

Automated Testing

Software Testing, Test Automation, Types of Tests



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

Table of Contents

1. Software Testing Overview
2. Test Levels and Test Types
3. Test Automation
4. Integration Testing
5. API Testing
6. Web UI Test Automation





Software Testing Overview

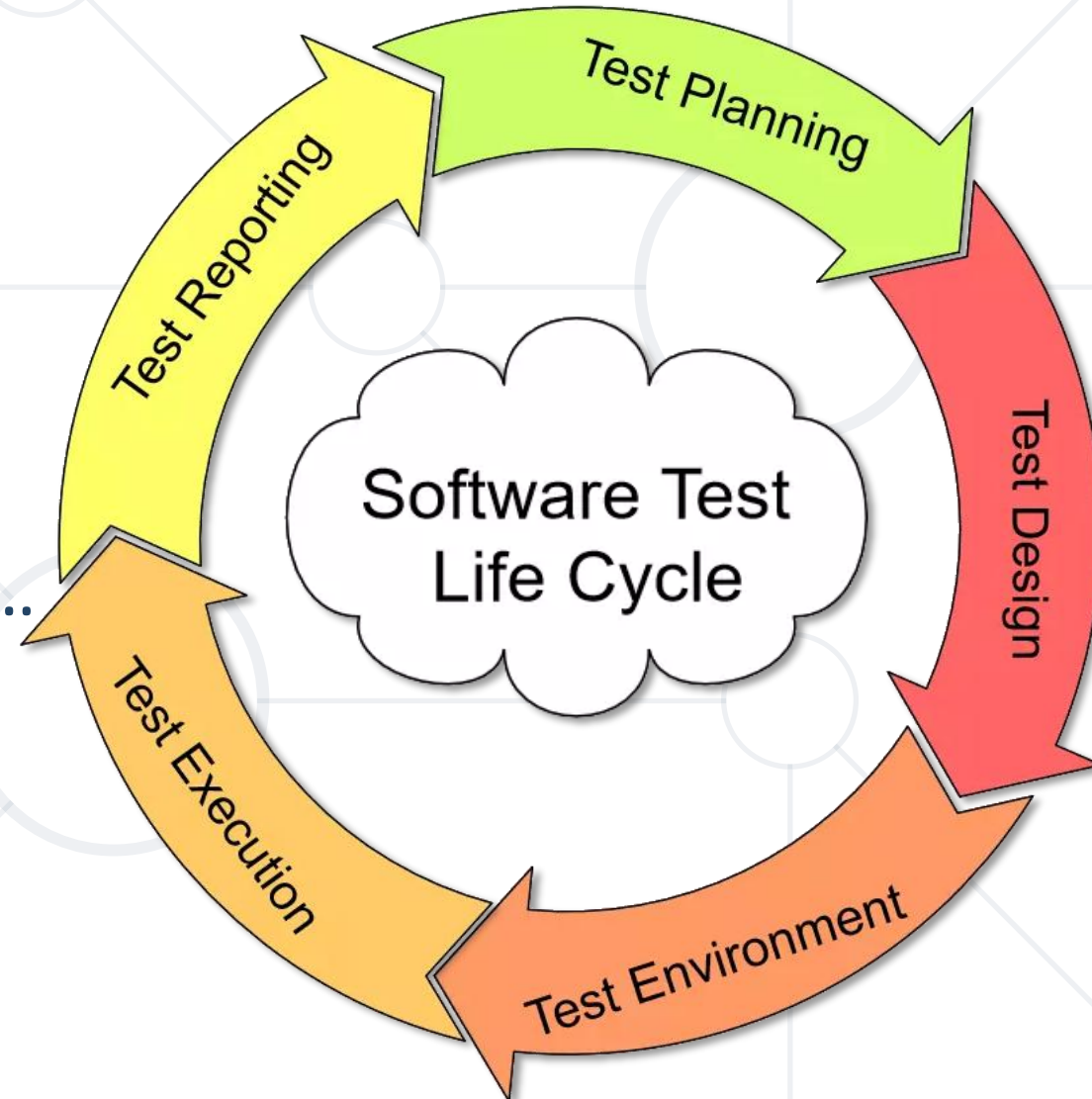
Finding Out How Well Something Works

- **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 software **requirements** (functional, non-functional, etc.)
- Testing aims to **find & report defects** (bugs)

- **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 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 (TDD)**
 - Writing actual test cases before writing the code
- Helps avoiding defects and makes code clearer
- Steps
 - **Write a test**
 - **Run the test**
 - **Write the code**
 - **Run the test**
 - **Refactor**

- Bug tracking tools are essential in **managing** and **maintaining** the **list of bugs**, reported during software testing
 - Easy reporting of defects
 - Enable categorization and prioritization of bugs
 - Provide utilities for documenting the steps to reproduce a bug
 - Allow tracking history and progress of each bug
- Commonly used tools
 - JIRA, Bugzilla, Trello, Asana, GitHub



Live Demo

Bug Tracking Tools

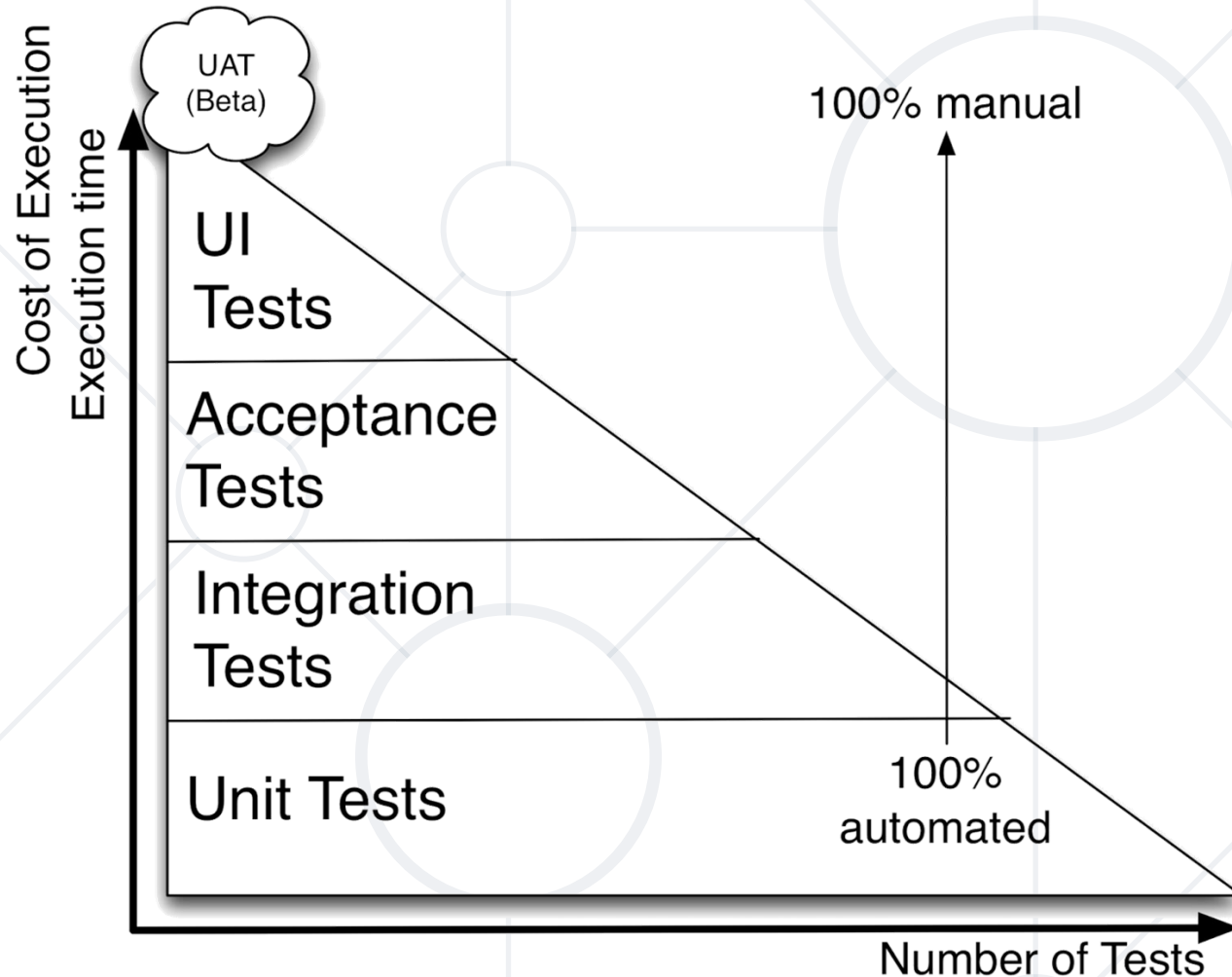


Test Levels and Test Types

The Step-by-Step Pyramid

- **Unit tests**
 - Test single component to ensure it works as expected in isolation
 - Typically, functions or methods
- **Integration tests**
 - Test interaction between components to verify they work together as intended
- **System tests**
 - Check the complete and integrated software to ensure it meets client's requirements
- **Acceptance tests**
 - Validate end-to-end business flow
 - Final verification to ensure the system meets the business needs

The Testing Pyramid

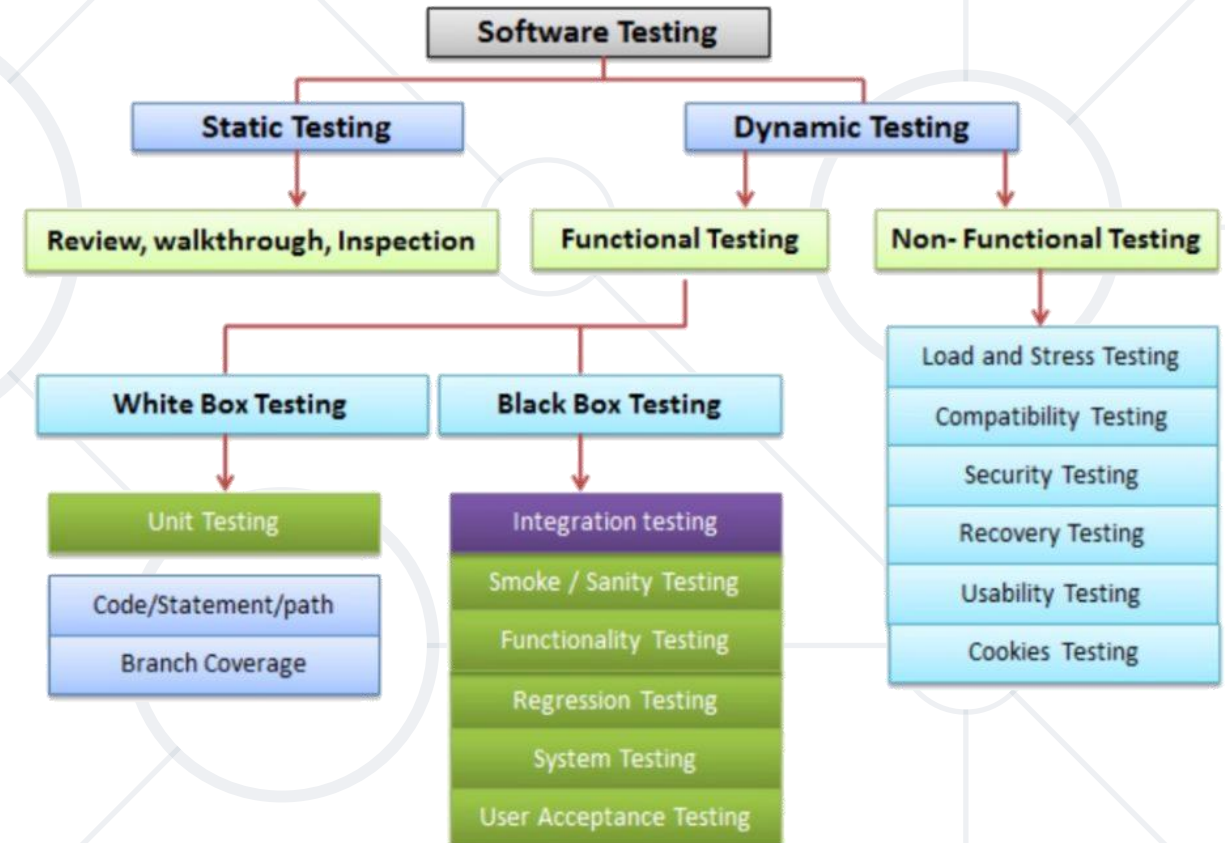


- **Unit tests**: fully automated
- **Integration tests**: fully automated
- **System tests / acceptance tests**: partially automated
- **UI / UX tests**: mostly manual

Test Types

- 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

Types of Software Testing:



- **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 introducing 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**

- 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**

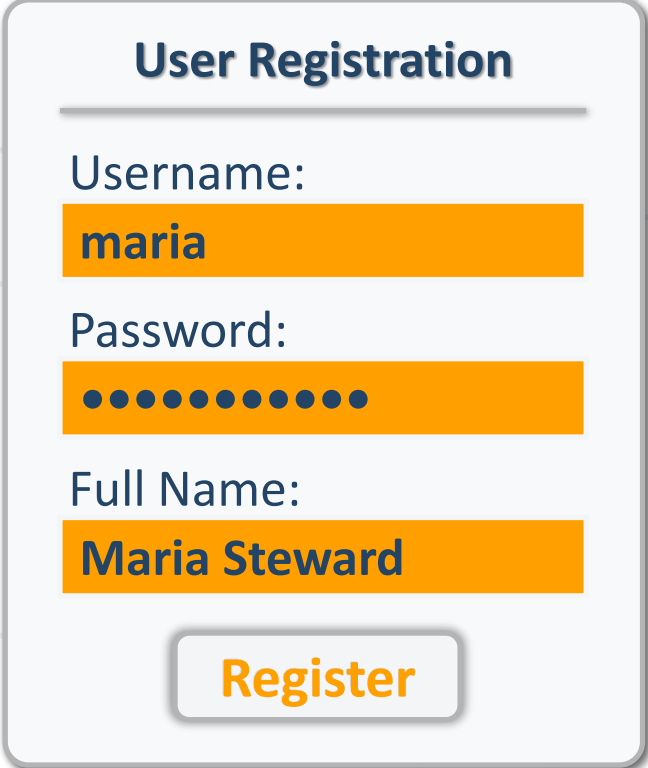
- A **positive** test
- A **negative** test

- **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

Register

Test Case – Formal Example

Project Name:	Google Email	 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 2. Enter Password 3. Click "Login" button	<Valid User Name> <Valid Password>	Successful login	Gmail inbox is shown		
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 2. Enter Password 3. Click "Login" button	<Valid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			
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 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Valid Password>	A message "The email and password you entered don't match" is shown			
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 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			



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

- **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

- **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

- Performed **after unit** testing and **before system** testing
- Identifies problems when **individual modules interact**
- **Incremental approach**
 - Two modules are tested 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 them together	Focus on individual components
Purpose	Validate that different parts of the application work correctly together	Validate functionality of separate parts of the application
Order in Dev Process	Performed after unit testing	First level of testing
Error Localization	Harder because of grouped components	Easier because of isolated components

- C# (ASP.NET Core)
 - xUnit, nUnit, Moq
- Java (Spring)
 - JUnit, Spring Test, Mockito
- JavaScript (Node.js/React/Vue)
 - Jest, Mocha (& Chai), Cypress
- Python
 - Pytest, unittest, Django Test Framework



Live Demo

Integration Testing



API Testing

Testing APIs

API Testing

- **API testing** tests APIs directly and as part of integration testing
- Crucial for verifying
 - Business logic
 - Interactions between different services
 - Microservices
 - Third-party APIs



- 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

- **Functional Testing**
 - Verify whether the API performs as expected
- **Load and Performance Testing**
 - Measures how the API performs under stress or high traffic
- **Security Testing**
 - Ensures the API is protected from vulnerabilities
- **Negative Testing**
 - Tests how the API handles invalid inputs or unexpected conditions

- **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**

- **Postman**
 - Used for manual and automated API testing
- **Swagger / OpenAPI**
 - Tools for generating API documentation and creating automated tests based on API definitions
- **RestAssured**
- **Newman**
 - Command-line tool used to run Postman collections in CI/CD pipelines



Live Demo

API Testing



Web UI Test Automation

Testing 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

- **Functional Testing**
 - Ensures that each element on the UI works as expected when interacted with
- **Cross-Browser Testing**
 - Validates that UI renders and functions correctly across different browsers
- **Responsive Testing**
 - Ensures UI adapts correctly to various screen sizes
- **Accessibility Testing**
- **Visual Regression Testing**

- 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)

- **Overview**

- Popular tool for automating browsers
- Useful for cross-browser testing
- Can be used with various programming languages

- **Selenium Family**

- Selenium IDE
- Selenium WebDriver
- Selenium Grid



Live Demo

Web UI Testing

- **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 tests** focus on UI and its functionality and performance



- 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>

