# Exercise: Test Levels and Test Types

This document defines the exercises and homework assignments for the [**"QA Basics" Course @SoftUni**](https://softuni.bg/opencourses/qa-basics).

Fill the provided **MS Word document template**. Put the **solution** for each exercise in the document. Name your document Test-Levels-Homework-FirstName-LastName.docx. Submit this document as your homework.

## Unit Testing in the Real Life: Testing a Battery

You are given a couple of **1.5V AA batteries**:

Graphical user interface, application

Description automatically generated

How could you test if they work as expected? Fill in the table what test you will perform.

### Hints

* Put the batteries into a flashlight (or other electric device) and **check if it works**.
* Check the **battery size** (height + diameter). Does it comply with the “**AA**” size standard?
* Check the battery **voltage**: measure the voltage using a digital multimeter.
* Check the **physical condition** of the battery: are there any damages?
* Check the **labels** on the battery. Are they accurate?

## Unit Testing in the Real Life: Testing a Bulb

You are given a **1.5V light bulb Е10**:

A close-up of a drop of water

Description automatically generated with low confidence

How could you **test if it works as expected**? Fill in the table what test you will perform.

## Unit Testing in the Software World: Age Checker

Let’s assume an **age checker function** that works as follows:

* If age is less than 13, returns “**child**”
* If age between 13 (inclusively) and 20 (exclusively), returns “**teenager**”
* If age between 20 (inclusively) and 65 (exclusively), returns “**adult**”
* If age is equal or bigger than 65, returns “**elder**”
* If the age is negative or above 150, returns “**error**”

Here are some examples:

* AgeChecker(5) 🡪 child
* AgeChecker(19.5) 🡪 teenager
* AgeChecker(20) 🡪 adult
* AgeChecker(75.3) 🡪 elder
* AgeChecker(-5) 🡪 error

You can try the age checker function here:

<http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com/age-checker/>

How will you **unit test this function**? Fill in the table what tests you will perform.

### Hints

Think of test cases to **cover each possible age range**. Think about the **border cases**.

## Unit Testing in the Software World: Income Checker

Let’s assume an **income checker function**, designed to categorizes given **monthly income** into one of the following categories: “**low**”, “**mid**”, “**high**. The income checker function works as follows:

* If the income is less than 1000, returns “**low**”
* If the income between 1000 (inclusively) and 3000 (exclusively), returns “**mid**”
* If the income s equal or bigger than 3000, returns “**high**”
* If the income is negative, returns “**error**”

Here are some examples:

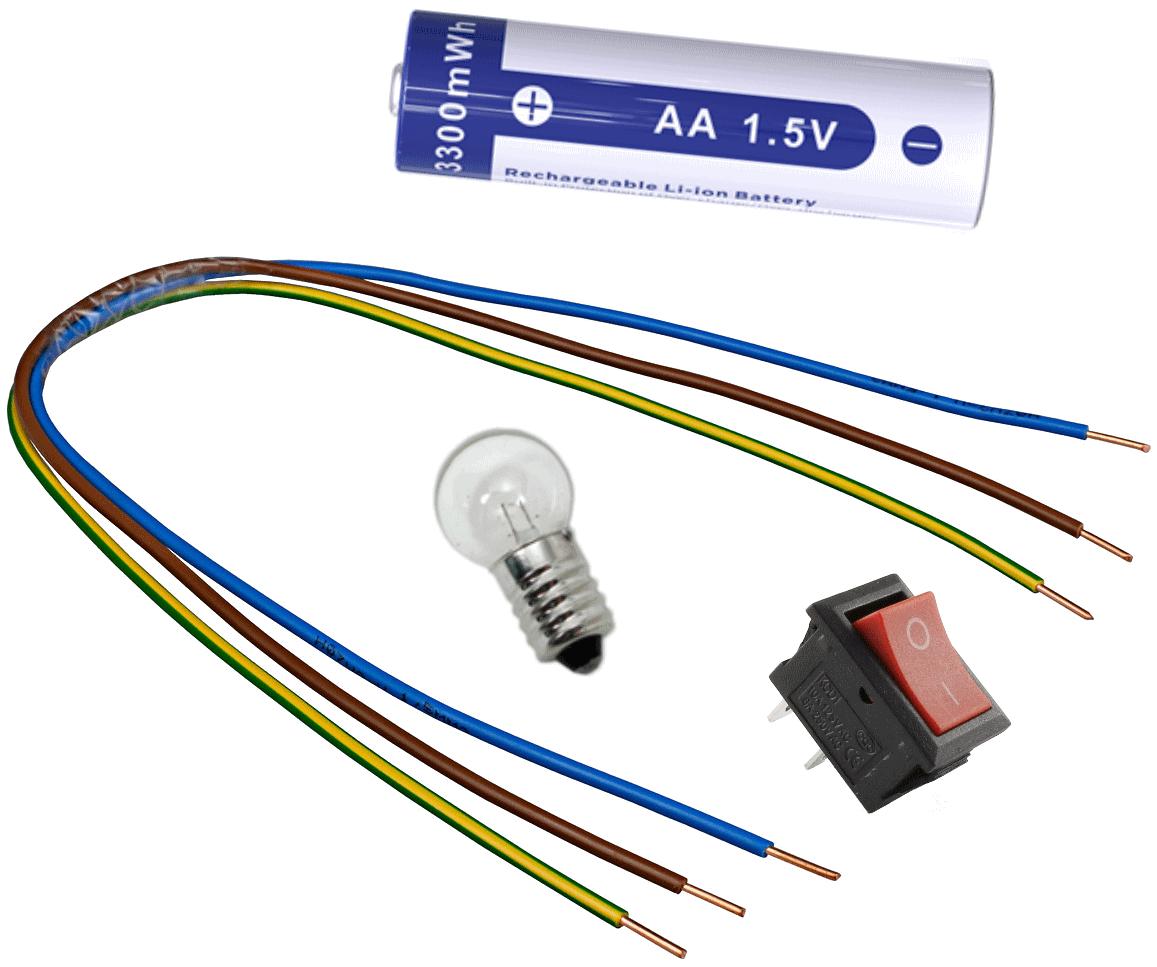
* IncomeChecker(250) 🡪 low
* IncomeChecker(1000) 🡪 mid
* IncomeChecker(2300.70) 🡪 mid
* IncomeChecke (7000) 🡪 high
* IncomeChecke (-5) 🡪 error

You can try the age checker function here:

<http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com/income-checker/>

How will you **unit test this function**? Fill in the table what test you will perform.

## Integration Testing in the Real Life: Lighting the Bulb

You are given the following components:

* **1.5V AA battery**
* **1.5V electrical bulb Е10**
* **Switch button**
* **Connecting wires**

All components are already **unit-tested**, and they work as expected.

Design **integration tests**, which verify that the bulb, batteries, switch button and wires work correctly together as a single electrical circuit.

### Hints

Your **first and most simple integration test** could use the following electrical circuit:

A picture containing shape

Description automatically generated

Your **second integration test** may involve the switch button. It could use the following electrical circuit:

Diagram

Description automatically generated

Test **negative** cases as well.

## \* Integration Testing in the Software World: Ads

You are given a Web app for publishing ads. It consists of:

* **Home Page**: shows all published ads. User can browse ads by categories and/or by towns. In order to publish an ad, the user should be registered. There is an option for the user to login or register.

Graphical user interface, website

Description automatically generated

* **Login Page**: implements user login. User can log in by providing correct username and password. There is a link to register

Graphical user interface, application

Description automatically generated

* **User Home Page**: available after successful user login 🡪 shows all published ads. User can browse ads by categories and/or by towns. There is an user navigation box with options to publish a new add, to view own published ads and to edit own profile. There is a “**Logout**” button.

Graphical user interface, application, website

Description automatically generated

Consider the above pages are different app **components** and they are already **unit-tested**. Your task is to design a set of integration tests, that verifies **the above 3 components work together correctly**.

### Hints

* You may check if after login the user home page is displayed correctly.
* Think about what should happen if someone tries to login with invalid credentials.
* Think about what should happen after logout.

## \* Integration Testing in the Software World: Credit Risk

Suppose you are given a function to calculate the **credit risk**, based on customer **age** and customer **income**. It should use the following already **existing components** (functions):

* **AgeChecker**(age) 🡪 child / teenager / adult / elder
* **IncomeChecker**(monthlyIncome) 🡪 low / mid / high

All components are already **unit-tested**, and they work as expected.

The **credit risk** is calculated using the following formulas:

* Age risk =
  + child 🡪 100%, teenager 🡪 60%, adult 🡪 10%, elder 🡪 20%
* Income risk =
  + low 🡪 50%, mid 🡪 30%, high 🡪 10%
* Credit risk =
  + ageRisk + incomeRisk - (ageRisk \* incomeRisk)

Examples:

* Credit risk for **14-year-old** person with **700 monthly income** is calculated as follows:
  + AgeChecker(14) 🡪 teenager 🡪 **ageRisk = 60%**
  + IncomeChecker(700) 🡪 low 🡪 **incomeRisk = 50%**
  + Credit risk = 60% + 50% - (60% \* 50%) = 110% - 30% = **80%**
* Credit risk for **85-year-old** person with **1600 monthly income** is calculated as follows:
  + AgeChecker(85) 🡪 elder 🡪 **ageRisk = 20%**
  + IncomeChecker(1600) 🡪 mid 🡪 **incomeRisk = 30%**
  + Credit risk = 20% + 30% - (20% \* 30%) = 50% - 6% = **44%**
* Credit risk for **30-year-old** person with **3500 monthly income** is calculated as follows:
  + AgeChecker(30) 🡪 adult 🡪 **ageRisk = 10%**
  + IncomeChecker(3500) 🡪 high 🡪 **incomeRisk = 10%**
  + Credit risk = 10% + 10% - (10% \* 10%) = 20% - 1% = **19%**
* Credit risk for **20-year-old** person with **-50 monthly income** is “**error**”, because the income cannot be negative.

You can try the credit risk checker function here:

<http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com/credit-risk/>

Design a set of **integration tests**, which checks that the credit risk calculator works as expected. Fill in the table what tests you will perform.

## System Testing in the Real Life: Flashlight

You are given a classical **electric flashlight**. Design a set of **system tests** to fully test the device.

### A picture containing light Description automatically generatedHints

* Test switch on / switch off the light
* Test battery replacement
* Test bulb replacement
* Test battery duration
* Test the illumination distance
* Shock resistance test
* Operation under high / low temperature

## System Testing in the Real Life: Digital Scale

You are given a classical **digital body weight scale**. Design a set of **system tests** to fully test the device.

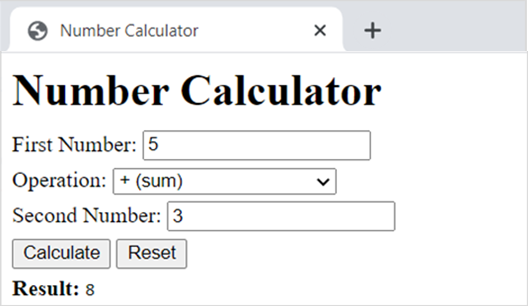
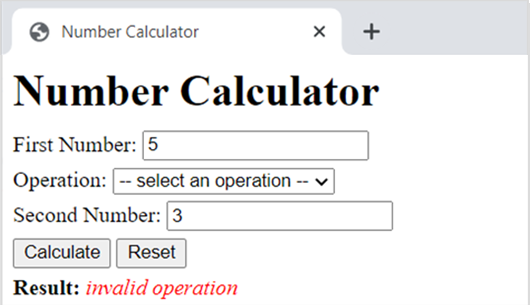
A picture containing indoor, device

Description automatically generated

## System Testing in the Software World: Number Calculator

You are given the following **number calculator app**:

<http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com/number-calculator/>

Design a set of **system tests** to fully test the app.

### Hints

You may consider testing with:

* Valid integers
* Valid decimal numbers
* Exponential numbers
* Infinity
* Invalid inputs
* Invalid operations

## Acceptance Testing in the Real Life: Flashlight

You are given a classical **electric flashlight**. Design a set of **acceptance tests** to test the device from a customer perspective.

Hints:

* The customer may take the flashlight, **switch on / off** the light, and assure it works.
* The customer may check the flash **illumination**.
* The customer may check how easy it is to **replace the batteries**.

## Acceptance Testing in the Real Life: Digital Scale

You are given a classical **digital body weight scale**. Design a set of **acceptance tests** to test the device from a customer perspective.

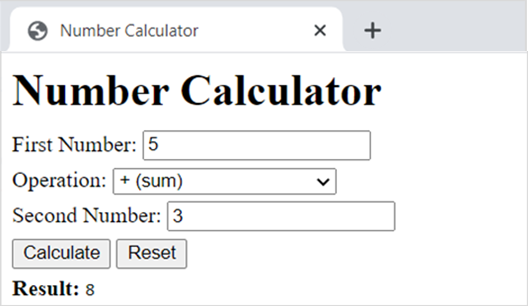
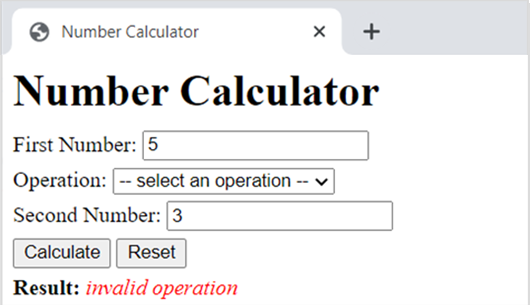
A picture containing indoor, device

Description automatically generated

## Acceptance Testing in the Software World: Number Calculator

You are given the following **number calculator app**:

<http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com/number-calculator/>

Design a set of **acceptance tests** to test the app from a customer perspective.

## Functional and Non-Functional Tests: Flashlight

A picture containing light

Description automatically generatedLook at all the tests that you have designed for the flashlight in the previous exercises (system and acceptance).

Can you divide them into **functional** and **non-functional**?