Comparative Study of Robot Framework and Cucumber as BDD Automated Testing Tools

Khaerunnisa[1], Nungki Selviandro[2], Rosa Reska Riskiana[3]
[1,2,3]Faculty of Informatics, Telkom University, Bandung, Indonesia
[1]khaerunnisa@students.telkomuniversity.ac.id, [2]nselviandro@telkomuniversity.ac.id, [3]rosareskaa@telkomuniversity.ac.id

Accepted 20 June 2023
Approved 11 July 2023

***Abstract*— Automation testing is much more efficient and accurate, the script is easy to document, and update compared to manual testing. Testing a website may necessitate time and effort to learn the tools to be used. Cucumber and Robot Framework are well-known open-source frameworks, according to Stack Overflow and GitHub. Cucumber and Robot Frameworks are known on an international scale, especially Robot Frameworks, which are often used by large companies. Each uses the Java and Python languages, which both support BDD. The comparative efforts of the two tools aim to help testers compare and determine automated testing tools in the BisOps Logee Port Web Admin case study based on the effectiveness and efficiency of the tools and create specific test cases as test documentation so that testers do not need to spend time analyzing both. Because this research involves evaluation and comparison, several criteria were chosen to support the evaluation process, namely functionality, reliability, usability, performance efficiency, and portability. The results of this study show that both tools can be recommended for novice QA's who want to learn the basics of automation by implementing BDD. Meanwhile, for QA's who have done automation before and want to do more in-depth configuration and reporting, it is recommended to use Robot Framework because the syntax is short, has lots of keywords that make it easier for testers, and can make the testing system shorter but more specific.**

***Index Terms*— automated-testing; behavior-driven development; comparative-evaluation; cucumber; Robot-framework**.

I. Introduction

Software product testing is one of the phases of the Software Development Life Cycle (SDLC), which aims to find errors in the source code that might cause bugs in software functionality. Testing can improve the quality of software. There are two types of testing, namely manual testing, and automation testing. Automation testing is far more efficient and accurate, and the script can be easily documented and updated compared to manual testing [1]. When automating testing on web-based applications, the tester may need to invest time and effort in learning the tools that will be used later [2].

The method used in this research is BDD. Behavior-Driven Development (BDD) is an agile software development methodology that assists teams in creating high-quality, fast-moving software [3]. BDD was first introduced by Dan North in the early 2000s as an easier way to teach and practice test-driven development (TDD) [4]. The BDD method was chosen because its main benefit is to facilitate organized communication within teams, meaning that product owners, developers, and testers will have a better-shared understanding of how the system works. Requirements written by the customer in a given-when-then format can be immediately used as a starting point for acceptance tests. This means that it is easier for non-developers to participate in the creation of acceptance tests [5]. Many tools that can be used for automation testing have been developed. There are at least 59 tools that can be used for automated testing. The first tool, Cucumber, was chosen because it is a well-known open-source testing tool that supports BDD. Cucumber has 3200 stars on GitHub and 634 forks. Robot Framework was chosen as the second tool because it is popular and supports BDD. Robot Framework has 5600 stars on GitHub and 1600 forks [5]. The first testing tool, Selenium Cucumber, was chosen because it is an internationally renowned framework and open source. The second test tool is the keyword-driven open-source framework (Robot Framework), which was also chosen because it is comprehensive and is used by many large international companies [6] and the robot framework is very easy to use in writing test scripts. The Robot Framework has a very modular architecture [7]. The two test tools are the most well-known testing tools according to the GitHub platform and support BDD, which will be used in this research.

In Juuso Jokio's research, namely "Test automation tools: Robot Framework vs. Selenium-Cucumber" [6], which focuses on automatic testing to test the functionality of the e-mail service, namely the login feature that is implemented in several browsers, there is no specific test case used by researchers. Therefore, in this study, the researcher uses the same framework as the previous study but with more specific test cases, not only for the login feature but also for other features, so that the researcher can see and analyze the differences between the two betters. Researchers tried comparative efforts on Cucumber and robot frameworks, which were carried out to assist testers in comparing and determining automated testing tool frameworks in the BisOps Logee Port Web Admin case study based on the effectiveness and efficiency of testing tools so that testers do not need to spend time trying and analyzing both, and testers making specific test cases as test documentation, which will help the testers in terms of trying and analyzing both. Because this research involves evaluation and comparison, several criteria were chosen to support the evaluation process, namely functionality, reliability, usability, performance efficiency, and portability [1].

The case study in this research refers to the Web Admin BisOps Logee Port, which is a web-based application for Internal Admin Operations that is useful for managing NPCT-1, NLE, and KOJA master data and is located in Indonesia. NPCT-1 is a web-based one-stop service platform for handling import and export containers and ordering fleets to and from Container Terminals; NLE is a web-based application for a logistics ecosystem that aligns the flow of international goods and documents traffic from the arrival of the means of transportation until the goods arrive at warehouses; KOJA is a web-based one-stop service platform for handling import and export containers and ordering fleets to and from Container Terminals.

This research effort was carried out to help the Logee port QA team (developers) in determining and comparing which tools are better and more efficient for the BisOps Logee Port Web Admin project so that the team does not need to spend time trialing and analyzing the automation tools to be used. Also, complete documentation will make it easier for future developers if they want to make improvements to the website, and developers can consider which features should be fixed, added, or even removed [8][9].

II. METHODOLOGY

This research will be carried out individually by researchers in two ways, namely, through observation and testing by researchers related to the comparative evaluation of automated testing tools. The testing tools that will be used in this study are the cucumber and robot frameworks by adopting Jureczko's research method, namely the process evaluation tool [10].“Fig. 1” below describes the research process stages.

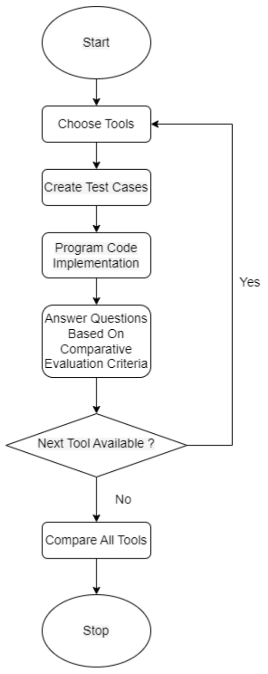

Start
Choose Tools
Create Test Cases
Program Code Implementation
Answer Questions Based On Comparative Evaluation Criteria
Next Tool Available ?
Yes
No
Compare All Tools
Stop

Fig. 1. Describes the Research Process Stages

The following is a detailed explanation of the stages of the research:

a) Choose Tools

There are two testing tools used in this study, namely Cucumber and the Robot Framework, which were evaluated using the Eclipse IDE. An IDE (integrated development environment) is needed as a basis for writing and running test cases and generating test script code. Eclipse IDE was chosen because it is a Java platform. The Java programming language is the most popular and frequently used lately [11]. IDE is an open-source, commonly used tool that can automate many functions that are usually written manually by developers and can be integrated with Cucumber and robot frameworks [12]. There are three main folders in the Cucumber project:

- \src\test\java which contains 2 more folders, page factory, and step definitions. The page factory folder is a folder used to write the function of the step in the step definitions folder based on the page or feature page being tested. Meanwhile, the step definitions folder is a folder that is used to write down the steps of the test in detail according to the test cases that have been made in the file .feature using the BDD method. In this folder, there is also a runner that is used if the tester wants to run a test, which will later be used to get a report from the test.
- \src\test\resources which contains the features folder, where this folder is the folder to hold all the feature files.
- The target folder is the folder that contains the report.

Meanwhile, there are two main folders in the robot framework project:

- Resources, in which there is a Pagefactory folder that is used to write the functions of the test cases in the test suite. The files in this folder contain *** Settings ***, *** Variables ***, and *** Keywords ***.
- Test Suite is a folder to hold files .robot is used to store test cases made using the BDD method. Files in this folder contain *** Settings *** and *** Test Cases ***.

b) Create Test Cases

In making test cases, both tools use the BDD principle, where the format of the test cases will be the same. In the first test tool, namely Cucumber, test cases are created in the feature file [13]. Whereas in the second test tool, namely the robot framework, test cases are made in the script file, namely the robot file [14].

Due to time constraints, the researcher chose to implement the main features of the Logee Port admin website. The selected test cases are the result of discussions with the Logee Port QA team regarding features that are often used in using the Logee Port admin website. There are six frequently used features: login features, dashboard menus, dashboards, containers, truck orders, and transactions. Of the 6 features, 12 test cases are often used and implemented in this study.

The following test cases are:

- feature to test login functionality.
- feature to test dashboard functionality Transaksi Per Periode.
- feature to test dashboard functionality Laporan Aktivitas Hari Ini untuk fitur belum dibayar.
- feature to test dashboard functionality Laporan Aktivitas Hari Ini untuk fitur telah dibayar.
- feature to test dashboard functionality Laporan Performa Bisnis.
- feature to test dashboard functionality armada.
- feature to test dashboard2 functionality lihat detail port.
- feature to test dashboard2 functionality lihat detail truk.
- feature to test transaksi functionality cari nomor proforma.
- feature to test the pemesanan truk page functionality.
- feature to test kontainer functionality.
- feature to test logout functionality.

c) Program Code Implementation

In Cucumber, the program code or test script code is implemented in the definitions step, which calls the main function on the page factory using the Java language, and all test cases defined in the feature file are mapped using annotations in the step definition file. Configuration for running test cases and reports is done on a test runner or file runner using Junit and Maven. Meanwhile, in the robot framework, the program code or test script code is implemented in resources using the Python language, and all test cases defined in the robot file are mapped using annotations in the resource file. configuration in running test cases and reports using the built-in robot framework feature, namely logs.

d) Answering Questions Based on Comparative Evaluation Criteria

The next step after the test cases have been implemented and executed is to answer questions based on comparative evaluation criteria. Questions are answered in two ways: through system testing and observation. The description of these two methods can be seen in “TABLE I” below. There are three modifications to the comparative evaluation questions referred to in Sandin's research [15].

The following are the three-modification explained:

- Modification of the functionality criteria in which Sandin's research reference implements the "unit testing" context while this research implements the "BDD testing" context.
- Elimination of one of the questions on the functionality criteria related to the number of methods of the two tools. This is done because based on observations, researchers do not use methods in their services. Researchers want to evaluate from a more objective standpoint. Therefore, as a substitute, the author describes the features in the analysis of the results.
- Adding questions to the usability criteria regarding the author's length of time studying and writing programs This is done because the author is using these two tools for the first time, and later this research can be useful for novice QAs who want to start automating.

“Table I” below explains modified comparison questions.

TABLE I. MODIFIED COMPARISON QUESTIONS [15].

Criteria	Questions	Possible Answers
Functionality	Test tool simplicity in BDD testing implementation? (Tested in system)	• Easy • Medium • Hard
	How many lines of code need to be executed for each case study? (Tested in system)	Lines of code

Reliability	Can the tool detect and perform error checking in any condition? (Tested in system)	• Possible • Not possible
Usability	Is the tool documentation provided available and can the user rely on that documentation to understand and learn about the tool used? (Tested based on observation)	• Easy (Available with many documents provided) • Medium (Available but needs more effort) • Hard (Available but need hard effort or not available at all)
	Can the user understand the code (test script)? (Tested in system)	• Easy • Medium • Hard
	How long has it taken me to study and write test code? (Tested in system)	Time to study and write the test code.
Performance efficiency	How long is the execution time to perform the past and failed testing and resource used? (Tested in system)	Execution time
Portability	Ease of installation.? (Tested based on observation)	• Easy • Medium • Hard
	Can the tool integrate with the development environment or in other words, run on different platforms? (Tested based on observation)	• Can run in many IDE • Can but in certain IDE only • Not portable

e) *Comparison of All Tools*

The comparison of the two test tools is done by comparing the performance results of the two test tools that have been executed based on test cases. After that, an analysis related to the comparison was carried out based on the questions that had been answered by the two testing tools. In the final step, the researcher makes a description and conclusion regarding the advantages and disadvantages of the testing tool.

III. Results and Discussion

In this section, the researcher evaluates the two test tools described in the previous chapter. The following are more detailed test results and analyses of test results:

a) *Test result*

"Table II" describes the results of the evaluation of the two tools used in this study, namely the cucumber and robot frameworks, based on the comparative evaluation criteria of the implemented test cases. The perspective reviewed is a more detailed matter for each criterion reviewed.

TABLE II. The Results of Evaluation of the Two Tools

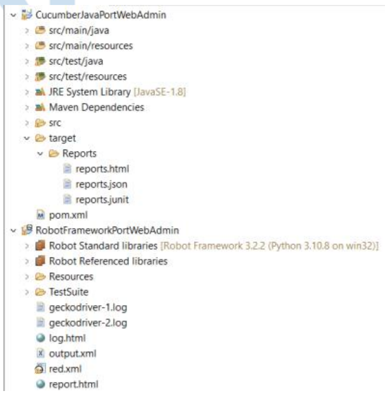
Criteria	**Aspects Considered**	**Tools**	
		Cucumber	**Robot framework**
Functionality	Test Implementation	Easy	Easy
	Lines of code	1.379 lines	567 lines
Reliability	Error checking	Possible	Possible
Usability	Documentation and learning	Easy	Easy
	Code readability	Medium	Easy
	Time to study and write the test code	16 days	10 days
Performance Efficiency	Execution time (seconds)	491.314 seconds (8 minutes 11 seconds)	276.438 seconds (4 minutes 36 seconds)
Portability	Ease of installation	Easy	Easy
	Integrated development environment	Can but in certain IDE only	Can run on many IDE

b) *Analysis of Test Results*

1) *Functionality*

In its implementation, the test cases and test scripts for both frameworks, namely the Cucumber and Robot Frameworks, adopted the BDD principles almost the same. Cucumber requires a file .feature to store BDD test scripts, while the robot framework requires a file .robot to store BDD test scripts.

The basic difference is the language used; Cucumber uses Java language. Meanwhile, the robot framework uses Python language. "Fig. 2" below shows cucumber with Java libraries and robot framework with Python libraries.



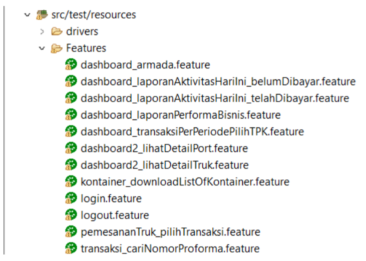
```
CucumberJavaPortWebAdmin
  src/main/java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
    Reports
      reports.html
      reports.json
      reports.junit
  pom.xml
RobotFrameworkPortWebAdmin
  Robot Standard libraries [Robot Framework 3.2.2 (Python 3.10.8 on win32)]
  Robot Referenced libraries
  Resources
  TestSuite
  geckodriver-1.log
  geckodriver-2.log
  log.html
  output.xml
  red.xml
  report.html
```

Fig. 2. Cucumber with Java Libraries and Robot Framework with Python Libraries

Another difference lies in the use of the library. The Cucumber library facilitates the implementation of annotations and methods that are implemented in test cases and test scripts. There are 10 annotations used in this study, which are the basic annotations needed so that test cases can be run. Meanwhile, the robot framework uses a keyword-based standard library, which in its implementation allows BDD test scripts that have been made before to be used as keywords and called back by other functions without the need for a constructor like the one in Cucumber. The keywords used in this study are 17; these are the basic keywords needed for the test cases to run.

In running one test case, the cucumber requires 3 files, namely:

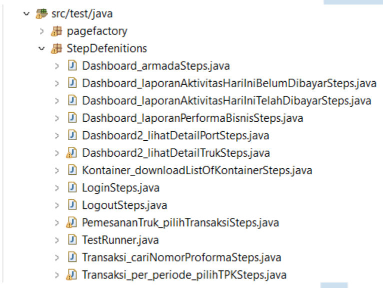
- File .feature (for writing BDD scripts, scenarios, or test scripts). "Fig. 3" below shows the cucumber file .feature.



```
src/test/resources
  drivers
  Features
    dashboard_armada.feature
    dashboard_laporanAktivitasHariIni_belumDibayar.feature
    dashboard_laporanAktivitasHariIni_telahDibayar.feature
    dashboard_laporanPerformaBisnis.feature
    dashboard_transaksiPerPeriodePilihTPK.feature
    dashboard2_lihatDetailPort.feature
    dashboard2_lihatDetailTruk.feature
    kontainer_downloadListOfKontainer.feature
    login.feature
    logout.feature
    pemesananTruk_pilihTransaksi.feature
    transaksi_cariNomorProforma.feature
```

Fig. 3. Cucumber File.Feature

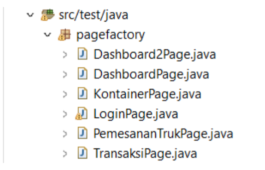
- file.java in the step definition folder (to write the implementation of BDD scripts that have been made in the file .feature). "Fig. 4" below shows the cucumber step definitions folder.



```
src/test/java
  pagefactory
  StepDefenitions
    Dashboard_armadaSteps.java
    Dashboard_laporanAktivitasHariIniBelumDibayarSteps.java
    Dashboard_laporanAktivitasHariIniTelahDibayarSteps.java
    Dashboard_laporanPerformaBisnisSteps.java
    Dashboard2_lihatDetailPortSteps.java
    Dashboard2_lihatDetailTrukSteps.java
    Kontainer_downloadListOfKontainerSteps.java
    LoginSteps.java
    LogoutSteps.java
    PemesananTruk_pilihTransaksiSteps.java
    TestRunner.java
    Transaksi_cariNomorProformaSteps.java
    Transaksi_per_periode_pilihTPKSteps.java
```

Fig. 4. Cucumber Step Definitions Folder

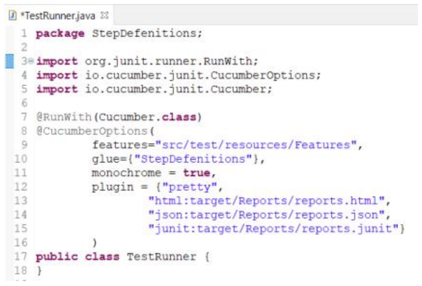
- file.java in the page factory folder (to write the functions that will be called in implementing the BDD script in the step definition folder). "Fig. 5" below shows the cucumber PageFactory folder.



```
src/test/java
  pagefactory
    Dashboard2Page.java
    DashboardPage.java
    KontainerPage.java
    LoginPage.java
    PemesananTrukPage.java
    TransaksiPage.java
```

Fig. 5. Cucumber PageFactory Folder

To run all test cases, one additional file is needed, namely the test runner (file.java, which contains the runner file so that the test cases can be run). The configuration of this test runner requires an annotation, namely @CucumberOption, which is shaped like an array of associations and can be changed according to needs. "Fig. 6" below shows the cucumber test runner.



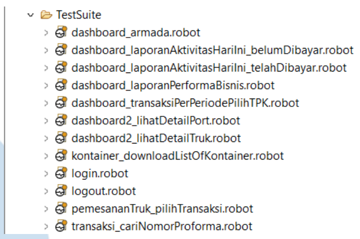
*TestRunner.java

```java
package StepDefenitions;

import org.junit.runner.RunWith;
import io.cucumber.junit.CucumberOptions;
import io.cucumber.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
        features="src/test/resources/Features",
        glue={"StepDefenitions"},
        monochrome = true,
        plugin = {"pretty",
                "html:target/Reports/reports.html",
                "json:target/Reports/reports.json",
                "junit:target/Reports/reports.junit"}
        )
public class TestRunner {
}
```

Fig. 6. Cucumber Test Runner

Meanwhile, in running one or all the test cases on the robot framework, only two files .robot are needed, namely:

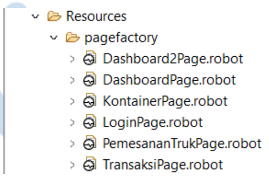
- File .robot in the test suite folder (to write BDD scripts, scenarios, or test scripts). "Fig. 7" below shows the Robot Framework Test Suite Folder.



```
TestSuite
  dashboard_armada.robot
  dashboard_laporanAktivitasHariIni_belumDibayar.robot
  dashboard_laporanAktivitasHariIni_telahDibayar.robot
  dashboard_laporanPerformaBisnis.robot
  dashboard_transaksiPerPeriodePilihTPK.robot
  dashboard2_lihatDetailPort.robot
  dashboard2_lihatDetailTruk.robot
  kontainer_downloadListOfKontainer.robot
  login.robot
  logout.robot
  pemesananTruk_pilihTransaksi.robot
  transaksi_cariNomorProforma.robot
```

Fig. 7. Robot framework TestSuite Folder

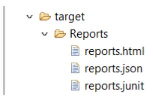
- File .robot in the resource folder (to write the implementation of the BDD script that was created in File .robot in the previous test suite folder, as well as the implementation of the functions needed in the BDD test script). "Fig. 8" below shows the robot framework Resources/PageFactory Folder.



```
Resources
  pagefactory
    Dashboard2Page.robot
    DashboardPage.robot
    KontainerPage.robot
    LoginPage.robot
    PemesananTrukPage.robot
    TransaksiPage.robot
```

Fig. 8. Robot framework Resources/PageFactory Folder

The last difference lies in the reporting. In Cucumber, in terms of generating reports, it is still done manually, namely by creating a manual folder and manually creating a .html file that will later be included in the test runner file so that later the report can be generated, and every time you want to open a report, you have to refresh it first so that the report or reporting can be updated to the file that was run most recently. "Fig. 9" below shows the cucumber Report Folder.



```
target
  Reports
    reports.html
    reports.json
    reports.junit
```

Fig. 9. Cucumber Report Folder

Meanwhile, in the robot framework, reports are generated automatically, so after installing the robot framework, the console will automatically have the message log and execution view options. In fact, after running the test case on the default console, a link will be given to see more detailed reporting. "Fig. 10" below shows the robot framework message log dan execution view console.

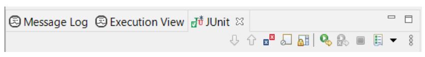
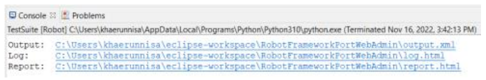


Fig. 10. Robot framework Message Log dan Execution View Console

"Fig. 11" below shows the robot framework default console.



```
TestSuite [Robot] C:\Users\khaerunnisa\AppData\Local\Programs\Python\Python310\python.exe (Terminated Nov 16, 2022, 3:42:13 PM)
Output:  C:\Users\khaerunnisa\eclipse-workspace\RobotFrameworkPortWebAdmin\output.xml
Log:     C:\Users\khaerunnisa\eclipse-workspace\RobotFrameworkPortWebAdmin\log.html
Report:  C:\Users\khaerunnisa\eclipse-workspace\RobotFrameworkPortWebAdmin\report.html
```

Fig. 11. Robot framework Default Console

There are 4 types of output in Cucumber: the default console, Junit console, HTML, and JSON. Whereas in the robot framework, there are six types of output: default console, console execution view, console message log, output.xml, log.html, and report.html. Cucumber only displays the BDD test script in the .html format output for cases with a green tick or a red cross symbol indicating whether a test case was successful or not. Whereas in the robot framework, there is a very informative log option, where in the output detailed information, such as time and others, is given, and if there is an error, a screenshot of the error page, along with the error code and some other additional information, is displayed.

2) *Reliability*

When implementing the library in test cases and code, there are no problems with the Cucumber and Robot frameworks. The problem that was found was the Selenium web driver, which executed the program too quickly while the browser was still in a loading state, which caused the test case to fail to run and the console to display the web driver used to reach out. This can be overcome by using the Selenium function itself, namely WebDriverWait(), which waits for a while before executing the next step, or you can also write code in the cucumber, namely Thread .sleep(millisecond); and on the robot framework, namely the sleep second keyword, which functions to wait for a few seconds before executing the next step according to the number of seconds' input.

3) *Usability*

Cucumber and robot frameworks have learning documentation or user guides regarding how to use them on their respective official websites.

Cucumber documentation can be accessed at the following link: https://docs.cucumber.io/docs/guides/ and robot framework documentation can be accessed at the following link: https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html. Both documents have good structure and detailed information. Both also have guides for beginners, such as in Cucumber's introduction and Robot Framework's getting started, and in both, there are examples of program code to give users a better understanding. When implementing the program code, both tools have end-to-end documentation. So, when hovering over the library or keywords, information about the libraries or keywords used will be displayed, which helps writers in writing program code using these two tools. As a result, it is possible to conclude that these two test tools are simple to document and learn. "Table III" below shows forum activeness comparison on stack overflow.

TABLE III. FORUM ACTIVENESS COMPARISON ON STACK OVERFLOW

Feature	Cucumber	Robot Framework
The whole question	10.585	6.499
Questions without answers	1.777	1.089
Questions without upvotes or answers are accepted	4.205	2.515

"Table IV" below shows a forum activeness comparison on GitHub.

TABLE IV. FORUM ACTIVENESS COMPARISON ON GITHUB

Feature	Cucumber	Robot Framework
Stars	2.507	7.500+
Watchers	223	484
Forks	2.000+	2.000+
Last Commit	15 November 2022 5.970 commits	12 November 2022 13.771 commits

In both tools, the program code can be read properly, as explained in the functionality sub-chapter. The difference lies in the difference in language, which makes the robot framework shorter because the syntax uses Python and because the robot framework doesn't require a test runner to run its test cases. The author takes 16 days to learn and write using the Cucumber tools and 10 days using the Robotframework tools.

4) *Performance Efficiency*

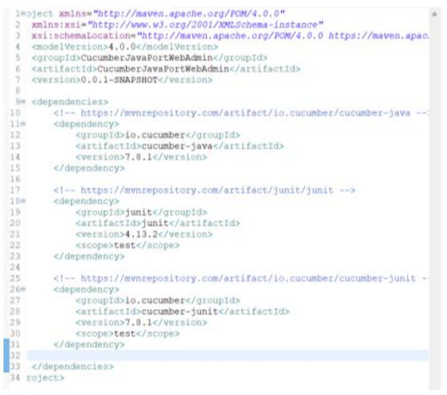
In the Sandin research, the test cases were run three times [15]. The author also does the same thing to find out the average execution time value of each tool or framework. Robot Framework executes test cases 2 times faster than Cucumber, this includes when Cucumber and Robot Framework generate reports automatically.

5) *Portability*

These two tools use different languages; therefore, the libraries and dependencies used are also different. The installation process or setup of the first test tool, namely Cucumber, can be seen as follows:

- Created a new Maven project.

- Adding Maven dependencies Cucumber Java | Cucumber JUnit | JUnit | Selenium Java. "Fig. 12" below shows cucumber dependency.

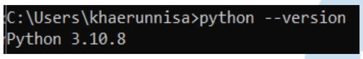


```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apac
  <modelVersion>4.0.0</modelVersion>
  <groupId>CucumberJavaPortWebAdmin</groupId>
  <artifactId>CucumberJavaPortWebAdmin</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
      <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
      <dependency>
          <groupId>io.cucumber</groupId>
          <artifactId>cucumber-java</artifactId>
          <version>7.8.1</version>
      </dependency>

      <!-- https://mvnrepository.com/artifact/junit/junit -->
      <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>4.13.2</version>
          <scope>test</scope>
      </dependency>

      <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -
      <dependency>
          <groupId>io.cucumber</groupId>
          <artifactId>cucumber-junit</artifactId>
          <version>7.8.1</version>
          <scope>test</scope>
      </dependency>

  </dependencies>
roject>
```

Fig. 12. Cucumber Dependency

The installation process or setup of the first test tool, namely the robot framework, can be seen as follows:

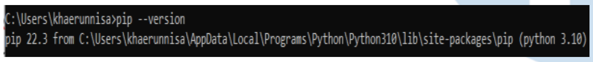
- Check if the device has Python, if not then install Python."Fig. 13" below shows Python version 3.10.8.



```
C:\Users\khaerunnisa>python --version
Python 3.10.8
```

Fig. 13. Python Version 3.10.8

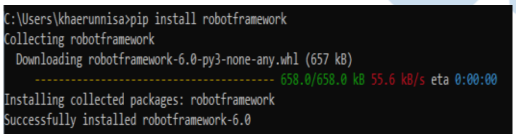
- Check if the device has pip, if not then install pip."Fig. 14" below shows pip version 22.3.



```
C:\Users\khaerunnisa>pip --version
pip 22.3 from C:\Users\khaerunnisa\AppData\Local\Programs\Python\Python310\lib\site-packages\pip (python 3.10)
```

Fig. 14. Pip Version 22.3

- In the command prompt, type "pip install robot framework to install robot framework" to install the robot framework. "Fig. 15" below shows robot framework version 6.0.

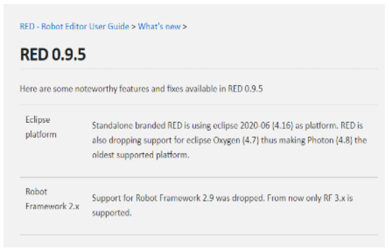


```
C:\Users\khaerunnisa>pip install robotframework
Collecting robotframework
  Downloading robotframework-6.0-py3-none-any.whl (657 kB)
     ---------------------------------------- 658.0/658.0 kB 55.6 kB/s eta 0:00:00
Installing collected packages: robotframework
Successfully installed robotframework-6.0
```

Fig. 15. Robot Framework Version 6.0

- Download Eclipse RED—the Robot Editor—from the Eclipse Marketplace.
- Added the path to RED in Eclipse / Windows / Preferences / RF / Installed FWs

The author was confused when trying to install the robot framework on the Eclipse IDE because, according to the robot framework user guide, the way to install it is to use the command prompt and input the pip install robot framework command. The version to be installed is the latest, namely version 6.0. Meanwhile, RED or the Eclipse robot editor can only support robot framework 3. x and Eclipse IDE version 2020-06 (4.16). Overall, the installation process for these two tools is easy. "Fig. 16" below shows the RED user guide.



RED - Robot Editor User Guide > What's new >

RED 0.9.5

Here are some noteworthy features and fixes available in RED 0.9.5

Eclipse platform	Standalone branded RED is using eclipse 2020-06 (4.16) as platform. RED is also dropping support for eclipse Oxygen (4.7) thus making Photon (4.8) the oldest supported platform.
Robot Framework 2.x	Support for Robot Framework 2.9 was dropped. From now only RF 3.x is supported.

Fig. 16. RED User Guide

Regarding integration with other platforms, based on what is stated on each tool's official website, Cucumber can only be implemented in the Eclipse IDE and IntelliJ IDEA and can also be implemented in the NetBeans IDE, which may still be under development. Meanwhile, robot framework can be implemented in almost any IDE, such as RIDE, sublime plugin, atom plugin, notepad++, IntelliSense for Visual Studio Code, and many more.

IV. Conclusion

Robot Framework is easier to use than Cucumber. This can be seen in the table of test results in Sub-Chapter 4. Where 5 differences make the robot framework superior, namely:

1. On the functionality criteria. The test script's lines of code require only 567 lines of code to run the 14 test cases discussed previously. Meanwhile, cucumber requires 1,379 lines of code to run the same test case.
2. On usability criteria. The readability of the program code on the cucumber has an intermediate status, while the robot framework has an easy status. This is because Cucumber uses the Java language and the robot framework uses Python, which makes the syntax shorter.
3. Still on usability criteria. The author's time spent studying and writing test code, where learning and writing code on Cucumber took the writer approximately 16 days, but only 10 days on the Robot Framework. This is because, apart from the short syntax of Python, you don't need to build a constructor to call one function to another like the one in Cucumber. With the keyword facility in the robot framework, it makes it easier for writers to call functions.
4. On Performance Efficiency criteria. The execution time of the Robot framework is quite short, at 4 minutes and 36 seconds, whereas the cucumber takes 8 minutes and 11 seconds to execute the same test case.
5. On the portability criteria, the robot framework can be integrated with almost any IDE

platform. Meanwhile, Cucumber can only be integrated into certain IDEs.

Although the Cucumber forum is more active on the Stack Overflow platform, with a total of 10,585 questions, the RobotFramework forum only has 6,499 questions. However, on the GitHub platform, the robot framework is far more popular, with more than 7,500 stars, while Cucumber only has 2,507 stars as of November 16, 2022. This shows that Robotframework is no less competitive than Cucumber in terms of popularity.

In terms of reporting, automatic reporting from the robot framework has very clear details, time information, error messages, and logs, even when an error occurs, a screenshot of the error page will be displayed, and many other features. which is not owned by the cucumber's automatic reporting.

The two tools in this study, Cucumber, and Robot Framework can be recommended for novice QAs who want to learn the basics of automation and implement automated BDD testing easily. However, specifically for QAs who have done automation before and want to do more in-depth configuration and reporting, the authors recommend using a robot framework because, in addition to having a short syntax, it also has many keywords that make it easier for testers and can make the testing system shorter but more specific.

Suggestions for future work are to make comparisons with different criteria and different case studies and to use more test cases to get a longer total time result so that the comparison can be seen more clearly.

ACKNOWLEDGMENT

This paper and the research behind it would not have been possible without the exceptional support of my lecturers, Nungki Selviandro, S. Kom, M. Kom, Ph.D., and Rosa Reska Riskiana, S.T., M.T.I. Their enthusiasm, knowledge, and exacting attention to detail have been an inspiration and kept my work on track. Their generosity and expertise have improved this study in innumerable ways and saved me from many errors.

REFERENCES

[1] SVS College of Engineering, Institute of Electrical and Electronics Engineers. Madras Section, and Institute of Electrical and Electronics Engineers, *Proceedings of 2017 Second IEEE International Conference on Electrical, Computer, and Communication Technologies : 22-24, February 2017, SVS College of Engineering, Coimbatore, Tamil Nadu, India*.

[2] T. Akhir, “Evaluasi Komparatif Automated Behavior Driven Development Testing Tool Framework pada Aplikasi Berbasis Web menggunakan Comparative Evaluation Criteria.

[3] T. Couto, S. Marczak, and F. Gomes, “On the Understanding of How to Measure the Benefits of Behavior-Driven Development Adoption: Preliminary Literature Results from a Grey Literature Study,” in *ACM International Conference Proceeding Series*, Dec. 2020. doi 10.1145/3439961.3440000.

[4] S M. Härlin, “Testing and Gherkin in agile projects.

[5] V. Österholm, “Overview of Behaviour-Driven Development tools for Web applications,” 2021.

[6] J. Jokio, “Test automation tools Robot Framework vs. Selenium-cucumber,” 2020.

[7] N. Huu and N. Triem, “RESEARCH AND COMPARE SOME FRAMEWORKS COMMONLY USED IN AUTOMATION TESTING GRADUATION THESIS SUMMARY INFORMATION TECHNOLOGY Student : Luong Khac Tuan Anh-ID: 17IT001,” 2017.

[8] U. Nugraha, S. Atikah Nurduha Robaiah, and D. Rospinoedji, “TESTING THE INFORMATION SYSTEM SOFTWARE USING BEHAVIOR DRIVEN DEVELOPMENT METHOD 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,” vol. 17, no. 10, 2020.

[9] H. Lopes Tavares, G. G. Rezende, V. Mota Dos Santos, R. Soares Manhães, and R. Atem De Carvalho, “A tool stack for implementing Behaviour-Driven Development in Python Language.” [Online]. Available: http://www.renapi.org/biblioteca-digital/ferramentas.

[10] M. Jureczko and M. Mlynarski, “Automated acceptance testing tools for web applications using Test-Driven Development,” 2010.

[11] K. Kumar and S. Dahiya, “Programming Languages: A Survey International Journal on Recent and Innovation Trends in Computing and Communication Programming Languages: A Survey”, [Online]. Available: http://www.ijritcc.org.

[12] “geer2005”.

[13] M. Wynne and A. Hellesøy, “What Readers Are Saying About..

[14] Sumit. Bisht, *Robot Framework Test Automation.* Packt Publishing, 2013.

[15] R. Mohamad and N. M. Yassin, “Comparative Evaluation of Automated Unit Testing Tool for PHP Development of Dengue-Entomological Surveillance System View project SBSE, meta-heuristic search algorithms and Artificial immune system View project.” [Online]. Available: https://www.researchgate.net/publication/313208886.