Mestrado em Engenharia Informática e Computação

Teste, Verificação e Validação de Software

2021/2022 | 1º Semestre

ACCEPTANCE TESTING

Exercises

Pedro Tavares | up201406991@edu.fe.up.pt

Tiago Verdade | up201704003@edu.fe.up.pt

Vitor Barbosa | up201703591@edu.fe.up.pt

Introduction

The exercises shown have been divided into three following separate chapters:

- *Implement acceptance tests using Cucumber for the isitFriday program.*
- *Write and implement acceptance tests for the given requirements in the BDD Style for an ATM system.*
- *Run a UI acceptance test for the Amazon website to validate the behavior required by the users.*

Before starting the exercises it is necessary to follow all the instructions from the installation guide.

Exercise 1 - Is it Friday?

Using the same project from the setup, verify you don't have a test that validates that Friday is Friday.

a. Add the following scenario to the ***isItFriday.feature*** file and add the necessary steps to the ***isItFriday_stepdefs.js*** file to pass this test.

```
Scenario: Friday is Friday
        Given today is Friday
        When I ask whether it's Friday yet
        Then I should be told "TGIF"
```

```
💡 Hint
   1. Only one Given step needs to be added, the When and Then steps
      provided in the setup can be reused.
   2. If you continue to fail the test after writing the Given step, check
      whether the behavior of the isItFriday function is correct.
```

b. We don't need to create a scenario for every day of the week. We will use **Scenario Outline** which allows us to run the same scenario several times with different combinations of values. Remove the previous scenarios and add a new scenario with the following description and change the steps in the ***isItFriday_stepdefs.js*** file according to this new test.

```
Scenario Outline: Today is or is not Friday
        Given today is "<day>"
        When I ask whether it's Friday yet
        Then I should be told "<answer>"
```

```
💡 Hint
  1. Change the Given step in the file isItFriday_stepdefs.js so that it
     receives a string argument similar to the Then step.
```

After completing the scenario and changing the respective steps add the following examples at the end of the file ***isItFriday.feature*** and run the tests to check if they all pass.

```
Examples:
       | day            | answer    |
       | Friday         | TGIF      |
       | Monday         | Nope      |
       | anything else! | Nope      |
```

Exercise 2 - ATM Machine

Consider that you are developing an ATM system. Using the **Behavioral Driven Development** principles, write the described scenarios for the following features:

1. As a customer, I want to login to an ATM, so that I can do transactions.
    a. Write a scenario that tests the behavior when the correct id and pin are entered by the user.
    b. Write a scenario that tests the behavior when the incorrect pin is provided by the user.
2. As a customer, I want to transfer money from my account to my friend's account, so that I can pay my debts.
    a. Write a scenario that tests the behavior when the transfer amount is less than the user's balance.
3. As a customer, I want to withdraw money from my account using an ATM, so that I can pay in cash.
    a. Write a scenario that tests the behavior when the user withdraws less than the user's balance.
    b. Write a scenario that tests the behavior when the user withdraws more than the user's balance.

💡 **Hint**

1. To avoid repeating the same steps in the feature's scenarios it may be interesting to use the keyword Background

Exercise 3 - ATM Machine using Cucumber

Using the scenarios created on the previous exercise, related to an ATM, implement the steps to validate the ATM behavior.

a. Copy your scenarios for the ***login.feature***, ***transfer.feature***, and ***withdraw.feature*** respectively
b. Implement the different steps in the ***atm_stepdefs.js***.

💡 **Hint**

1. In this exercise, you will need to create an instance of a Bank and an ATM before running the next steps. To do so, you can use the keyword **Before** to run a function before all the other steps.

```
Before({ tags: "@atm" }, function () {
  this.bank = new Bank();
  this.atm = new ATM(this.bank);
});
```

Exercise 4 - UI Acceptance Test

For this exercise we want you to complete an automated UI acceptance test for Amazon.com. For this purpose, we shall use **Cypress** to create scripts for the automation client to run. Here's an example script with the following requirements:

a. As a user, I want to open Amazon.com and search for a product, so that I can choose a product from a list.

We begin by creating our ***search-product.feature***, followed by our required scenario with respective Given, When, and Then clauses.

```
Feature: Search the product

  Scenario: with the product name
    Given I open the Amazon page
    When I type the “Nintendo Switch”
    When I click the search button
    Then “Nintendo Switch Lite - Yellow” should be listed
```

We can now run the example script written in **cypress/integration/step_definitions/search-product/** using the following line:

```
$ npm run cypress:open
```

With that, you should now be able to see Cypress run the test and observe if the requirements were met. Using the code from search-product as a reference, try to create tests for the following requirement:

1. As a user, I want to open Amazon.com and sign in, so that I can access my account and find new products.
    a. Write a scenario where the user does not enter an email and is given the message **“Enter your email or mobile phone number”.**
    b. Write a scenario where the user enters an invalid email address and is given the following message **“We cannot find an account with that email address”**.

```
💡 Hint
  1. In order to more easily filter the required HTML elements use the
     inspector tool
  2. Don’t forget to use cy.get(<El>) to get DOM elements, and functions
     click() and type(<text>) to click and write to DOM elements.
  3. If you need to see any method specification you can consult it here
```

Solutions

We can find our solutions here.