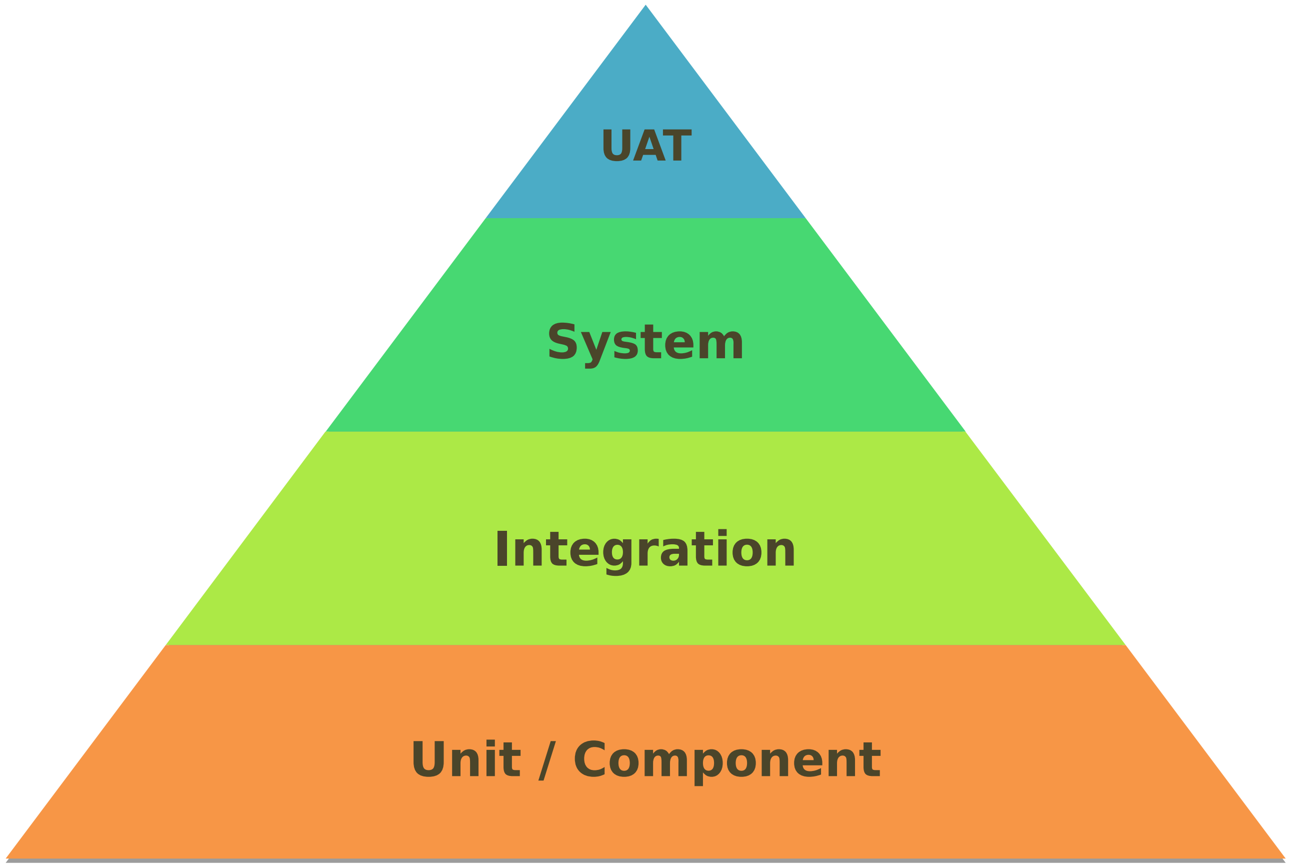
Software Testing

Levels and Types of testing

www.softacademy.bg

Agenda

- Levels of testing (unit, integration, system, acceptance)
- Types of testing
  - Black, White, Grey Box testing
  - Functional & Non-functional testing
  - Smoke, Sanity, Regression testing
  - Positive & Negative testing
  - Front-end & Back-end testing
  - Manual & Automation testing



Testing levels

Unit / Component tests

- Focus is on testing logic of individual function / classes / components
- Requires usage of simulations (mocks / stubs / drivers) to isolate the module under test
- Most commonly performed by the person writing the code
- Popular agile development practice is the “Test driven development” (TDD) or ‘test-first’ approach, in which developer first writes a test, and then the code to make

  that test pass
- Advantages: fast, small, easy to automate & maintain, easy to investigate when fail (single point of failure)
- Disadvantages: cannot guarantee if the actual business functionality (as a whole) works according to requirements

Integration tests

- Focus on validating how different functions / classes / components communicate with each other
- Many different approaches and meaning related to integration testing (incremental, “big-bang”, button-up, top-down, etc.)
- The more integration points are tested at the same time, the more difficult it is to investigate once test fails
- Depending on level of integration, can be performed by either developers or testers
- Advantages: can validate if different parts of the code / system communicate successfully
- Disadvantages: more difficult to automate and maintain, slower than unit tests, harder to investigate failure reason (no single point of failure)

System tests

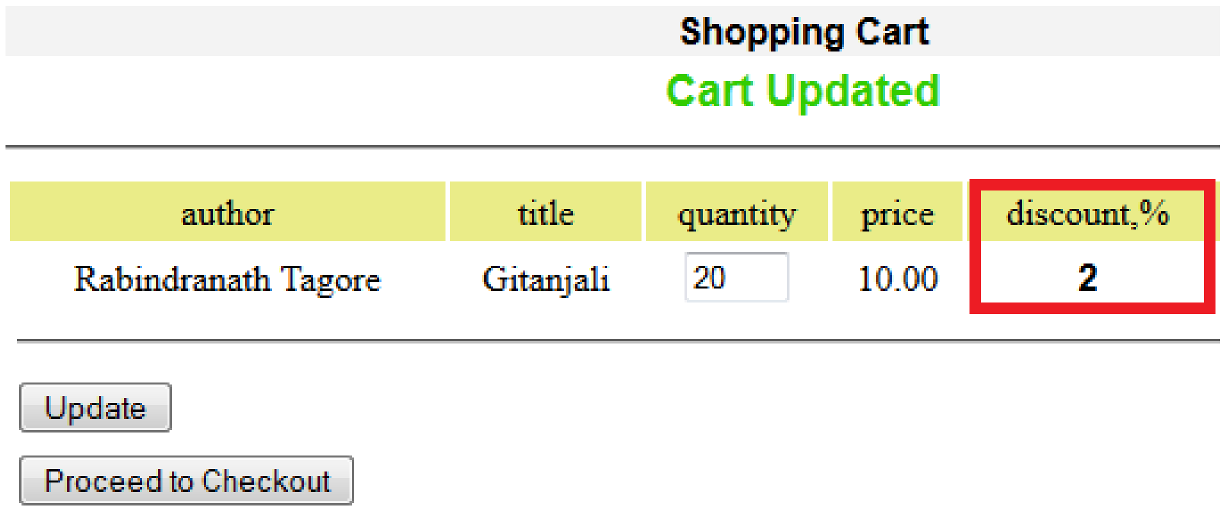
- Also known as System integration tests
- Focus is on testing the system as whole – with realistic environment setup, integration with external systems, etc.
- Tests are often validating end-to-end scenarios (user journeys / workflows)
- Usually performed by testers, since they require more user-perspective and knowledge of the entire system
- Advantages: closest to real user scenarios and setup
- Disadvantages: require complex setup, slow to prepare and execute, usually hard and brittle to automate

Acceptance tests

- Validating if the software adheres to acceptance criteria
- Most often performed by clients / business representatives / regulator
- Advantages: real users / client feedback
- Different types of acceptance testing:
  - User acceptance testing (UAT)
  - Contract / Regulation acceptance testing
  - Alpha / Beta testing

Testing types – Black, White, Grey Box

- Black box testing
  - tests are created based on requirements specification
  - internal structure is not considered (“black box”), only input / output
- White box testing
  - tests are created based on internal structure of the code / system
  - validating code coverage and behavior (example: unit tests)
- Grey box testing
  - combination of both Black and White box testing
  - provides better coverage, as tests take into consideration both input / output and knowledge of the software structure


Shopping Cart

Cart Updated

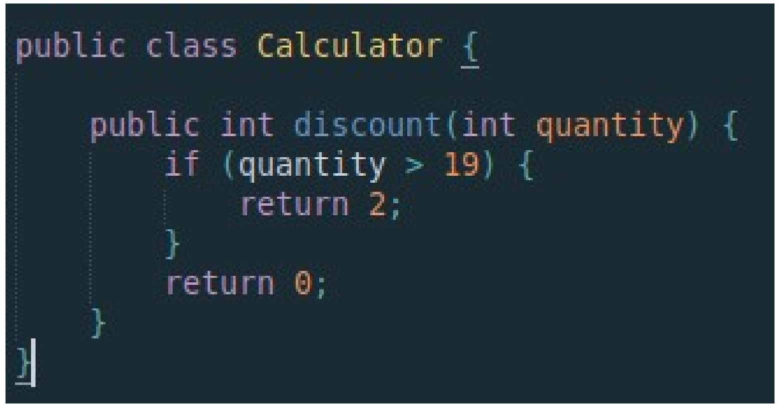
author	title	quantity	price	discount,%
Rabindranath Tagore	Gitanjali	20	10.00	2

Test:

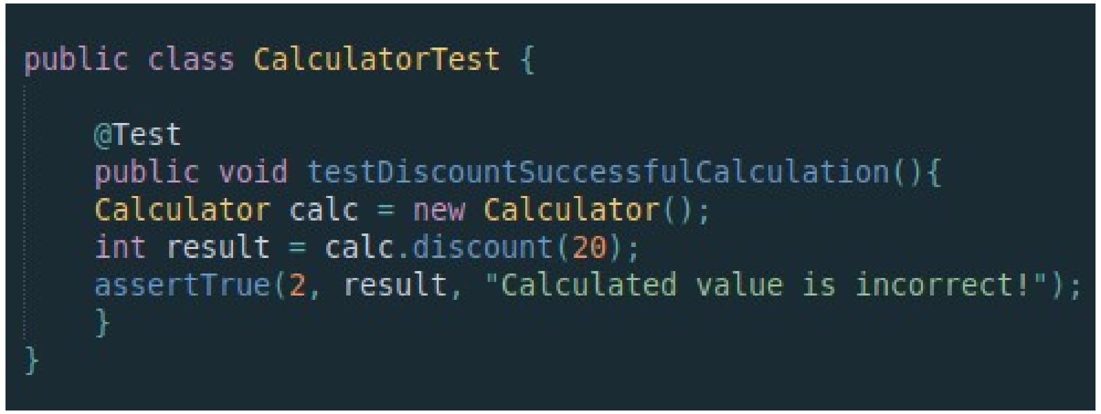
1. Open Shopping Cart
2. Enter quantity = 20
3. Verify discount % field

Expected result:
discount % = 2

Black Box test example

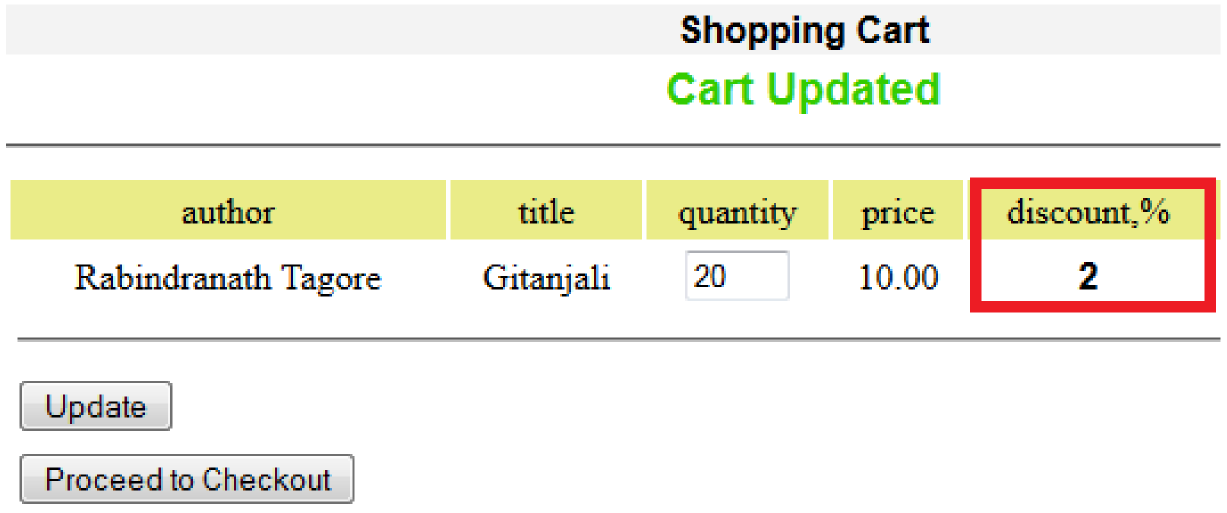


```
public class Calculator {

    public int discount(int quantity) {
        if (quantity > 19) {
            return 2;
        }
        return 0;
    }
}
```



```
public class CalculatorTest {

    @Test
    public void testDiscountSuccessfulCalculation(){
    Calculator calc = new Calculator();
    int result = calc.discount(20);
    assertTrue(2, result, "Calculated value is incorrect!");
    }
}
```

White Box test example



Shopping Cart

Cart Updated

author	title	quantity	price	discount,%
Rabindranath Tagore	Gitanjali	20	10.00	2

transaction_id	quantity	discount_percent	user_id
42423567	20	2	6783

Test:
1. Open Shopping Cart
2. Enter quantity = 20
3. Verify discount % field

Expected result: discount % = 2

1. Proceed to checkout
2. Verify results in database

Expected result: In db.transactions, new record is added in log table, containing transaction details

Grey Box test example

Testing types – Overview

- Numerous testing types exist, based on different classifications
- These classifications may be based on scope of testing, knowledge of internal software structure, approach & technique used, objective of testing, etc.
- Some of the commonly used classifications / testing types include:

  - Black, White, Grey Box testing

  - Functional & Non-functional testing

  - Positive & Negative testing

  - Front-end & Back-end testing

  - Manual & Automation testing

Testing types – Functional testing

- Functional Testing – testing the system against functional requirements / specification. Validates if the requirements are properly satisfied by the application.
- Commonly refer to as Functional testing types:

  - Unit testing

  - Integration testing

  - System testing

  - Smoke testing

  - Sanity testing

  - Regression testing

Testing types – Smoke, Sanity, Regression

- **Smoke test**

  – very limited scope, validating if build is stable to consider any further testing
- **Sanity test**

  – limited scope, but broader compared to the smoke test

  - validates if critical functionalities of the software are working in the new build, before considering detailed testing
- **Regression test**

  - validating if changes made to the software have broken working functionalities

  - usually includes all mandatory tests ensuring a new version meets release exit criteria

Testing types – Non-functional testing

- Non-functional testing: verification of non-functional requirements, i.e. how the software is performing its functions (speed, portability, reliability, etc.)
- Types of non-functional testing:
  - Performance testing (stress, load, volume, etc.)
  - GUI / Visual testing
  - Compatibility testing
  - Security testing *
  - Usability testing
  - Accessibility testing
  - Localization testing
  - Resilience testing
  - etc.

* *may be classified as having both functional and non-functional aspects*

Performance testing

- Performance testing is done to:
  - check the response time of our software
  - find and remove/workaround bottlenecks
  - evaluate scalability options
- Performance testing should have specific objectives:
  - Can the system handle 100 simultaneous users with response time < 1 second and no error?
  - How the system behaves with 1000 users?
  - What load it takes for the system to crash?
  - What is the slowest component of the system?

GUI / Visual testing

- Validation of the graphic user interface (GUI)

  - layout

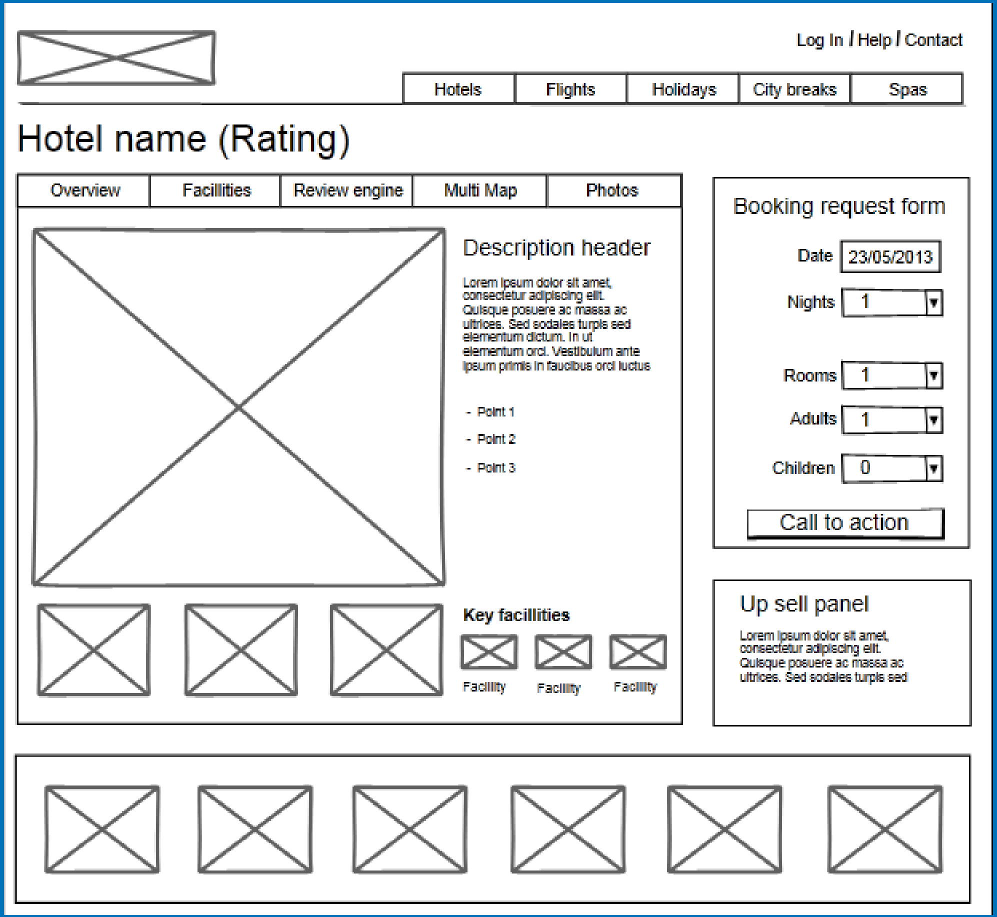
  - fonts

  - colors

  - images

  - etc.

- GUI specification is in the form of mock-ups and wire-frames



Compatibility testing

- Validation of software's compatibility with supported platforms / hardware / software (OS, browsers, devices, software components, etc.)
- Examples:

  - **Browser compatibility testing** – applicable for web applications; validates if the software can run on different combinations of OS / browsers / browser versions

  - **Backwards compatibility testing** – validates whether the newly developed or updated software works with older version of the environment
- Compatibility testing often involves running each test against multiple combinations of platforms / environments, which makes it a good candidate for automation

Security testing

- Main purpose of security testing is to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders
- Security testing is crucial part of software development, especially for Internet-facing (web) applications
- Common web application vulnerabilities: SQL injections, Cross-site scripting (XSS), Broken authentication & session management, etc.
- Types of security testing include:

  - **static analyses** - reviewing source code for potential vulnerabilities

  - **penetration testing** - simulating attacks against an application to identify actual vulnerabilities

Usability (UX) & Accessibility testing

- **Usability testing** goal is to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions
- Software usability / user experience (UX) is extremely important quality aspect that may significantly impact success of a software product
- Different techniques related to usability testing: checklists, questionnaires, A/B testing, etc.

- **Accessibility testing** is a sub-set of usability testing, performed to ensure that the application is usable by people with disabilities
- Testing techniques involve usage of wide range of tools (static analyses, screen readers, etc.) to acceptance-level tests done by people with disabilities
- Web Content Accessibility Guidelines (WCAG) – set of documents providing a single shared standard for web content accessibility

Localization and Internationalization testing

- **Localization** (l10n) testing validates whether software content adaptable to meet the cultural, lingual and other requirements of a specific region / locale
- Testing focuses on any locations in the software where customization of date, time, numeric formats, language and any other content specific for specific locale should be present

- **Internationalization** (i18n) testing validates if the design of the application allows for localization (i.e. be able to provide localization for any locale)

Resilience testing

- Resilience in software describes its ability to withstand stress and other challenging factors to continue performing its core functions and avoid loss of data.
- Resilience testing (sometimes referred to as ‘chaos testing’) is deliberate injection of failures (shutting down virtual machines / containers hosting the application, manipulating network traffic, cutting connection to database, etc.) to determine how the software handles such situations
- Test results help software engineers design the system to better handle such situations

Testing types - Positive & Negative testing

- **Positive** or **“Happy Path” testing** objective is to test application’s main positive flows.
- It does not look for negative or error conditions. The focus is only on the valid and positive inputs through which application generates the expected output.

- **Negative testing** ensures that the application can gracefully handle invalid input or unexpected user behavior
- For example: if a user tries to type a letter in a numeric field, an error message should be displayed
- Negative testing helps improve the quality of the application by finding its weak points

Testing types - Front-end & Back-end testing

- **Front-end testing** – testing the user interface of a software application
- Front-end tests usually include validations on: functionality, visualization, usability, accessibility, performance, compatibility, security of the user interface components of the system

- **Back-end testing** – testing ‘behind’ the user interface, i.e. not using the UI to validate behavior of other system components (server, database)
- Examples of back-end tests:

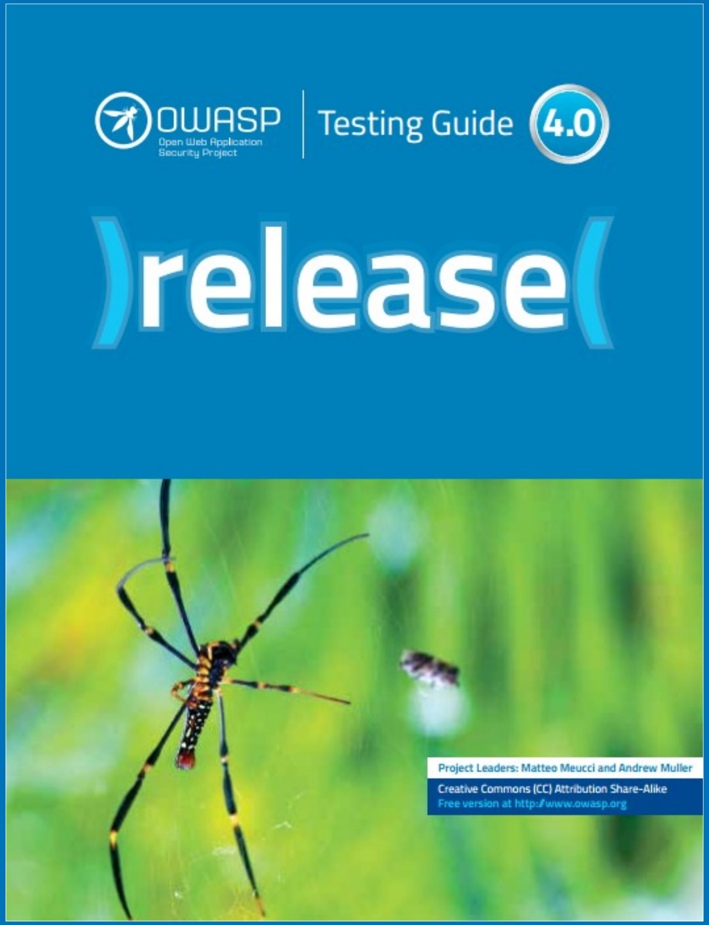
  - API testing – testing functional and non-functional aspects of API services

  - Database testing - validating data integrity database structure, procedures, etc.

Manual & Automation testing

- Software can be tested either manually (validation is performed by a human) or automatically (validation is performed by a machine)
- **Manual testing** is the process of using the functions and features of an application as an end-user would, in order to verify the software is working as required
- Some types of testing require mandatory manual execution (for example: exploratory testing, usability testing, etc.)
- **Automation testing** is the process of using machines (software tools) for running tests; advantages include reducing execution time and increasing accuracy
- Automation testing requires tests to exist as code (test scripts) that are executed and results validated against predefined expected results (assertions)
- **Semi-automation testing** is an approach which uses automation for helping manual execution; for example: automatically generating test data used in manual tests

Further reading



OWASP Open Web Application Security Project | Testing Guide 4.0

)release(

Project Leaders: Matteo Meucci and Andrew Muller

Creative Commons (CC) Attribution Share-Alike

Free version at http://www.owasp.org

SOFT ACADEMY