PalArch's Journal of Archaeology of Egypt / Egyptology

# TESTING THE INFORMATION SYSTEM SOFTWARE USING BEHAVIOR DRIVEN DEVELOPMENT METHOD

*Ucu Nugraha[1], Siti Atikah Nurduha Robaiah[2], Djoko Rospinoedji[3]*

Information System Department, Widyatama University, Indonesia

ucu.nugraha@widyatama.ac.id[1]

**Ucu Nugraha, Siti Atikah Nurduha Robaiah, Djoko Rospinoedji. Testing The Information System Software Using Behavior Driven Development Method--Palarch's Journal Of Archaeology Of Egypt/Egyptology 17(10), 3732-3742. ISSN 1567-214x**

**Keywords: Testing, Automated Testing, Behaviour Driven Development, Scenario Testing, Cucumber, Katalon Studio.**

**ABSTRACT:**
The testing phase is the stage that determines whether the software is in accordance with user requirements or not. The testing phase does not fully guarantee the quality of the software made but it can give more confidence to its users. In reality, during development, there were so many obstacles that resulted in the application not being maximally tested. Behaviour Driven Development (BDD) is a set of practices in agile software development that is designed to assist teams in building and delivering software that has more value, quality and speed. BDD can encourage collaboration between project team members and business teams, where all test scenarios are written in non-technical language that can be understood by all stakeholders. The purpose of this research is to implement BDD which is integrated with automation testing tools in a software development. Tools used in this research are Cucumber and Katalon Studio. With this research, testing is expected to be carried out more quickly and can bring all stakeholders, both technical and non-technical parties, into one table that is used to share knowledge about the system and testing requirements.

## INTRODUCTION

Along with the development of increasingly fast technology, various information services that appear to provide a lot of convenience for its users. An increase in the development of information technology has pushed software developers to find the right method of making information services. In making an information service needed clear direction on the features and functions of the application to be developed.Software development includes several stages including requirement gathering, analysis, design, implementation (coding), testing and maintenance. The testing phase is the stage that determines whether the software is in accordance with user

requirements or not. The testing phase does not fully guarantee the quality of the software made but it can give more confidence to its users. But in reality, during the development period, there were so many obstacles that resulted in the application not being maximally tested. Currently there are still many testers only testing in the traditional way, namely by trying one by one menu in the application when the application has been developed. This testing method takes a long time and is not accurate because the tester might forget to test one of the menus in the application. Especially if the project is done on a large scale, to test all the software requires a lot of time and resources. Automation testing is an indispensable tool in the midst of lack of resources, money and time to test an application. By using automation testing, the tester can do the test quickly and can be repeated.However, only relying on automation testing is not enough. When making automation scripts often the tester is confused about what to test, where to begin, how much to test and many more questions that arise. Behaviour Driven Development (BDD) method can be one solution to answer that question. By applying the BDD method, the development team can focus on the things that stakeholders / customers hope the software can do. The BDD method can facilitate the tester in making test scenarios that can be understood by various parties and ensure that the software workflow is in accordance with needs. All test scenarios will be made at the beginning of the project and agreed by all parties, it aims to reduce misunderstandings that can occur during the development process. By implementing BDD it means that all features already have a test scenario and this test will make it easier for the developer during maintenance.The purpose of this research is to apply the BDD method that is integrated with automation testing in a software development with a case study of making enterprise repair management (ERM) applications. The technology that will be used in this research is Cucumber and Katalon Studio. Cucumber is one of the tools used to run automated acceptance tests created in BDD format. Cucumber can be integrated with various desired automation tools and one of them is Katalon Studio. Integrating Cucumber and Katalon Studio can make it easily read by all parties, both technical and non-technical. Because writing Cucumber uses Gherkin language which allows the tester to write test scripts using natural language. So hopefully there are no more obstacles when non-technical parties want to see the automation testing script that has been made by the tester and testing can be done more quickly.

## *Literature Review*

### *A. Information System*

An information system is a collection of several interrelated components to produce certain information, thus forming an organized data collection along with procedures for its users [1]. An increase in information technology services requires software developers to create quality technology. At present the use of information technology, especially the internet has brought everyone to carry out various activities more accurately and quickly. So that technology becomes an important thing to be applied in achieving the effectiveness and efficiency of information management [2].

### *B. Testing*

Testing is the process of evaluating a system or its components with the aim of discovering whether the system has been free from security holes, program errors and meets specified standards. According to ANSI / IEEE 1059 Standards, testing can be defined as the process of analysis a piece of software to ensure that the software is made as you wish and free of defects / errors / bugs [3].

Testing is one of the stages in software development that aims to find errors in applications that have been made by developers. When testing if no errors are found it does not mean the application is very good. However, it is possible that the technique used is not good enough to detect errors. The testing phase does not fully guarantee the quality of the software made but it can give more confidence to its users.

### *C. Automated Testing*

Automated testing is one of the techniques in software testing that is run with the help of tools, scripts and software. Automated tests rely on test scripts that run automatically. The test script is based on a previously created test case. Automated tests can be done in a more effective and efficient way compared to manual testing. With automated tests, it can make it easier for testers to carry out repeated testing especially if the applications being tested are large scale. Along with the increased ability of automated tests, it can offset the high demand for making applications that are fast and inexpensive [4].

Software testing is an important step, but can be time consuming and expensive. Automation testing becomes an indispensable tool in the midst of a lack of resources, costs and time needed to do testing on a system. In a project there are three important factors including time, cost and quality. By implementing automation, it is expected to reduce the time and cost incurred to carry out testing while taking into account the quality of the resulting application [5].

A testing tool might not be enough to solve all the problems that might occur. With effective communication, automated and manual testing will complement and improve the quality of the application being made [6]. The type of testing (manual or automation) used depends on various factors such as project requirements, budget, timeline, etc. Usually automation testing is used for projects that have more repetitive work (for example, need to do the same regression testing repeatedly). While manual testing is usually used for projects that have requirements that change continuously [7]. Whatever type of test is used aims to complete the project that was created successfully while maintaining quality results.

### *D. Behaviour Driven Development*

Behaviour Driven Development (BDD) is a series of practices in agile software development that are designed to assist teams in building and delivering software that has more value, quality and speed [8]. This method focuses on things that stakeholders hope the system can do, in addition BDD can encourage collaboration between project team members and business

teams [9]. BDD was first introduced by Dan North in the early to mid-2000s as an easier way to teach and practice Test-Driven Development (TDD) [8].

The BDD method follows the principle of test-first, where scenarios are developed before software development. This scenario illustrates the activities that need to be carried out by users and has the advantage of displaying information so that it is easily understood by project team members and business teams. The advantage of BDD is that this method provides a common language based on simple and structured sentences expressed in English (in the native language of the stakeholders) which facilitates communication between project team members and the business team [8]. This will significantly reduce misunderstandings and eliminate unnecessary code and functional waste [10]. By implementing BDD means all features have a test scenario that can ensure that the software workflow is in accordance with user needs and will facilitate the developer during maintenance. Basically, BDD activities consist of three repetitive steps, including:
-Create a scenario that contains the functionality of the feature to be created along with details of what is expected to be done by the feature.
-Document the scenario in a way that can be automated and checked for approval.
-Implementation of the behaviour described by each documented scenario, starting with automated tests to guide code development.

Documented scenarios over time can become assets for team members when there is a change in software. The documentation reflects the mutual understanding between team members and this understanding will continue to develop along with the development of software that is built [11].

***E. Cucumber***

Cucumber is one of the automation tools framework that implements BDD (Behaviour Driven Development). This tool introduces a domain-specific language (DSL) named Gherkin which allows writing feature documentation written using natural language and can be implemented into automated testing using other languages [10].

The acceptance and integration specifications in BDD are written using the Given, When and Then steps. Given illustrates the initial context. When describes an event and Then describes the expected results, which serve as acceptance criteria. These specifications are called scenarios and are grouped into a story or feature [9]. Figure 1 is an example of writing specifications.

```
Scenario: Breaker guesses a word
  Given the Maker has chosen a word
  When the Breaker makes a guess
  Then the Maker is asked to score
```

Figure 1. Example specification [12]

Each scenario is a list of steps that Cucumber must complete. Cucumber will verify that the software complies with the specifications and produce reports that indicate success or failure for each scenario [12]. Figure 2 is a general description of how cucumber works.

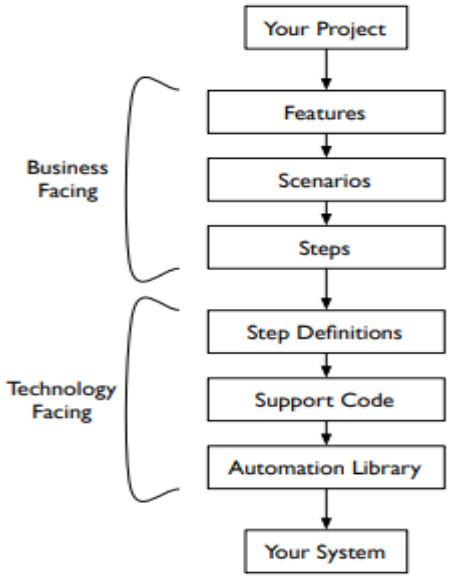


Figure 2. How cucumber works [13]

When running the cucumber command on the console, it will read the scenario in the Gherkin file (file extension. Feature). Each scenario in Gherkin is a collection of steps that will be mapped to steps definitions. Cucumber will generate reports that indicate success or failure for each step of the definition. In this way, it will be easier for the tester to find out the cause of the failure of the execution result [5].

### *F. Katalon Studio*

Katalon studio is an automation testing tool developed by Katalon LLC. Katalon Studio is built on the Selenium open source test automation framework, Appium with a special IDE interface for testing web applications, APIs, mobile and desktop applications. This allows developers and QA to organize, create, run, report and manage automated testing efficiently [14]. Katalon studio provides a dual interface that can be used to create test cases, including manual display and script display. Manual display is used for users who are less technical and script display is used for users who are experienced in writing complex syntax automation tests. So there are no more obstacles for the tester in making automation tests.

## *METHODOLOGY*

This research methodology uses the agile software development approach. The focus of the agile method is to reduce overhead in the software process and be able to quickly respond to changes without excessive rework. There are several terminologies in agile including, Test Driven Development (TDD), Acceptance Test Driven Development (ATDD) and Behaviour Driven Development (BDD). In this study, researchers used BDD terminology, tests

were written in non-technical language that could be understood by all stakeholders

In agile environment, BDD plays an important role because it strongly supports the use of agile during development and testing. Figure 3 is a summary of the BDD terminology cycle.

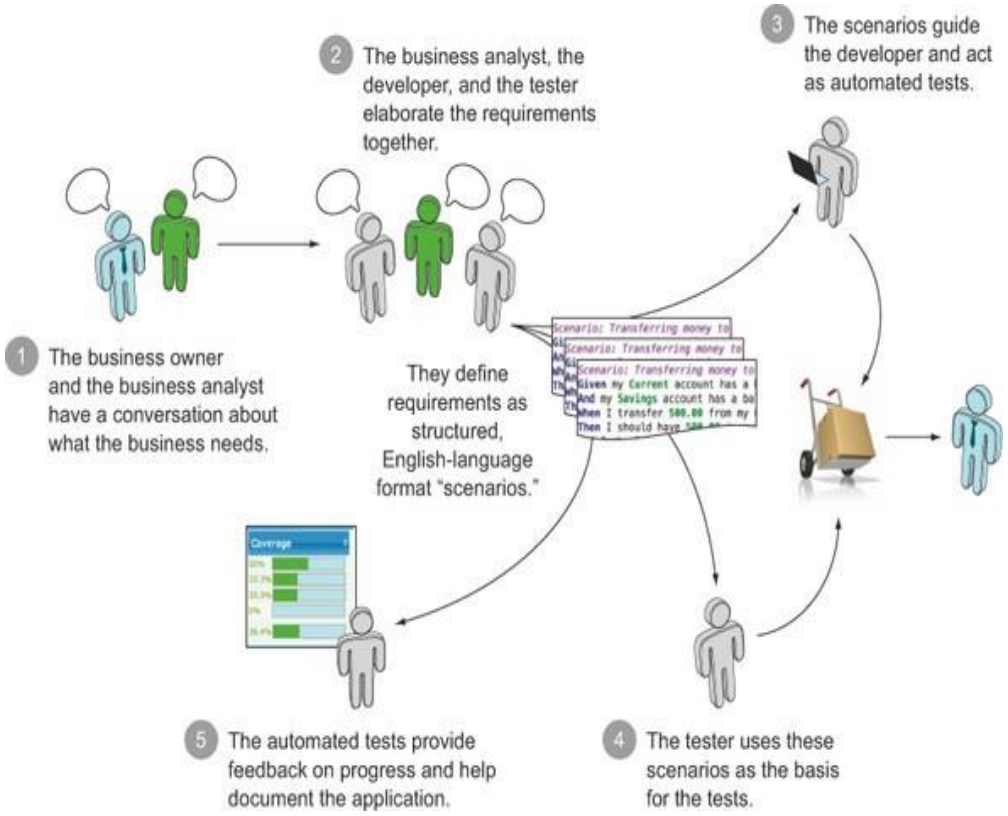


Figure 3. BDD Cycle [8]

BDD brings all stakeholders, both technical and non-technical parties, into one table that is used to share knowledge about the system and testing requirements. BDD consists of a series of steps that must be followed, including:

Identify business features.

Write a scenario along with the steps that need to be done for each scenario based on the features selected.

Implementation of test automation.

Run the test automation.

Generate test report.

## RESULTS AND DISCUSSION

In this section, the results of this study will be presented. The results to be displayed are the steps in making a test scenario, implementing test automation and the results of the test scenario.

### *A. Identification of Business Features*

In the enterprise repair management (ERM) application that will be tested, there are several features including login, repair order, repair assignment, repair action and QC.

### *B. Making Testing Scenarios*

Following are the test cases to test functionality in ERM applications.

Table 1. Scenario Test

| Test Scenario | Test Case | Test Steps | Expected Behavior |
|---|---|---|---|
| Check User Login | Log in as customer service | 1. Go to site ERM<br>2. Enter email: sitics@mailinator.com<br>3. Enter password: P@ssw0rd123<br>4. Click Sign In | Successfully logged in and redirected to the Dashboard page |
| Repair Order | Create repair order | 1.Click Create Repair Order Menu<br>2.Click button Create<br>3.Choose brand: Xiaomi<br>4.Input customer information: Siti Atikah<br>5.Input Carry in Date: 05-04-2020<br>6.Input device information: IMEI 12345<br>7.Add Problem: Power cannot power on<br>8.Click button Create RO | Data successfully saved and displays a device repair form |

### *C. Implementation of Automation Test*

To implement the test scenario that has been made into the automation test can be done in several stages including:

Make a script on Cucumber

The following are the tests, steps and scenarios that have been written using the BDD format in the Gherkin file (file extension. Feature).

```
Feature: Login Feature

  Scenario Outline: Log in as customer service
    Given User navigates to login page
    When User enters <email> and <password>
    And Click on Sign In button
    Then User is navigated to dashboard page

    Examples:
      | email                | password                 |
      | sitics@mailinator.com | iFGeFYmXIrU6ruIopQUS+w== |
```

Figure 4. Feature File

Creating a script automation on the Katalon

Script creation is done by the record & playback method provided by the Katalon. The results of the record will be the test stored by the Katalon and

make it a test case. Figure 5 is an example of the results of the record that has been done.

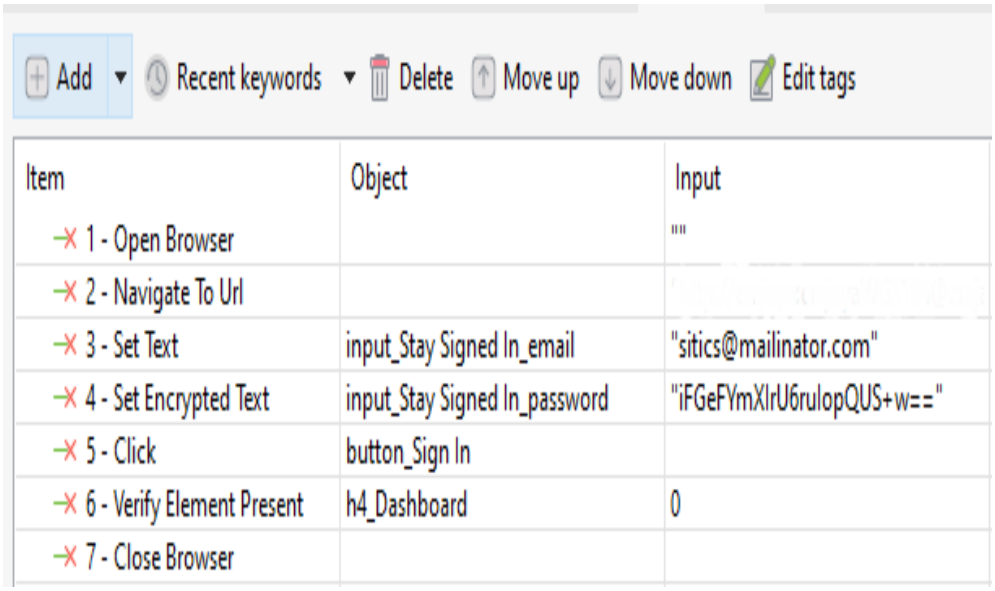


Figure 5. Test Case

Make Step Definitions

After creating a scenario on Gherkin and making a test case on the Katalon. The next step is to map the steps in the scenario into step definitions. This step aims to combine scenarios and test cases that have been made. Figure 6 is an example of making step definitions.

```
class LoginSteps {

    @Given("User navigates to login page")
    def navigateToLoginPage(){
        println ("\n I am inside navigateToLoginPage")
        WebUI.openBrowser('')
        WebUI.navigateToUrl('                                    /signin')
    }

    @When("User enters (.*) and (.*)")
    def enterCredentials(String email, String password){
        println("\n I am inside enterCredentials")
        println("Email : "+email)
        println("Password : "+password)
        WebUI.setText(findTestObject('Object Repository/Login/Page_Admin Login          /input_Stay Signed In_email'), email)

        WebUI.setEncryptedText(findTestObject('Object Repository/Login/Page_Admin Login  Erajaya/input_Stay Signed In_password'),
            password)
    }

    @And("Click on Sign In button")
    def clickSignIn(){
        println("\n I am inside clickSignIn")
        WebUI.click(findTestObject('Object Repository/Login/Page_Admin Login          /button_Sign In'))
    }

    @Then("User is navigated to dashboard page")
    def verifyHomePage(){
        println("\n I am inside verifyHomePage")
        WebUI.verifyElementPresent(findTestObject('Object Repository/Login/Page_Dashboard          /h4_Dashboard'), 0)
        WebUI.closeBrowser()
    }
}
```

Figure 6. Step definitions

Run the test automation
The following is an illustration when running test automation.

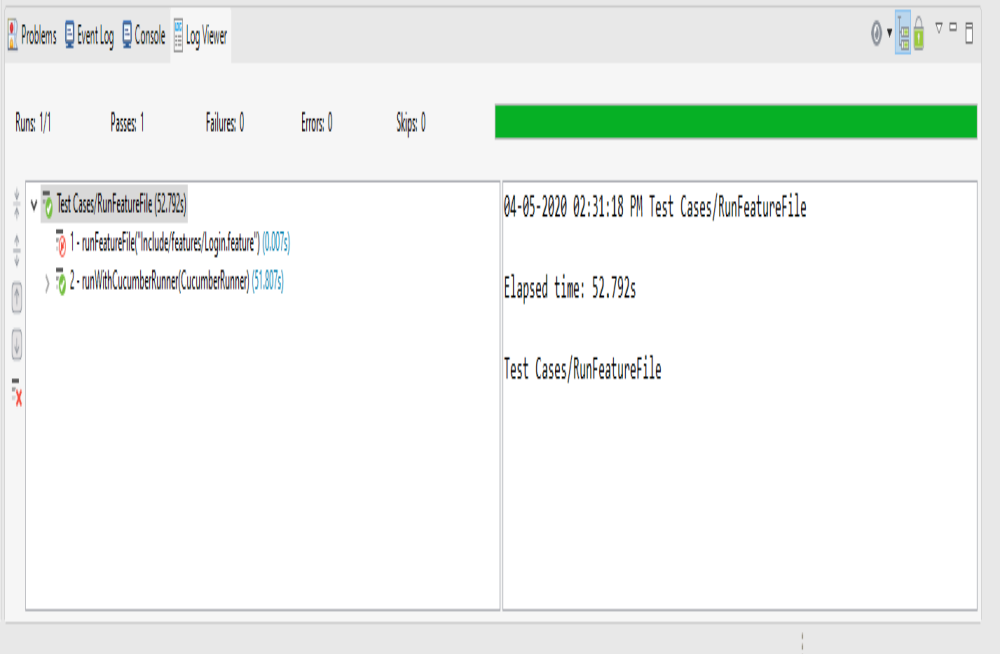


Figure 7. Run automation test
In the left column, you can see which test cases have been run along with the status information passed / failed. And in the right hand column you can see the time needed to run the test automation.

### *D. Generate Report*

The following is the report form generated in the form of html.

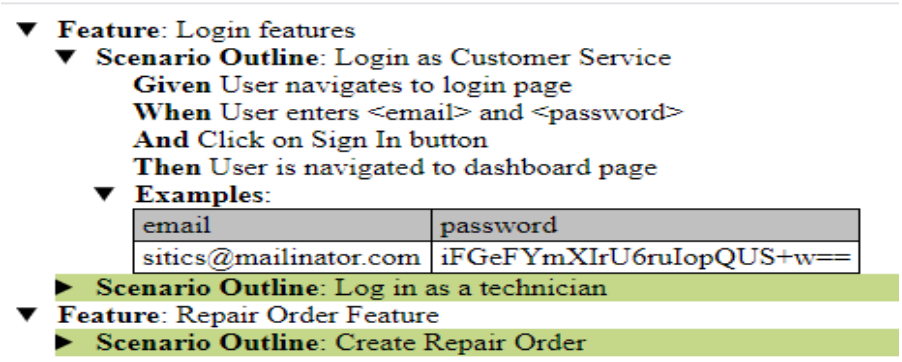


Figure 8. Report

The resulting report format can be changed according to user requirements. In the report can be seen what steps have been run by automation tools. If there is one step that fails, it will be marked in red along with a description of what caused the step to fail. And if all steps are successful it will be marked in green as shown in Figure 8. With such report format, all stakeholders can understand the results of the testing that has been done.

## CONCLUSION

Based on the discussion above, the following conclusions can be drawn:

-The use of Behaviour Driven Development (BDD) greatly helps the testing process and the development process. By applying BDD, all scenarios that will occur in the application are defined at the beginning of the project. This scenario can be used as a reference for developers when creating applications, as well as by testers when testing. So as to minimize misunderstandings during the development period. The scenario can be used to share knowledge about the system and testing requirement to all stakeholders involved in the project.

-The use of cucumber and Katalon studio can simplify and speed up testing. With the help of cucumber, script automation can be read easily so that non-technical parties can understand the script that has been made. And with the help of Katalon Studio, automation script creation can be done quickly by utilizing the record & play features that have been provided by Katalon Studio.

-However, using automation tools requires more effort than manual testing. But if developers and testers are used to using automation tools such as cucumber and Katalon studio, the positive impact of using automation testing tools will be more pronounced.

## REFERENCES

U. Nugraha, "Design of Information System for Population Data Collection Based on Client-Server at Bagolo Village," AIP Conference Proceedings, vol. 1855, no. 1, p. 060003, 2017.

B. Y. Ucu Nugraha, "REFERENCES Website and Mobile Based Emergency Reporting Information System," Journal of Advanced Research in Dynamical and Control System, vol. 11, pp. 957-967, 2019.

P. Roger S. Pressman, Rekayasa Perangkat Lunak (Pendekatan Praktisi). Andi Publisher, 2012.

J. R. J. P. Elfriede Dustin, Automated Software Testing: Introduction, Management and Performance, United States: Addison-Wesley, 1999.

L. J. Siagian, Otomatisasi Pengujian Perangkat Lunak (Software Test Automation), Yogyakarta: Deepublish, 2018.

M. W. Kanglin Li, Effective Software Test Automation - Developing an Automated Software Testing Tool, San Francisco: SYBEX, 2004.

S. S. Vishawjyoti, "Study And Analysis of Automation Testing Techniquest," Journal of Global Research in Computer Science, vol. 3, no. 12, pp. 36 - 43, 2012.

J. F. Smart, BDD in Action : Behavior-Driven Development for the whole software lifecycle, United States: Manning Publications Co., 2015.

G. R. V. M. D. S. E. A. Hugo Lopes Traves, "A tool stack for implementing Behaviour-Driven Development in Python Language," July 2010. https://www.researchgate.net/publication/45928540_A_tool_stack_for_implementing_Behaviour-Driven_Development_in_Python_Language.

W. Ye, Instant Cucumber BDD How-to, Packt Publishing, 2013.

Cucumber, "Behaviour-Driven Development," SMARTBEAR, https://cucumber.io/docs/bdd/.

Cucumber, "Introduction Cucumber," SmartBear, https://cucumber.io/docs/guides/overview/.

A. H. Matt Wynne, The Cucumber Book: Behaviour-Driven Development for Tester and Developers, United States: The Pragmatic Programmers, 2012.

Katalon Studio, "Welcome to Katalon Studio," Katalon LLC, https://docs.katalon.com/katalon-studio/docs/overview.html#documentations.