES

*Dators atbilst oficiālajai ražotāja RENEW programmai. RENEW ir ražotāja rūpnīcā atjaunots dators. Dators nav lietots. Atkarībā no izvēlēta datora, tiem ir 1 - 2 gadu garantija.

PIEGĀDE
BEZMAKSAS
PIEGĀDE

Tikai šodien līdz 23:59!

3 dienas
SUPERCEŅAS SMARŽĀM UN
KOSMĒTIKAI!

Akcija spēkā līdz 20.10!

XIAOMI MI BAND 3

Supercena - tikai 36.99
EUR!

ATJAUNOTI
DATORI

Ieskaties!

PIEGĀDE BEZ MAKSAS!
AR KODU FREE

TIKAI ŠODIEN

PIEGĀDE BEZ MAKSAS!
AR KODU FREE

TIKAI ŠODIEN

PIEGĀDE BEZ MAKSAS!
AR KODU FREE

TIKAI ŠODIEN

E-VEIKALS

NR.1

LATVIJĀ

- Telefoni, planšetdatori
Apple veikals
- Datortehnika, preces
birojam
- Datoru komponentes, fiksā
produkti
- Sadzīves tehnika
- TV, audio, video, spēļu
konsoles
- Smaržas un kosmētika
- Skaistumam un veselībai
- Sports un fitness
- Bērniem, mazuļiem
- Zoo preces
- Dārzam, remontam un mājai
- Auto un moto preces
- Tūrisma preces
- Mājsaimniecības preces
- Santehnika un apkure
- Foto tehnika, optika
- Apavi, apģērbs un aksesuāri
- MAZLIETOTAS UN
ATJAUNOTAS PRECES
- DĀVANU KARTES

Visas kategorijas



Kāds ir mana
pasūtījuma statuss?

0% LĪZINGS*

ATJAUNOTI DATORI*

STACIONĀRIE
PORTATĪVIE

LIELISKA IESPĒJA IEGĀDĀTIES LĒTĀK!

IESKATIES

*Dators atbilst oficiālajai ražotāja RENEW programmai. RENEW ir ražotāja rūpnīcā atjaunots dators. Dators nav lietots. Atkarībā no izvēlēta datora, tiem ir 1 - 2 gadu garantija.

**PIEGĀDE BEZMAKSAS
PIEGĀDE**
Tikai šodien līdz 23:59!

**3 dienas
SUPERČENAS SMARŽĀM UN
KOSMĒTIKAI!**
Akcija spēkā līdz 20.10!

XIAOMI MI BAND 3
Supercena - tikai 38,99
EUR!

**ATJAUNOTI
DATORI**
Ieskaties!

PIEGĀDE BEZ MAKSAS!
AR KODU FREE

TIKAI ŠODIEN

PIEGĀDE BEZ MAKSAS!
AR KODU FREE

TIKAI ŠODIEN

PIEGĀDE BEZ MAKSAS!
AR KODU FREE

TIKAI ŠODIEN

Agenda

- Test Documentation. Test Plan. IEEE 829.
- Check list. Simplest form.
- Test Case.
- Test Management Tools.
- Test Coverage. Traceability Matrix.
- Automation. What could be automated?
- Test cases structuring and organization
- Practice

Practice/HomeWork 5

Цели и задачи

1. Научиться работать в Scrum:

- * [practice] Провести Grooming meeting по требованиям, изучить что за система, которую надо покрывать тестами
 - Рассказать что за система, что она делает, для чего предназначена
 - Посмотреть разделение на фичи, а требования выделено два тестовых объекта и в каждом есть свои подпункты (назовем их фичи)
 - * [practice] Провести Planning meeting:
 - выделить список задач
 - оформить их в беклог
 - создать задачи в трелло
 - распределить их и назначить на тестирующих из вашей команды
 - * [HW] Провести Daily meetings (синхронизироваться в течении работы над домашним заданием между командой, предлагать и просить помощь, review, работать на качество тест плана вместе, командой)
 - * [Lesson #6] Провести Demo session на занятии номер 6. Подготовить небольшую презентацию работы (вольный формат) и описать почему было описано что описано, пройти кратко по основным пунктам и “продать” тест кейсы продукт оверу (это буду я)
 - * [Lesson #6] Провести Retrospective на занятии 6 по итогу работы в команде и трудностей с которыми столкнулась команда по ходу работы
2. Научиться описывать тест кейсы по пройденным техникам тест дизайна, работать с требованиями без продукта, который можно “пощупать”

3. Результатом работы должно быть:

- * Два документа с матрицей и всеми тест кейсами по одному документу от каждой команды, в формате Гугл докс или эксель. Все задачи по Борде (трелло) передвинуты в Done, откомментированы и видны результаты работы всей команды.
- * Презентация и подготовленное демо для менеджера (может проводиться как одним человеком, так и все могут взять свои фичи и рассказать по очереди)

4. Дополнительные комментарии к работе:

- * Тест кейсы желательно описать на английском языке
- * Коммуникация в трелло может быть на любой удобной для вас языке
- * Все тесты должны быть проревьювлены как минимум одним коллегой и это должно быть отмечено в трелло.
- * В конце работы Лид команды должен создать карточку в трелло и заассайнить ее на преподавателя для проверки работы и комментариев к ней.

Grooming meeting

- **Цель:** система позволяет администратору быстро и удобно извлекать, редактировать или удалять необходимые записи о покупателях.
- Основная информация о покупателях:
 - имя (от 3 символов);
 - фамилия (от 3 символов);
 - дата рождения (в формате день-месяц-год);
 - емейл;
 - город проживания;
 - дата последней активности (в формате день-месяц-год).

1. Вход в систему

1.1. Элементы на странице (Feature 1_login)

Страница для авторизации администратора открывается по умолчанию. Поля «E-Mail», «Пароль» и кнопки «Вход в систему» и «Очистить» отображаются на странице. Кнопка «Вход в систему» не активирована пока поля не будут заполнены. При нажатии на кнопку «Очистить» поля на странице очищаются автоматически.

После нажатия на кнопку «Вход в систему» система проверяет валидность введенных данных.

1.2. Сообщения об ошибке (Feature 2_error_messages)

Если пользователь ввел неверный email или пароль, соответственные поля должны быть подсвечены красной рамкой по контуру. Следующие сообщения об ошибке должны отображаться под кнопкой «Вход в систему»:

- введен неверный емейл: Проверьте, пожалуйста, ваш email адрес!
- введен неверный пароль: Пароль не соответствует введенному email адресу.
- логин и пароль не верны: Данные не корректны для выполнения операции.

1.3. Успешная авторизация (Feature 3_success_login)

В случае верно введенных данных, пользователь перенаправлен на страницу «Панель управления». Навигационное меню со следующими элементами должно отображаться на странице: «Поиск»; «Изменить данные»; «Удалить покупателя»; «Выход из системы».

Навигационное меню всегда отображается на странице, независимо какой пункт был выбран администратором. Только кнопки «Поиск» и «Выход из системы» активны по умолчанию.

План работы над заданием номер 5

Как начинать анализировать требования и писать тест кейсы?

1. Прочитать спецификацию
2. Прочитать спецификацию еще раз
3. И еще раз прочитать спецификацию
4. Рассказать самому себе как вы поняли спеку (или коллеге)
5. Представить себе или нарисовать как бы выглядел функционал который надо потестировать
6. Проанализировать функционал (свою часть) и описать чеклист позитивных функциональных тестов.
7. Проанализировать также какие негативные сценарии вероятно будут найдены и использованы пользователем
8. Составьте чеклист с негативными сценариями
9. Распишите каждую проверку детально, описывая какие шаги в системе надо выполнять для той или иной проверки.