Automated Testing

Software Testing, Test Automation, Integration and API Testing and Web UI Test Automation

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

sli.do

#Dev-Ops

Table of Contents

Software Testing Overview

Finding Out How Well Something Works

Software Testing

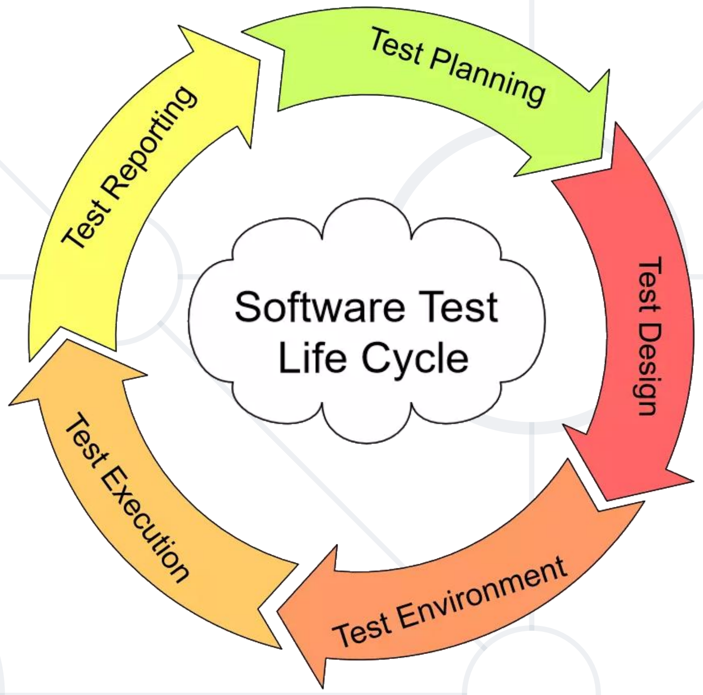
- **Testing** is an important part of the application lifecycle
  - New features need to be verified, before delivered to the clients
- **Testing** covers a wide spectrum in application development
  - There are several **levels** of testing, many **concepts** and **different types** of testing
- **Testing** checks whether the developed software **conforms** to the **requirements**
- Testing aims to **find & report defects** (bugs)

Importance of Software Testing

- **Ensures quality**
  - Helps identify errors and defects
- **Reduces risks**
- **Cost-Effective**
  - Detecting and fixing bugs in the early stages of development can save money in the long run
- **Ensures compliance** to requirements and regulations
- **Facilitates improvement**
  - Provides continuous improvement in the software development process

The Software Testing Process

- **Test planning**
  - What, when, how to test?
- **Test design**
  - Test scenarios & test cases
- **Setup test environment**
  - Install, configure, prepare test data, …
- **Test execution**
  - Perform the tests
- **Test reporting**
  - Log the test results and bugs found



Manual and Automation Testing

- **Manual testing**
  - Type of software testing in which tests are **executed manually without** using any **automated tools**
  - A human performs the tests **step by step**, without test scripts
  - Tests are executed **one by one** in an individual manner
- **Automation testing**
  - Type of software testing in which tests are **executed automatically** via test **automation frameworks**
  - Testers utilize **tools** and **scripts** to automate testing efforts
  - Requires **coding** and test **maintenance**

Different Levels of Software Testing

- **Unit tests**
  - Test individual component
  - Created by developers
- **Integration tests**
  - Test interaction between components (e.g., APIs)
  - Created by developers / QA automation engineers
- **System tests / end-to-end tests**
  - Test the entire system
  - Created by QA automation engineers

Test Driven Development

- **Test Driven Development (TDD)** == writing actual test cases before writing the code
- Helps avoiding defects and makes code clearer
- Steps
  - **Write a test**
  - **Run the rest**
  - **Write the code**
  - **Run the test**
  - **Refactor**

Bug Tracking Tools

Live Demo

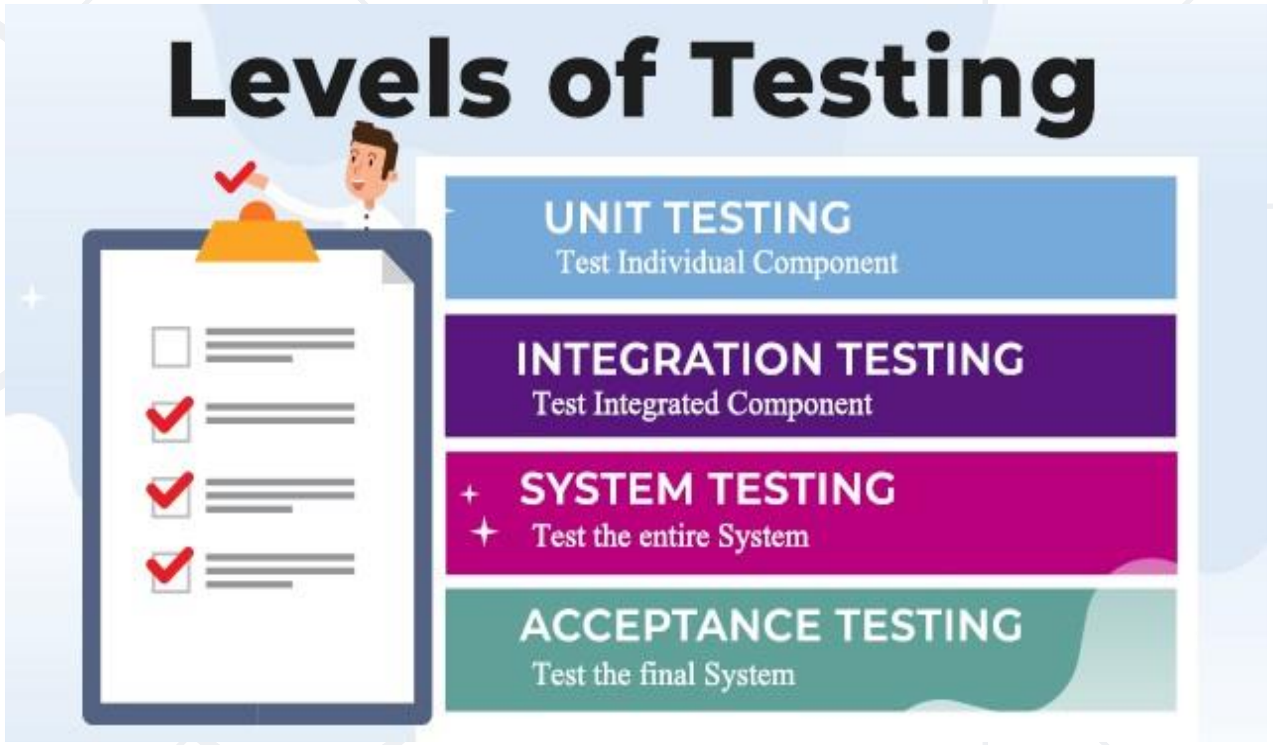
https://github.com/twbs/bootstrap/issues
https://github.com/twbs/bootstrap/issues/31392
https://daosio.atlassian.net/jira/software/c/projects/CART/issues

Test Levels and Test Types

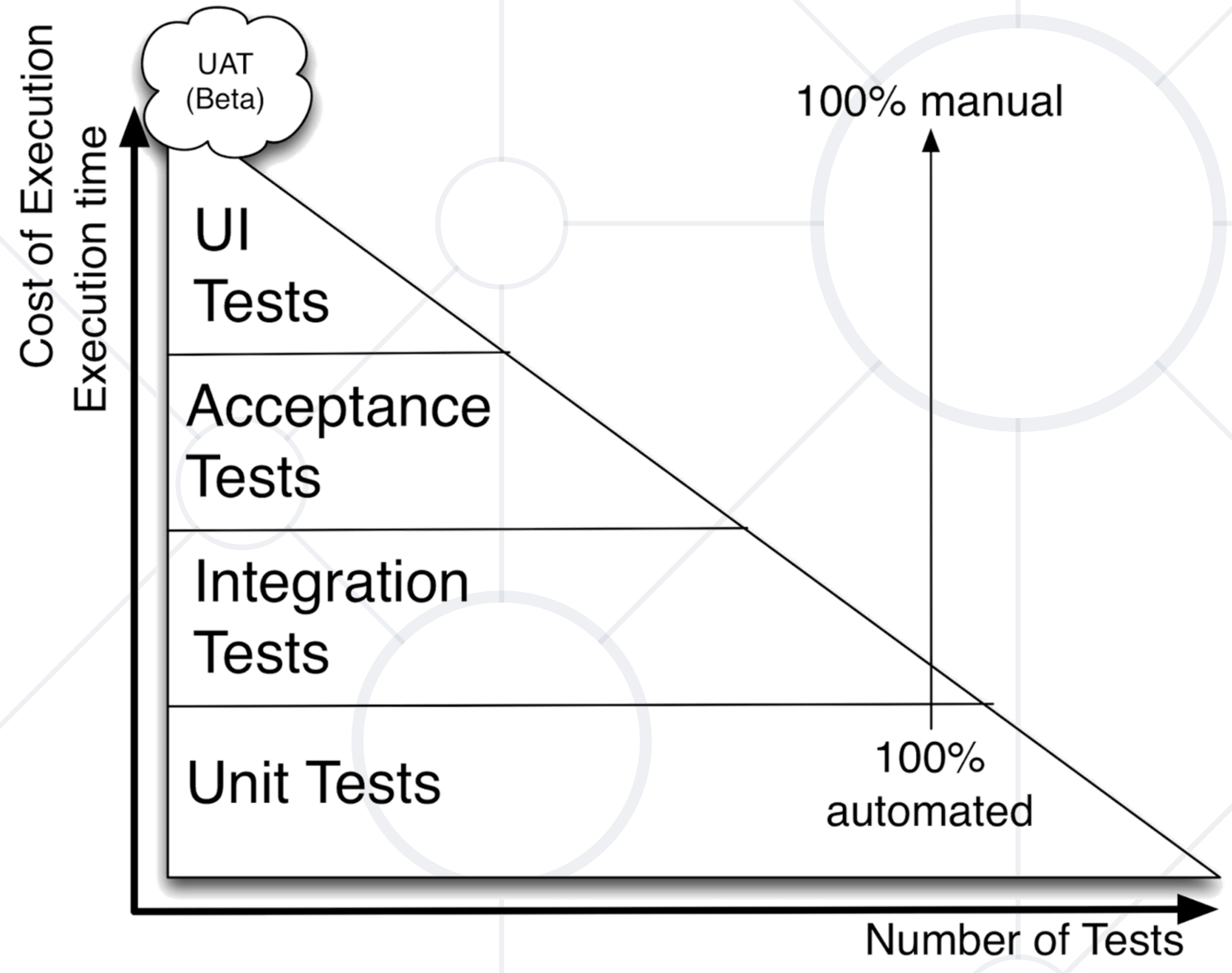
The Step-by-Step Pyramid

Test Levels

- **Unit tests**
  - Test single component
- **Integration tests**
  - Test interaction between components
- **System tests**
  - Tests whether the product meets client's requirements
- **Acceptance tests**
  - Test the entire system



The Testing Pyramid

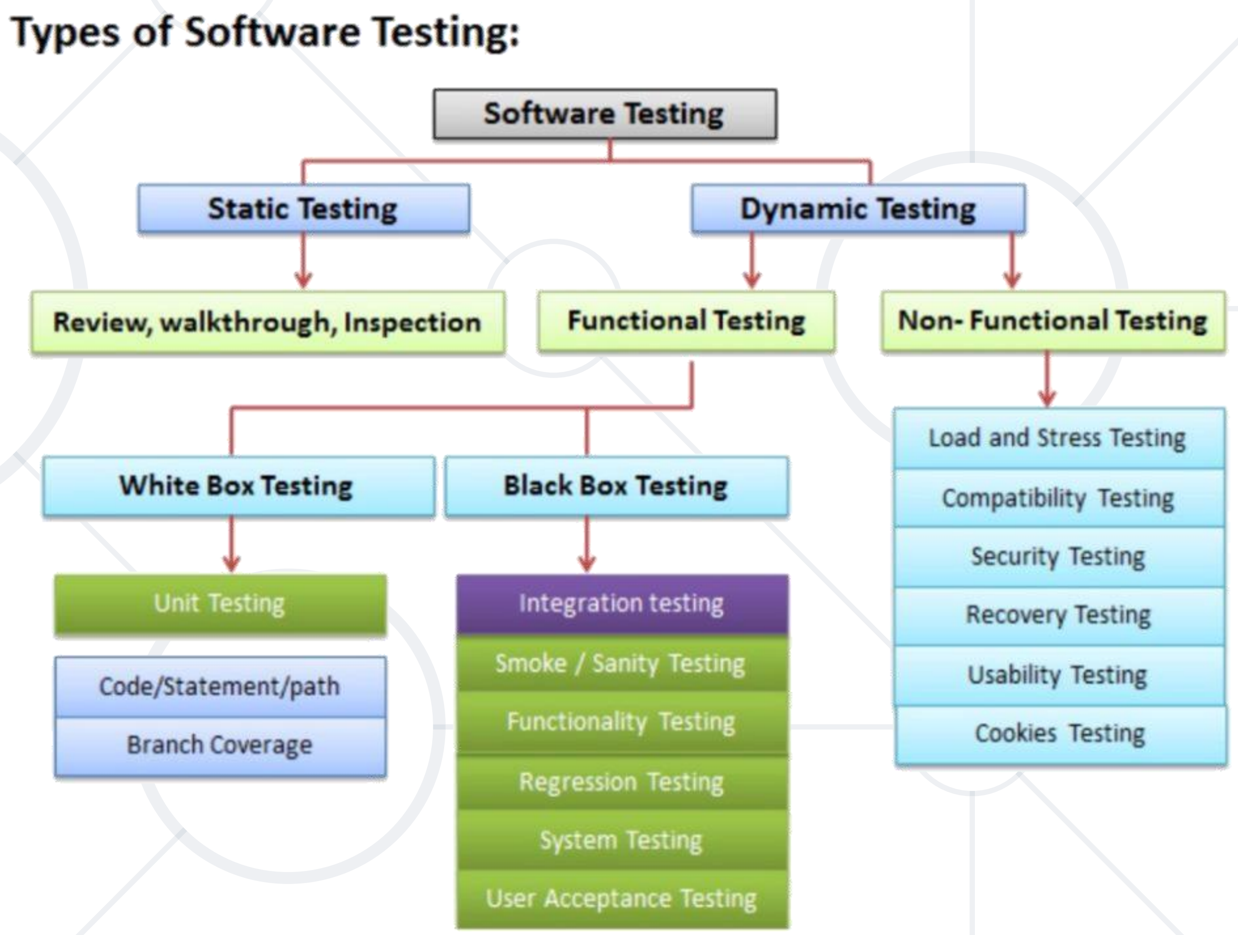


© Allan Kelly

- **Unit tests**: fully automated
- **Integration tests**: fully automated
- **System tests** / **acceptance tests**: partially automated
- **UI / UX tests**: mostly manual

Test Types (1)

- Functional and non-functional
- Black-box and white-box tests, regression tests
- Stress tests, load tests, UX and usability tests, security tests
- Manual vs. automated tests



Test Types (2)

- **Any test type** can be performed at **any test level**

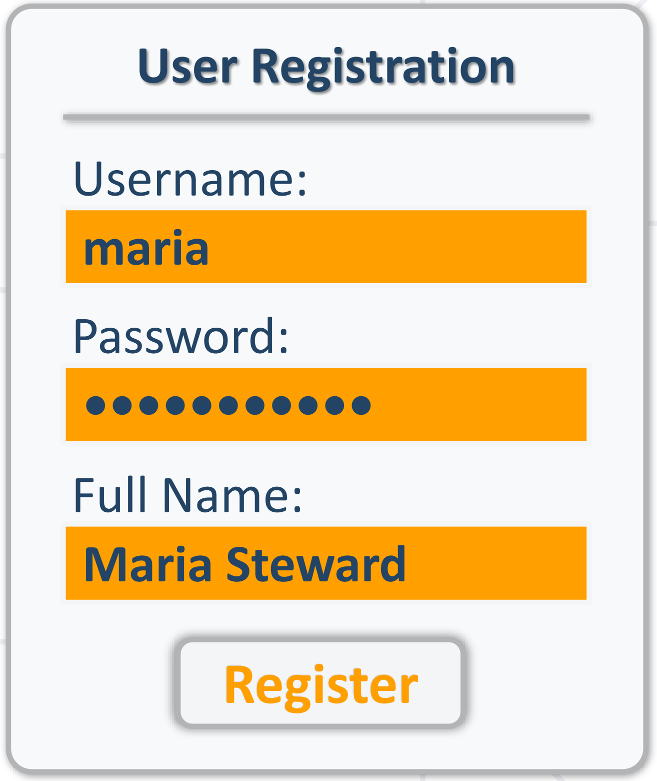
Test Level	Description
Regression Testing	Ensures that a fixed bug **won't happen again**
Load / Stress Testing	Test the application's limits by attempting **large data processing** and intr oducting **abnormal circumstances** and **conditions**
Security Testing	Test if the application has any **security flaws** and **vulnerabilities**
Other Types of Testing	Manual, automation, UI, performance, black box, end-to-end testing, A/ B, etc.

Test Case

- **Test case** == a set of steps, conditions, and inputs used to test a software system to determine if it functions correctly
- At **least two cases** to fully test certain **scenario**
  - A **positive** test
  - A **negative** test
- **Components**
  - Title (+ optional description)
  - Steps to follow
  - Expected result

Test Scenarios and Test Cases – Example

- Sample **test scenario**
  - User registration
- **Test cases** for this scenario
  - Non-existing username → success
  - Duplicated username → error
  - Empty username or password → error
  - Too long username → error
  - Invalid characters in username / password → error



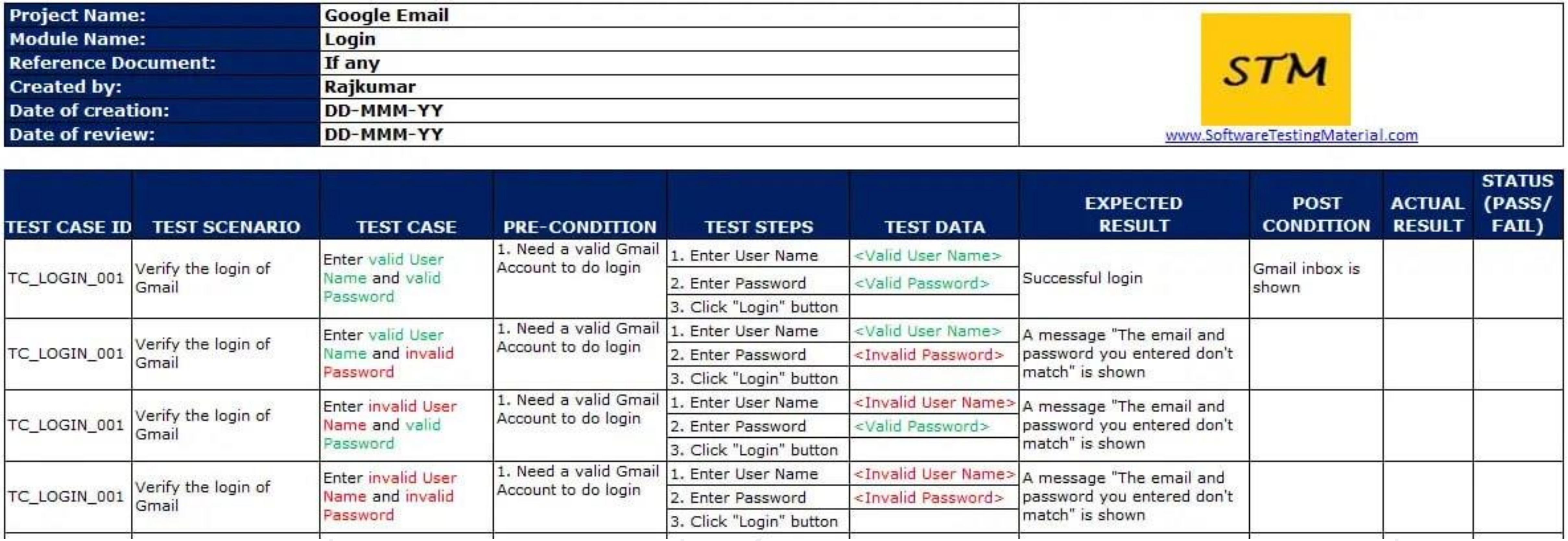
User Registration

Username: **maria**

Password: ●●●●●●●●●●●

Full Name: **Maria Steward**

Test Case – Formal Example



Project Name:	**Google Email**	STM www.SoftwareTestingMaterial.com
Module Name:	**Login**	
Reference Document:	**If any**	
Created by:	**Rajkumar**	
Date of creation:	**DD-MMM-YY**	
Date of review:	**DD-MMM-YY**	

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/ FAIL)
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	Successful login	Gmail inbox is shown		
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					

Test Automation

Test Automation

- **Test automation** is important part of software development
- Test automation is done at many levels
  - **Unit tests**
  - **Integration tests**
  - **UI tests**
- **Test automation tools** record and execute recorded tests
  - Testing **frameworks** (JUnit, NUnit, Mocha, …)
  - Automated testing **tools** (Selenium, Appium, Sikuli)
  - **Web** testing, **API** testing, **mobile** testing

Benefits of Test Automation

- **Improved accuracy**
  - Eliminates human errors in test execution and results
- **Faster feedback**
  - Allows for quick test execution, providing immediate feedback on software quality
- **Increased test coverage**
  - Enables testing of various scenarios, reducing the risk of missing critical test cases
- **Enables continuous testing**
  - Reduces the resources needed for repetitive manual testing

Integration Testing

Combine Individual Modules and Test as a Group

Integration Testing

- **Integration testing** test several components together
  - Combines units and tests them as a group
- Aims to expose faults in the **interaction between integrated units**
  - Checks how well the individual parts work together
- Integration testing is implemented by
  - **Testing framework** + test stubs / mocks

Approaches

- **Top-Down**
  - Testing starts at the top of the control flow or architectural structure
- **Bottom-Up**
  - Testing starts at the bottom of the control flow or architectural structure
- **Hybrid**
  - Combines Top-Down and Bottom-Up approaches
- **Big Bang**
  - All of the units are tested together at the same time

Role in Software Development Life Cycle

- Performed **after unit** testing and **before system** testing
- Identifies problems when **individual modules interact**
- **Incremental approach**
  - Two modules are testes at the same time → another one is added → and another one → and so on…
    - Ensures smooth interaction between components
    - Helps localizing errors quickly
    - Enhances test coverage

Unit vs. Integration Testing

	Integration Testing	Unit Testing
Scope	Groups components to test t hem together	Focus on individual component s
Purpose	Validate that different parts o f the application work correct ly together	Validate functionality of separa te parts of the application
Order in Dev Process	Performed after unit testing	First level of testing
Error localization	Harder because of grouped c omponents	Easier because of isolated com ponents

API Testing

Testing a APIs

API Testing

- **API testing** tests APIs directly and as part of integration testing
- Aims to determine if APIs meet requirements for
  - **Functionality**
    - Includes request and response, endpoints, error codes, etc.
  - **Reliability**
    - Test consistent connection
  - **Performance**
    - Includes API's response time under various conditions
  - **Security**
    - Includes authentication, permissions and access controls

Benefits

- **Early bug detection** at the business layer
- **Ease of integration** between components
- Improves **test coverage**
- **Faster testing** since UI test are not needed
- **Language independent**
  - Data is exchanged in XML or JSON, I/O are selected through HTTP
- Enhanced **application security**
- Supports **Agile** and **CI/CD**

Integration and API test with JS & Mocha

Live Demo

https://github.com/nakov/MVC-app-integration-tests-example-mocha

Web UI Test Automation

Testing a APIs

Web UI Testing

- **Web UI testing** tests components which **users interact** with
- Aims to determine if APIs meet requirements for
  - **User Experience**
    - UI == first point of contact with app for users
  - **Functionality**
    - Ensures all visual components work as expected
  - **Compatibility**
    - Checks whether all devices display web app correctly
  - **Performance**
    - Tests how UI performs under different conditions

Playwright

- Widely used **Node.js** library for end-to-end testing of web application
- Allows **automation** of browser tasks
- **Supports** all modern browsers
- Enables testing of **complex** UIs
- **Automates** form submissions, UI interactions, keyboard input, etc.
- **Fast** and **scalable**
  - Runs in headless mode (without UI)

Web UI Testing with JS and Playwright

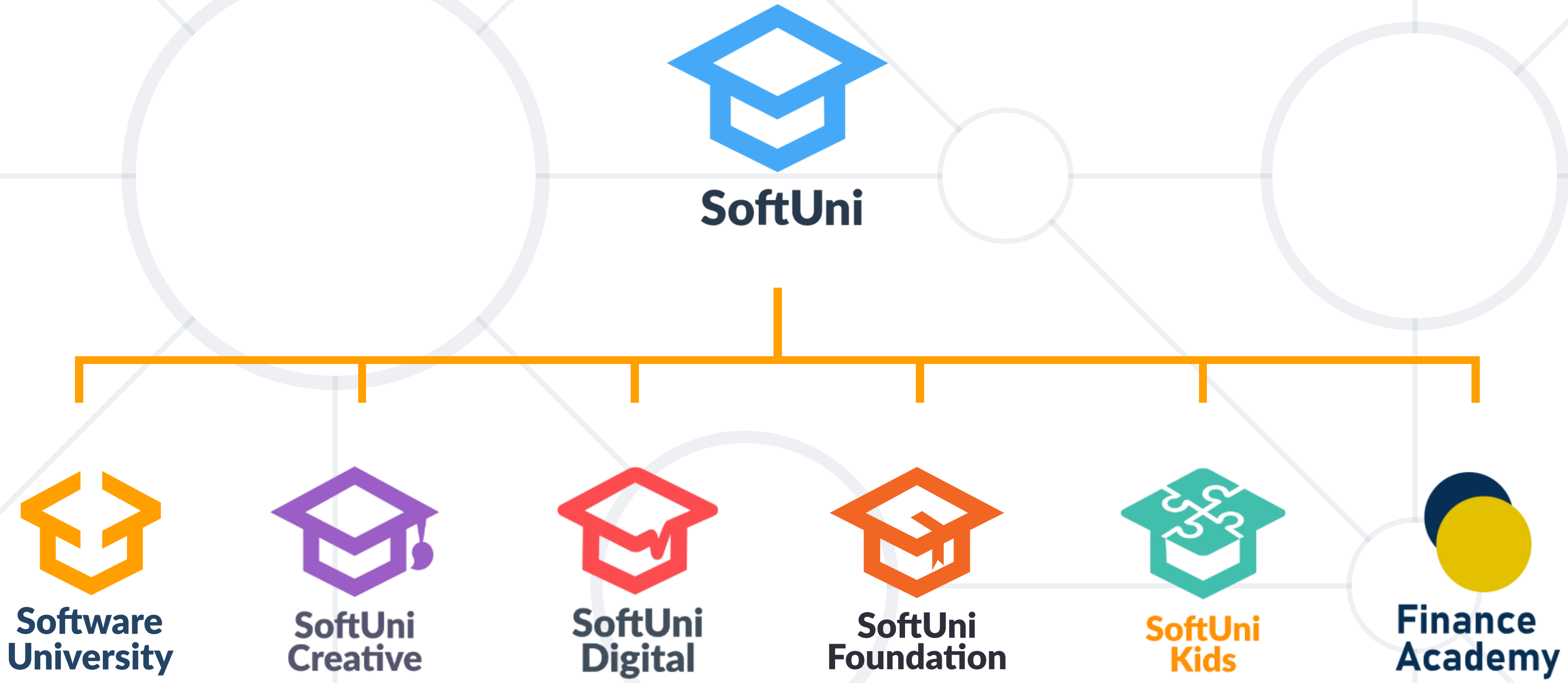
Live Demo

Summary

- **Testing** is important part of software development
- Two types of test – **manual** and **automation**
- Test automation has many **benefits**
  - Improves accuracy, increases test coverage, etc.
- **Integration** testing tests several units together
- **API** testing tests APIs only
- **Web UI test** automation is mostly done with Playwright

Software University

Questions?



SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS POKER | CASINO | SPORTS a Flutter International brand

INDEAVR Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank Решения за твоето утре

BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University

CODE IT UP

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg