International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Automation of Android Application Testing PT. Bareksa Portal Investasi Using Appium Framework

Johan[1], Rosny Gonydjaja[2]
[1]Department of Technology and Engineering, Software Information System, University of Gunadarma, Indonesia
[2]Department of Technology and Engineering, University of Gunadarma, Indonesia

Abstract**—** *PT. Bareksa Portal Investasi is a company that has a business run in financial technology. Products are offered to the customer are mutual funds such as money market, mixed funds, stock, and fixed income. The technologies are used in Bareksa as a platform for transaction are website, Android, and IOS application. One of most popular media that is used for customer doing their transaction is Android application. Major problems happen while developing new features are duration and lack accuracy of result from the application testing toward the standard when the testing run manually. Considering at these situations, this article writes about the implementation of building automatic testing of Bareksa Android applications with framework Appium. The results of the test implementation can prove the percentage of the success of the test, test failure, and the time efficiency of the whole test. These data will be printed in the form of a test report. The printing of the report indicates that this test automation tool was successful in checking the application feasibility.*

Keywords**—** *Testing, Automation, Appium, Android*.

I. INTRODUCTION

PT. Bareksa Portal Investasi (Bareksa) is a company that has a core of business in financial technology service as a portal of various mutual fund products. Mutual fund products have been existed in Bareksa grouped in 4 main products such as money market, mixed, shares, and fixed income mutual fund [1].

Application based on Android, which is owned by Bareksa, has an extreme effect of increasing customer amount. The recent total of the customer in Bareksa already more than 200.000. The biggest percentage accesses it is using Android. According to the data on July 2018 from government institution named Otoritas Jasa Keuangan, whole the mutual fund customer has a 822.000 total [2]. It means Bareksa has minimally 20% market share of mutual fund.

As a feedback responses from company toward total, interesting, and transaction activity of mutual funds which in happening of high growth, Bareksa does many development and update to financial services based on digital peripheral. Development of digital tools are specialized to technology information division which has 16 members in it such as backend and frontend developer. Android application development of Bareksa for making up performance, user interface, user experience, new features are supporting transaction of mutual fund products.

Android application of Bareksa in development processes uses some steps to do such as planning, designing, coding, testing, implementing, and documenting. Testing is a step for verification and validating of application toward planning documentation. It is important factor to increate comfortability and security of application for customer to access.

The obstacles had been happened in period of testing are testing of whole component from user activity that connected to mutual fund transaction in Bareksa. The interface component is crucial thing to be noticed because it as a representation of customer behavior to access Bareka application [2]. The number of components are quite large and it was tested manually by personal needs take time too much. Tester should record in writing to the reporting form all the anomaly that is happened to the application. Application will be rechecked after new update of bugs or error fixing till all the trouble in the application be avowed clear. Anomaly is a condition of unnormal situation that caused error and defect to the application. Using a manual testing will tend to human error such as undetected anomaly happened [3].

II. LITERATURE REVIEW

A. Related Works

These are some research about manual and automation testing which had written as a journal is shown in Table 1.

TABLE 1. Related Works

Year	Researcher	Research Content
2012	G. Shah, P. Shah and R. Muchhala	As a result of this research is a basic of using Appium software. Some of functionality from it is explained well and given the sample of using it.
2013	N. Ahmad, M.W. Boota and A.H. Masoom	The method used in this study is to take measurements using the principle of testing smartphone usage power to users. The parameters used include the appearance and efficiency of the device when used by the user. The object tested is an Android smartphone and IOS.
2013	R. Narang	Explain the basic principles of software engineering in full with digital book formats. Information on the testing method is explained about the whitebox and blackbox testing methods in detail.

B. Testing

Testing is a one of another phase in life cycle development system. Testing was worked by internal team of development after finishing coding process. Some of type of testing are [4]:

1. Functional testing.
2. Non-Functional testing.
3. Security testing.

Testing is a procedure that established by various aspect in a life. It should be done because the production application has

Johan and Rosny Gonydjaja, "Automation of Android Application Testing PT. Bareksa Portal Investasi Using Appium Framework," *International Research Journal of Advanced Engineering and Science*, Volume 4, Issue 2, pp. 252-257, 2019.

a high quality and a strong guarantee to the users. Controlling an application quality must use a standard measurement [4]. These are list below steps for keeping quality control:

1. Defining some defect for grouping it into three main categories such as critical, medium, low.
2. Rechecking whole of project documentation.
3. Retesting all written code manually.
4. Retesting the application using standard scheme. It is unit testing, system testing, and integration testing.
5. Making a note for defect component and decide which is the most priority for fixing.
6. Using a standard score for deciding the application passed the test or not.

The result of testing phase is a data of note from *bugs* in the application that can be disturb the system. All of those notices are reported into report form. *Bugs* are able to categorize is shown Table 2 [4]:

TABLE 2. Bugs Category

Category Type	Description
Critical	This can cause a fatal error to the application. The user cannot access the application well.
Moderates	Some errors are causing error in particular component and it is not too fatal for application operational.
Simple Bugs	This bug can produce warning to the application but operational is working well.

C. *Blackbox Method*

Blackbox Method has a definition the testing, which has based on documentation from all determined requirements and specification, another name of this method is functional testing [5].

Referring to this definition, a series of application testing do not require access to the source code of program because the tester only take a focus for verify and validate software compare to documentation from input to output.

Verifying activity is a test that is carried out is to review the specifications that have been formed in the physical application to see the compatibility with the initial provisions contained in the application design stage. In addition, verification activities will be supplemented by validation, where the activity is to reassure it.

D. *Appium Framework*

Appium is based on the official website documentation www.appium.io is an open source software that has a role as an application testing automation tool. This tool can be used for various types of platform objects tested such as mobile devices and websites [6].

Fig. 1. Logo of Appium

The details of testing object Appium can be classified into 3 types of testing object such as [3]:

1. Native Apps

An application that had been written using Standard Development Kit (SDK) originally launched from smartphone provider, as example native applications are Android using Java and IOS using Objective-C.

2. Mobile Web Apps

It is an application with a website platform but is given special settings to be able to run well in appearance and functionality in various personal digital devices such as smartphones and tablets. The features present in this type will not be different from websites that are opened through search engines on laptops or personal computers (PCs).

3. Hybrid Apps

An application that is made a combination of native apps and website techniques. The codes that are formed based on HTML5 and Javascript, then the code is translated into an interpretation of the original code of each operating system in the mobile device. Examples of Hybrid technologies include PhoneGap and Ionic Framework.

According to the main documentation of the official Appium website, there are 4 philosophies that underlie Appium's presence as an automated testing tool solution. Those are 4 philosophies of Appium [6]:

1. Appium does not use the concept of recompiling applications in run of automation testing processes.
2. Supporting multi language programming to write test code scenario.
3. Simple mobile automation because API (Application Programming Interface) has been provided for automation.
4. Appium is an open source Framework.

In addition to these philosophies, several advantages and disadvantages of Appium are as follows:

Benefits:

1. Appium can run in various PC Operating System such as Linux, MacOS, and Windows.
2. Appium can be downloaded and installed without paying its original license.
3. Appium uses multi language of programming support: Python, Java, Javascript, and Ruby.
4. Appium is able to test website and mobile application object.
5. The final report of Appium testing can be converted to website html form and pdf extension.

Based on direct observation and literature review on the Appium application below are some of Appium's weaknesses including:

1. It is not supporting for Android Operating System under 4.1 version.
2. Appium documentation is still only a few versions. It does not support multiple display behavior in IOS devices.

E. *TestNG Report*

TestNG is a library used in the Java language to generate a display automated test results in the Appium software test. Referencing to the site www.toolsqa.com (accessed April 3, 2018), the advantages of test can be described as follows [7]:

1. Built with annotations that are easier to use and understand.
2. The test method can depend on other methods.
3. Test cases can be run separately by grouping.
4. Create test reports in the form of HTML or XML.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Fig. 2. Example of Testing Log

III. PROPOSED METHOD

The research method applied for the implementation of an automated testing system with Appium has 5 stages including: (1) problem identification; (2) system analysis and needs; (3) configuration of the Appium test application; (4) making testing programs; and (5) application testing. Flowchart depiction can be seen in Figure 3.

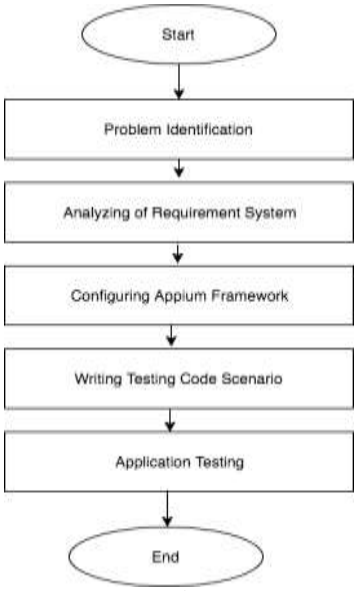


Fig. 3. Flowchart of Research Method

Based on the depiction of the flow diagram in Figure 3. the initial stages carried out were about problem identification. In the background section of this research, we have explained some of the vertices of the problem, namely the problem of limited personnel, time, and the speed of adding features is the main problem identified for the application testing section. Tests run by manual testing use the Blackbox Method. The correlation between these problems becomes an initial identification of a solution to be carried out by making an automated testing system.

After exploring the core of the problem in stage (1), the next step is to analyze the system and needs. Looking back at the constraints that are the background of this research, the system requirements that need to be fulfilled are as below:

1. The application can test all components of the input, process, and output in the Bareksa Android application
2. Showing test reports in an informative form such as test success, percentage, and length of testing time.

In terms of user analysis, Bareksa customers and prospective customers can use the Bareksa application. The difference is only in the limited access to transactional activities in the prospective customer mode. Customers or prospective customers can simulate Non-transactional features.

In terms of technical analysis, this automated testing implementation was designed with the following specifications:

1. Intel Core I5 3.1 GHz Macbook Pro 2016.
2. 8 Gb 2133 MHZ Random Access Memory (RAM) LPDDR3.
3. Solid State Disk (SSD) 512 Gb.

In step (3) is an initialization process in starting to design the automatic testing of Bareksa Android applications with framework Appium. Appium will be able to run if the entire configuration of the Appium library is installed according to the conditions so that it can compile Java program code which will become the testing process flow. The stages of installing Appium framework can be described as follows [6]:

1. Download Appium master installment https://github.com/appium/appium-desktop/releases/tag/v1.6.2. (accessed 10 August 2018).
2. Format the software that is downloaded has the extension .dmg
3. Double-click on the .dmg extension then navigate to Application Folder
4. After successfully installed, there will be an Appium icon in the list of installed applications.

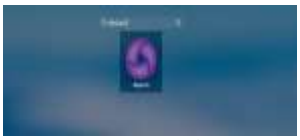


Fig. 4. Appium Successfully Installed

Fig. 5. Bareksa Android Application

The process in point (3) if it has been done, then the next step of the research is to write the code for the Appium test program. The program code written in this study is using the Java language. Code writing algorithms are written based on basic reference or operational standard application procedures for each page and its components. The reference standard will be the basis of testing to automatically examine each component of input and output. In Table 3, the operational standards for the Bareksa application will be written.

Except to the operational standards of each page in Table 3, the flow of testing carried out in the automatic testing of Bareksa Android applications is found in Figure 5.

Johan and Rosny Gonydjaja, "Automation of Android Application Testing PT. Bareksa Portal Investasi Using Appium Framework," *International Research Journal of Advanced Engineering and Science*, Volume 4, Issue 2, pp. 252-257, 2019.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

TABLE 3. Procedural Standard for Bareksa Application

	Page Name	Operational Standard	Description
1	Registration	• Form must be filled • E-mail, phone number, and all input data must be valid.	Form registration as a new customer
2	Login	• E-mail must be valid and already registered successfully • Password must be match to user data password in the system	Authentication page for customer
3	News	• The contents are text, image and link for reading only	Publishing new article of mutual fund objects.
4	Products	• Contains a sequence list of mutual fund products in alphabetical order • Expanding the details of each product • Taking action to but mutual fund products	Showing all list of mutual fund product in Bareksa
5	Simulation	• Content includes entering nominal initial investment data • Monthly investment entry • The calculation results will contain a graph of investment simulation and the amount of the calculation cost	Displaying a simulation page for users trying mutual fund investment simulations
6	Transactions	• Content in the form of nominal prices for mutual funds • There is a detailed	Displays the order of transactions owned by the costumer
7	Profile	• Content the user's personal data with text display only • Facilities for changing profiles and will be re-checking by Bareksa	Displaying complete profile of customer.
8	Portfolio	• The form of total user investment • List of mutual fund products that have been owned by user	Displaying the nominal number of presentations, total savings, and the total of customer mutual fund portfolios.

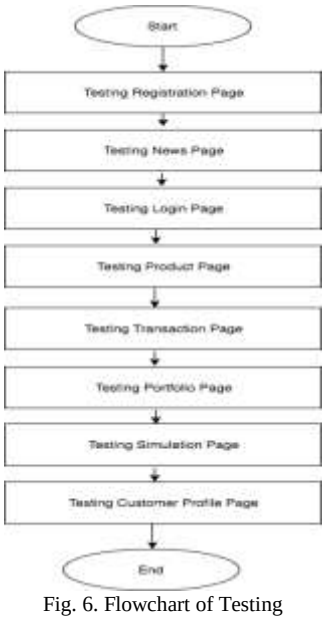


Fig. 6. Flowchart of Testing

Figure 6 illustrates the sequence of the Appium process in checking applications on each page whether it is in accordance with the test scenario built based on Table 4.

TABLE 4. Application Testing Scenario

	Feature	Testing Scenario
1	Login	• Check the e-mail form and password whether it successfully appears properly then it is declared passed the test • If the e-mail entered is not in accordance with the default e-mail format, error mitigation will appear. If mitigation is successfully displayed it will be declared passed the test
2	Registration	• Checking all registration form whether successfully performing well then declared to pass the test • Displaying the results of error mitigation on each form that is not suitable, if it is seen then the application is declared to have passed the test
3	Simulation	• Checking the existence of data and display for product, type of mutual funds, and form enter the investment value. • If the input does not match, then an error mitigation appears that can be detected by the test equipment and then passed the test • If the initial investment and initial input on simulation form and simulation summary are the same the it is declared passed the test, if not then the opposite
5	Transaction	• If the display is not available, the application does not pass the test • If the nominal enter sell, buy and exchange produces an error message, the test tool detects that there is still an error in the application • If the nominal enter, sell, buy and exchange does not produce an error, the inspection will proceed to summary of the transaction, whether it is equal to the nominal traded, if it is the same the pass the test
6	Change Password	• If the display is not available, the application does not pass the test • If the error mitigation is seen in the form, the test tool can detect the error and the application is passed the test • If the application goes to the message successfully and the test tool detects a successful message that appears then the application runs according to the function
7	News and Analysis	• If the displays is available, the test tool detects that the Boolean value is true, if it does not appear the opposite. If the Boolean status is true then the application passes the test
8	FAQ	• If the displays is available, the test tool detects that the Boolean value is true, if it does not appear the opposite. If the Boolean status is true then the application passes the test
9	Customer Portfolio	• If the displays is available, the test tool detects that the Boolean value is true, if it does not appear the opposite. If the Boolean status is true then the application passes the test
10	Product Detail	• If the displays is available, the test tool detects that the Boolean value is true, if it does not appear the opposite. If the Boolean status is true then the application passes the test • If data on the product page is not the same as the product details and the testing tool detects the Boolean status is false then the application does not pass the test

The description of Table 4 and Table 5 writing the program is done using the help of the Eclipse code editor. Use the program code editor to do the application-testing program. Writing Java program code that will perform the automated

Johan and Rosny Gonydjaja, "Automation of Android Application Testing PT. Bareksa Portal Investasi Using Appium Framework," *International Research Journal of Advanced Engineering and Science*, Volume 4, Issue 2, pp. 252-257, 2019.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

testing process has several method annotations and their functions as follows:

1. Before Method

This method segment is useful for initializing media or devices used for testing in the form of real emulators or Android devices [6].

2. Test

This annotation will run various method names written to interrupt each component that is examined and compared to the standard operating procedures and test scenarios.

3. The process of testing automation of the application can be carried out after stage (4) is carried out. Appium testing is carried out with these steps:
    1. Make sure the Appium server is running and running properly.
    2. Compile Java programs written by pressing the run icon in the Eclipse editor code.
4. Wait for the process of compiling the Java Appium program.
5. Appium server will initialize each configuration of the research media by interrupting the media written in the program code of the Before Method section.
6. The interruption process will run the test according to the scenario description can automatically be seen running on the test media.
7. If the application has finished checking all the test reports will automatically form in the test folder's file folder, the report-output folder [7].

Fig. 7. File Location report

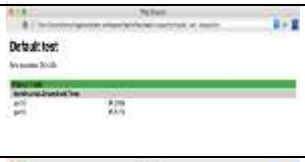
IV. Result and Discussion

The submitting author is responsible for obtaining agreement of all coauthors and any consent required from sponsors before submitting a paper. It is the obligation of the authors to cite relevant prior work. Authors of rejected papers may revise and resubmit them to the journal again.

The test results on the Android Bareksa application are the final information received by the examiner to determine whether the application can be upgraded to the production stage, which is uploaded to the Android application store page, Google Play store.

Tests are carried out on 5 devices where each device is given a scenario for the application situation to pass the test and does not pass the test. Appium's ability to detect any incompatible components can be seen from this test report. For situations that meet, the green indicator will appear for applications that pass the test and are red for those situations that failed the test [3].

TABLE 5. Report of Test Result

Media	Testing Scenario	Screenshot
Emulator Android P1	Success	
Emulator Android P2	Success	
Emulator Android P3	Success	
Emulator Android P4	Success	
Emulator Android P5	Success	
Emulator Android P1	Failed	
Emulator Android P2	Failed	
Emulator Android P3	Failed	
Emulator Android P4	Failed	
Emulator Android P5	Failed	

The test results in Table 5 are the results of testing the application that displays screenshots of the results of Appium's test in mode showing the method part that was carried out when the conditions were successful and failed. Besides being

Johan and Rosny Gonydjaja, "Automation of Android Application Testing PT. Bareksa Portal Investasi Using Appium Framework," *International Research Journal of Advanced Engineering and Science*, Volume 4, Issue 2, pp. 252-257, 2019.

able to display the time display mode, the test report can display the percentage of the number of methods tested.

The formula for determining method presentations that have a successful phase or failed phase is as below:

$$\frac{success\ methods}{all\ method} x\ 100\ \% \quad \text{(a)}$$

Test experiments conducted by Framework Appium on Bareksa applications have a test method numbering 2. If the test succeeds all methods, show a green indicator or success, the percentage of the calculation formula (a) will produce a total of 100%, otherwise if one has an indicator it fails will display a percentage of 50%.

Analyzing more deeply to see if the results of this test succeed in answering about the length of the test time is to describe the data on the screenshot of Table 6 as follows:

TABLE 6. Testing Data Time

Emulator	Testing Scenario	Estimated Time Method A (second)	Estimated Time Method B (second)	Estimated Time (second)
P1	Success	47.994	46.731	94.725
P2	Success	39.801	46.442	86.243
P3	Success	34.083	39.527	73.61
P4	Success	34.102	42.295	76.397
P5	Success	34.102	42.295	76.397
P1	Fail	42.806	35.105	77.911
P2	Fail	42.806	35.105	77.911
P3	Fail	42.806	35.105	77.911
P4	Fail	42.071	49.215	91.286
P5	Fail	53.856	40.649	94.505
Average of Testing Time				82.6896

Explanation:

- P1 to P5 is an Android emulator device that runs the test implementation.
- The type of experiment consists of the type of test with conditions of success and failure.
- The test time for methods A and B is the modular method of the tested component group.
- The duration of the test is the total time of testing performed on each device along with the average test time of the overall testing of the Bareksa Android application.

The data generated in the testing experiment in Table 6 is the data into consideration whether the application can be upgraded to a production application. The production application is an application that is confirmed to be uploaded to the playstore to be used by Bareksa customers. Percentage worth 100% with a successful indicator is a determination to determine the increase in the application to the play store.

V. CONCLUSION AND RECOMENDATION

A. Conclusion

Based on the implementation carried out in this study on the testing of Bareksa Android applications using framework Appium, a number of things can be concluded as follows:

1. Using the Appium application, can provide solutions to the problems of application testing faced by PT. Bareksa Investment Portal related to testing routines with limited human resources, test time, and speed of adding features.
2. The application of this test has a direct impact on the time of the Bareksa application test in its entirety which based on the data in Table 6 - the average test time in either successful or failed conditions has a test time value of 82.6896 seconds.
3. This automated test answers the problem at the speed of adding features because by using this automation the renewal of the test algorithm can be adjusted directly in the programming language used in this Appium framework.

B. Recommendation

Suggestions that can materialize to develop this research and writing are:

1. For PT. Bareksa Investment Portal, is expected to be able to implement test automation in addition to the Android application, it can also implement automation on other website, server and service applications within the company.
2. For researchers or readers of this journal, the results of this study can be used as a reference to review and develop the application of Appium in conducting testing automation. The development of this feature can also be applied in a blend between a single application test process and the automation of the test results directly into the production stage for the next study.

REFERENCES

[1] Bareksa, "Bareksa," *PT. Bareksa Portal Investasi*, Available: www.bareksa.com. 2013.
[2] P. Raut and S. Tomar, "Android Mobile Automation Framework," *International Journal of Engineering and Computer Science*, vol.3, issue 10, pp. 8555-8560, 2014.
[3] G. Shah, P. Shah and R. Muchhala, "Software Testing Automation using Appium," *International Journal of Current Engineering and Technology*, vol. 4, issue 5, pp. 3528-3531, 2014.
[4] R. Narang, "Software Engineering Principales and Practices," New Delhi: McGrewHill Education, 2015.
[5] S. Nidhira and J, Dondeti, "Black Box and White Box Testing Technique –A Literature Review," *International Journal of Embedded Systems and Applications*, vol. 2, issue 2, pp. 29-50, 2012.
[6] Appium, "Appium," The JS Foundation, Available: www.appium.io, 2012.
[7] L. Sharma, "ToolsQA," ToolsQA, Available: www.toolsqa.com, 2014.