Minister of Education, Culture and Research of Moldova

Technical University of The Republic of Moldova

Software Engineering and Automatics

REPORT

Laboratory Work N.3
Software Testing

Performed by:
st. gr. FAF-161 V.Bantuș

Checked by: M.Catruc
a.univ.

Chisinau, 2019

Prerequisites:

- Selenium.
- Projection of Test Scenarios and Test Cases.
- Equivalence classes.
- Decision Table , State-Diagram and Transactions-Diagram.
- Black-Box technique.

Objectives:

- Understand the notion about scenarios.
- Ability to create and project the test cases.
- Understand the methodology of development the application:
    - Test-Driven development
    - Code-Driven development
- Forming the partitioning skills in the input data equivalence classes.
- Use decision tables and state and transition diagrams to create test cases.

Tasks:

1. Develop a test scenario for 2-3 functionalities of a software application.
2. Determining the criteria for organizing equivalence classes and decision tables.
3. To make a conclusive test according to these criteria including testing the limits.
4. To highlight test cases where erroneous results can be obtained.

Implementation of tasks:

Task 1:

Before performing this task I have read about scenarios and what is the purpose of them in Software Testing. So, the scenario is defined as any functionality that can be tested. It is also called **Test Condition or Test Possibility**. Also , I have read about the advantages of using the scenarios in process of testing the software.

The advantages of using scenarios are :

- Ensure the complete test coverage
- The application will work for the most common use cases.
- They help to determine the end-to-end transactions.
- Test scenarios are critical.

Scenario

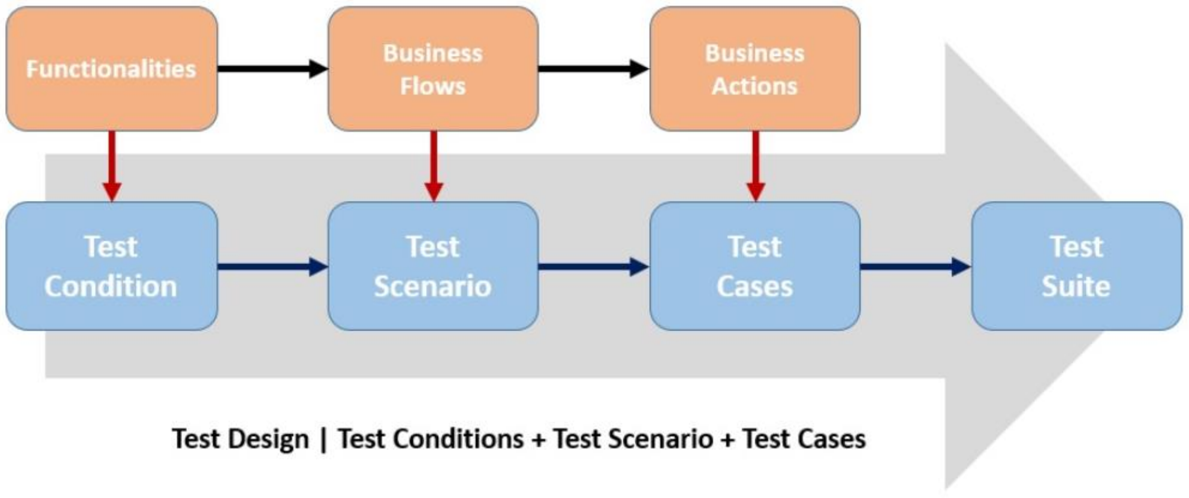


Fig.1: Scenario representation

So, for this laboratory work I chose to perform the scenarios for the **e-Commerce** platform.

The test scenarios are :

- **Check the login functionality.**

For this scenario I can enumerate some test cases that will describe the flow of the scenario:

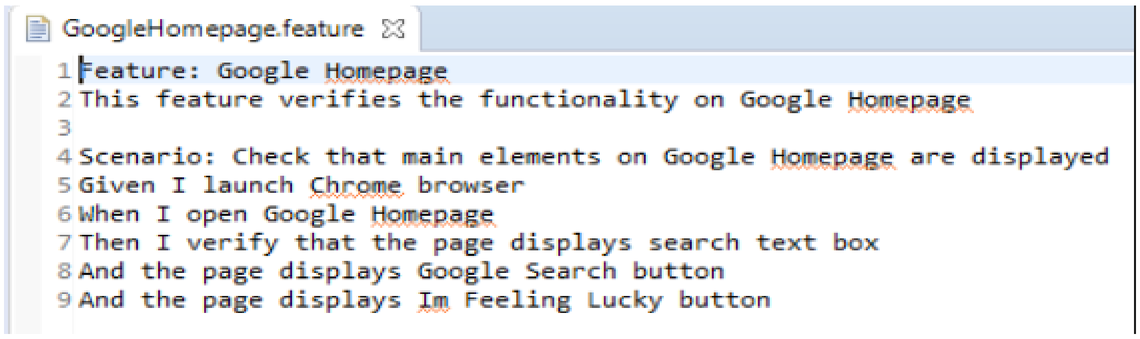
Check the system behavior:

- when valid email id and password is entered.
- when *invalid* email id and *valid* password is entered.
- when *valid* email id and *invalid* password is entered.
- when *invalid* email id and *invalid* password is entered.
- when email id and password are left blank and Sign in entered.
- Forgot your password is working as expected
- when valid/invalid phone number and password is entered.
- when "Keep me signed" is checked

- **Check the payments functionalities.**
  - when the valid/invalid card number is introduced.
  - when the valid/invalid CVV number is introduced.
  - when the valid/invalid Name and Surname are introduced.
  - when the amount of money are sufficient or not.
  - Check the system constraints , if this accept the all types of cards.

Also , I want mention that we can write scenarios in code, we have here the feature files and the testing technologies as Cucumber, Gherkin.

Scenario


GoogleHomepage.feature

```
Feature: Google Homepage
This feature verifies the functionality on Google Homepage

Scenario: Check that main elements on Google Homepage are displayed
Given I launch Chrome browser
When I open Google Homepage
Then I verify that the page displays search text box
And the page displays Google Search button
And the page displays Im Feeling Lucky button
```

Fig.2: Cucumber feature file, writing scenarios.

Task 2:

In this task we should to perform the equivalence classes and the table decision. This 2 parts of testing can be combined ore can be characterized separately. So, in my case of analyzing I can mention a lot of equivalence classes, based on different criteria.

One of the equivalence classes can be related to the amount of discount according to the amount of goods that person have purchased.

So , let's define the partitions according to the equivalence class. The condition in this case is to get the discount in the following way:

- User should get the 8% discount on purchasing goods from 100 to 500.
- User will get the 12% discount from 501 till infinity.

Now think on it, to just check the no discount condition means when user purchase a goods less than 100 then system should not give discount to him/her. To evaluate this for each and every condition tester need to write 100 test cases. For the second condition means when user purchase a goods in between 501 to 1000 goods, this requires 499 test cases. So this is impossible to write a huge number of tests. Here, we can say about the **Boundary value analysis** for solving the problem with this huge number of tests.

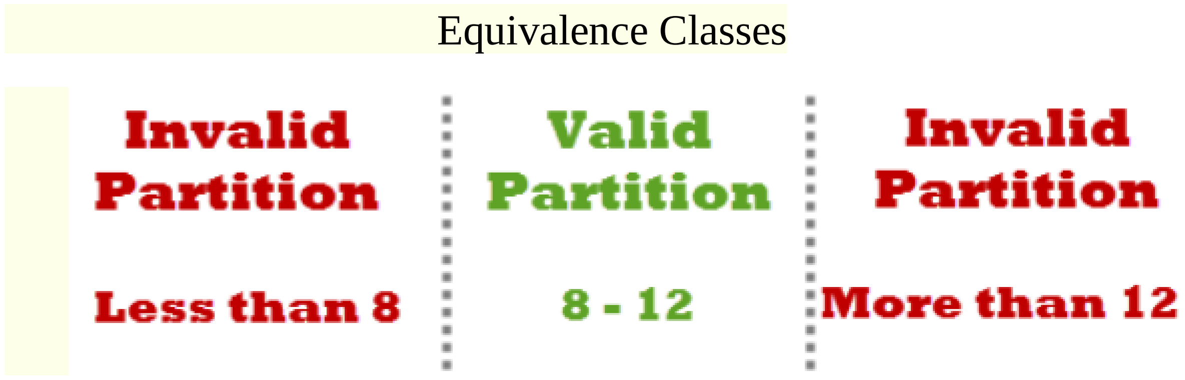


Fig.3: Equivalence classes representation

The second equivalence class in this case is that the user should have the state of the client for applying the discount policy.

Client – the user who has the account for 1 year.
User - the user which has the account less than 1 year.
Number of goods = G

Decision Table

Rules / Conditions	**R1**	**R2**	**R3**	**R4**
Client&&G=80	T	F	F	F
Client&&G=120	F	T	F	F
Client&&G=510	F	F	T	F
User&&G=amount	T	F	F	T
Result				
Discount	0%	8%	12%	0%

Task 3:

In this task I can perform a lot of Test Cases for testing the e-commerce platform, but I will describe several of them.

1. **General Test Cases :**
   a. Check that user is able to navigate through all the products across different categories.
   b. Check that all the links are redirecting to correct product/category pages and none of the links are broken.
   c. Check that the company logo is clearly visible.
   d. Check that all the text – product, category name, price and product description are clearly visible.
   e. Check that all the images are clearly visible.
   f. Check that category pages have a relevant product listed specifically for the category.
   g. Check that correct count of total products is listed on the category pages.

2. **Search Test Cases:**
    a. Check that the products displayed are related to what was searched for.
    b. Check that the products should display an image, name, price and maybe customer ratings and number of reviews.
    c. Check the more relevant product for the search term are displayed on the top for a particular search term.
    d. Check that all items in next page is different to the previous page, i.e. no duplicates.
    e. Check that when both sort and filter have been applied, they remain as we paginate or more products are loaded
    f. Check that count of products is correctly displayed on the search result page for a particular search term.
    g. Check that filtering functionality correctly filters product based on the filter applied.
    h. Check that filtering works correctly on category pages.
    i. Check that filtering works correctly on the search result page.
    j. Check that correct count of total products is displayed after a filter is applied.
    k. Check that all the sort options work correctly – correctly sort the products based on the sort option chosen.
    l. Check that sorting works correctly on the category pages.
    m. Check that sorting works correctly on the search result page.
    n. Check that sorting works correctly on the pages containing filtered result, after applying filters.
    o. Check that product count remains intact irrespective of sorting option applied.

Task 4:

I think that in every software we can develop each time new and new errors, because it is impossible to test the whole application , the all parts of it, the all smaller parts of the application. So, according to my topic I think that the error results can occur in some of the following cases :

1. Spelling and the grammatical part in each page. This means that should follow the International standards, also the text should be left justified.
2. Database Testing. Here the system can get saved the information in database in incorrect form. The each column should have primary key and an index(for quickly getting the information). Also the null value should not be allowed for the primary key columns.
3. Enter the invalid login or password can get an error result. This happen when you try to introduce some credentials, but in your database this user doesn't exist.
4. Path of the files. Could be the cases when the path is introduced wrong and the user will not get the file or the information in specific places.
5. Email sending. The each email should use standard CSS for all emails. The email address should be validated before sending.

Conclusion:

In this laboratory work I obtained skills and gained information about testing the web application and especially the E-Commerce platform. I learned about types of testing: QA and Security and Functional-Non-functional. Testing such of application we should lost a lot of time or even more time than develop it.

Also I have learned how to analyze the software, how to try to cover the whole application and project the minimal tests for checking the functionalities. Here, for creating the test for test not just the result, but also the implementation we can use the techniques of covering the 100% of (decisions, instructions, branches).