Comparison of Automatic and Manual Regression Testing on Mobile Application Health Technology with Black Box Testing Method

Y F Putri[1], A B P Irianto*[2], S Sharma[3]

[1,2]Department of Informatics, Universitas Atma Jaya Yogyakarta, Indonesia

[3]Department Department of Electronics and Computer Engineering, Institute of Engineering Pulchowk Campus, Tribhuvan University, Kathmandu, Nepal

E-mail: yoannafransisca9912@gmail.com[1], bagas.pradipta@uajy.ac.id[2], suman.sharma@ioe.edu.np[3]

Abstract. The opening of opportunities for health tech services in Indonesia can lead to increased competition. This makes the company continue to innovate by updating or adding functionality to software systems that have been developed. Because of this, software testing with a shorter time is required so that test performance remains maximum. The object of this research is the Teman diabetes application developed by PT. Global Urban Essentials. In this research, the software testing conducted includes automatic regression testing and manual regression testing. The two testing approaches will be compared to evaluate their effectiveness in detecting bugs and their efficiency in terms of time usage. The method used in automatic regression testing is the black box testing method, implemented within the Software Testing Life Cycle (STLC) that is include requirement analysis, test planning, test case development, test environment setup, and test execution. The primary focus of this research is to asses the effectiveness of automatic and manual regression testing in identifying bugs/errors and measuring efficiency in time utilization, based on the analysis of test reports generated from both methods. In this study, it was found that testing carried out by automation is 9.77% faster. In addition, a bug was found in the shopping feature as a result of the new features being released. This bug was discovered after running automated regression testing and was not found in manual regression testing because the bug testing was missed.
Keywords: application Testing; black Box; comparison; regression testing, software development.

1. Introduction

Health tech or technology-based health services began to develop rapidly, especially in the Asia Pacific region. This is evidenced by a survey from *Gallen Growth Asia* [1][2]. The survey reveals that health tech has developed from 2017 and peaked in 2018 with an investment value of $ 3.3 billion[1] and stated that Indonesia is one of the countries with the potential for health tech to work together with the existing health industry. Opportunities for health tech can arise due to the distribution of health services that are not optimal, besides the fact that the size of the population in Indonesia is quite large it also makes health tech services more potential to develop [1].

The opening of opportunities for health tech services can lead to increasing competition so companies must continue to innovate by updating functionality in software systems that have been developed [3] [5]. The existence of updates or additions to this functionality makes testing a crucial role[5] in providing assurance of software quality and presenting key studies of specifications, design,

and coding [6]. If application testing is not carried out, a bug will arise that causes errors in hardware or software so that it cannot run as expected and can result in financial losses [6].

Gojek suffered huge losses of up to hundreds of billions caused by the Google Play Card bug. This bug allows users to get a Google Play Card Balance with a nominal value ranging from 500 thousand to 1.2 billion by paying only 22 thousand [7]. The National Institute of Standards and Technology (NIST) 2002 conducted a study that a bug in a software application causes an economic loss in the United States of $ 59.5 billion annually [8]. Financial losses such as in both examples can be minimized by applying manual testing and automatic testing [5]. Both methods of testing are needed in order to provide optimal results in using the test time and finding bugs [9].

One type of testing that can be done is regression testing, which is used to check the integration between updates or additional functionality with old features that already exist in the application [5]. However, manual regression testing has a weakness, like the requires a long period of time in the testing process [10]. This is stated in the results of the research "Comparing the effort and effectiveness of automated and manual tests", where manual testing takes 13 minutes longer than automatic testing to execute 233 test cases [11]. The relatively long period of time in the testing process can cause human error in manual testing which results in ineffective bug discovery [12]. Therefore, another test is needed, namely regression testing which is carried out by automatic. In addition, automatic regression testing is performed because it can be used repeatedly, thus saving time [5]. The object of this research is the Teman diabetes mobile application developed by PT. Global Urban Essentials is engaged in health tech

2. Literature Review

Software testing is a critical phase in the software development life cycle (SDLC), ensuring that applications function as intended. Among various testing approaches, regression testing plays a crucial role in verifying that recent changes or updates do not negatively impact the existing functionalities of an application. Regression testing can be conducted manually or through automation, with each method presenting unique advantages and limitations. This literature review explores the comparison between automated and manual regression testing using the black-box testing method, highlighting their effectiveness, efficiency, and practical applications.

2.1. Regression Testing in Software Development

Regression testing is performed after modifications, such as bug fixes, enhancements, or system upgrades, to ensure that existing features remain unaffected. According to Myers et al. [13] , regression testing is essential for maintaining software reliability, particularly in applications requiring frequent updates. Manual regression testing involves human testers executing predefined test cases, while automated regression testing employs specialized tools to execute test scripts automatically.

2.2. Black-Box Testing in Regression Testing

Black-box testing is a software testing technique where the internal workings of an application are not known to the tester. Instead, the tester focuses on inputs and expected outputs to verify functionality [14]. This method is particularly useful for functional and regression testing, ensuring that software behavior aligns with requirements. Regression testing, when conducted using the black-box approach, emphasizes external correctness rather than internal code structure [15].

2.3. Comparison of Automated and Manual Regression Testing

2.3.1. Efficiency and Execution Speed

One of the primary advantages of automated regression testing is its speed and efficiency. Research by Sharma & Jain (2019) [16] indicates that automated regression testing is significantly faster than manual testing, allowing multiple test cases to be executed in a short period. Manual testing, on the other hand, is time-consuming and prone to human errors, particularly when testing large-scale applications.

2.3.2. Accuracy and Reliability

Manual regression testing depends on human intervention, which can introduce inconsistencies and errors [17]. Automated testing eliminates this issue by executing test scripts consistently. A study by

Bertolino & Polini (2013) found that automated regression testing improves accuracy by reducing false positives and negatives in test results [18].

2.3.3. Cost Implications and Scalability

While automated testing has a higher initial cost due to tool procurement and script development, it becomes cost-effective in the long run . Manual testing, although requiring no automation setup, incurs higher costs over time due to labor-intensive execution and repeated test cycles.

For complex applications requiring extensive regression testing, automation is more scalable. Automated tests can run concurrently across multiple devices and environments, a feature that manual testing lacks. Black-box testing frameworks, such as Selenium, Katalon Studio, and Appium, enhance scalability by supporting various platforms [19].

2.3.4. Flexibility and Exploratory Testing

Despite its advantages, automated testing lacks the adaptability and intuitive judgment of human testers. Manual testing is more flexible for exploratory testing, where testers dynamically interact with the application to discover unexpected issues [20]. Black-box testing in a manual setting allows testers to assess usability and user experience, which automation tools cannot effectively evaluate.

3. Research Methodology

3.1. Research methods

The method used in designing automatic regression testing is the black box testing method using the Software Life Cycle Testing System (STLC). STLC is used to make manual and automatic regression testing more structured, besides that STLC is part of the black box testing method [21]. The stages in STLC start from requirement analysis, test planning, test case development, test environment setup, test execution, and test cycle closure [22].

In this study, the STLC stage was carried out only up to the test execution stage as in Figure 1, because the focus in this study was on the results of the automatic regression testing that had been carried out and would be compared with the results of manual regression testing so that the effectiveness of the testing in finding bugs / errors could be seen.

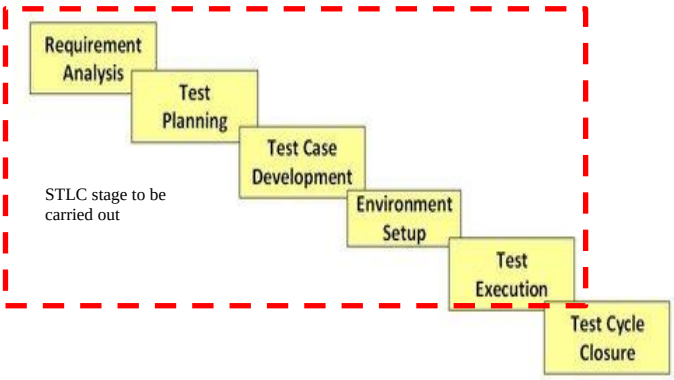

Figure 1. STLC stage to be carried out

3.2. Research Stage

Problem identification serves as the initial stage of this study on PT. Global Urban Essentials, focusing primarily on software testing to ensure the quality and reliability of the Teman Diabetes application, as illustrated in Figure 2. After that, it will enter the STLC phase which starts from analyzing the needs and functional flow of the Teman diabetes being tested. This analysis uses a use case diagram. Use case diagrams are used because they describe the interaction relationship between users and the system. In addition, it is used to find out what functions are contained in the software system [23]. After the analysis is complete, a test scenario will be made in accordance with the needs and functional flows that have been previously analyzed. The next step is to do manual regression testing before automatic is carried out, after which it is followed by the preparation of manual regression testing reports.

Then proceed with the selection of tools that will be used for automatic regression testing. The tools that will be used are Katalon Studio. Katalon Studio was chosen because Katalon Studio is simpler and easier to use QA. If it has been selected, the tools will be setup so that they can be used.

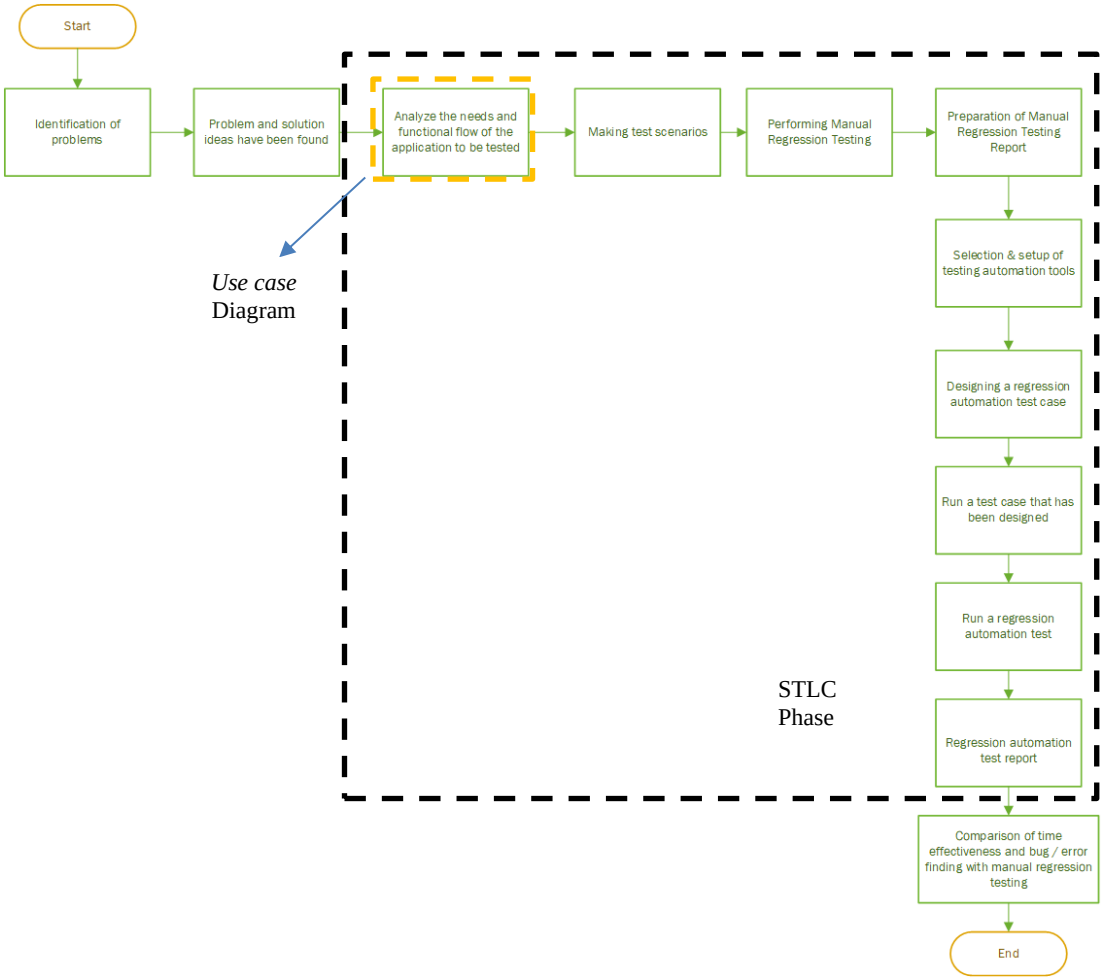

Figure 2. Research Stages

The next flow is the design of an automatic test case for automatic regression testing to be performed. The test case is designed according to the previously created test scenario. After the test case is designed, it will be run first before being used for automatic regression testing, this is done to check the suitability of the designed test case. If the test case is appropriate, automatic regression testing will start using the Katalon Studio tools. In Katalon Studio, a test report will be generated automatically and can be seen by the tester on the link https://analytics.katalon.com/ which has been integrated before. From the results of the automatic regression testing report, it will be possible to see the time effectiveness and the discovery of bugs or errors from the application and will be compared with manual regression testing.

4. Results and Discussion

4.1. Result

In this result, we will discuss the process and results of automatic and manual regression testing in the Teman Diabetes application that has been done. Testing on the Teman Diabetes application tests a new feature, namely the menu kelola feature. This Setting Menu feature has a function to connect the teman diabetes Application and the Doctor To Doctor (D2D) Application, where diabetes patients who use the teman diabetes application can share health data in real time with medical professionals who handle it, where medical personnel can access diabetes patient health data through the application.

4.1.1. Requirements Analysis

The QA (Quality Assurance) team will analyze the software requirements that have been obtained from the Product Manager, System Analyst, and others. This analysis aims to find out the details of the software, modules, features and functions that will be built, reviewed, and validated if there are

deficiencies and / or completes the clarity of the requirements that have been given. The requirement we found from the use case diagram showed in figure 3 and figure 4.

4.1.1.1 Use Case Diagram Level 0 Teman diabetes Application

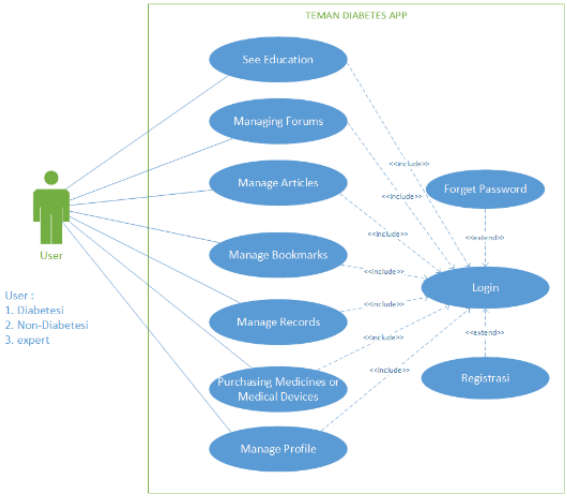

Figure 3. Use case Diagram level-0 Teman Diabetes Application

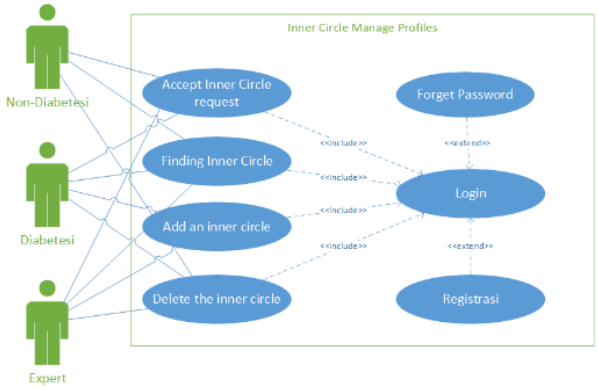

Figure 4. Use case diagram level-1 Teman Diabetes Application

Figure 3 above are the functions of the teman diabetes application for which automatic regression testing will be carried out. Teman diabetes application users are divided into three consisting of diabetics, non-diabetics, and experts. Before being able to use the functions contained in the Teman Diabetes Application, these users are required to register by entering their name, email, and password. After the email and password are registered, users can log in, if the user forgets the registered password, they can input the registered email in the forget password feature.

When the user is logged in, the system will enter the homepage. On this homepage there is an educational feature that provides information about diabetes, this information is presented in five categories, namely the latest content containing educational class information and educational videos provided by doctors, then what is the category of diabetes which contains an introduction to diabetes in the form of videos or reviews. The third category is diabetes complications, which include complications from diabetes. Then the fourth is the diabetes management category which contains tips for healthy living for diabetics. Then finally there is a category of myths and facts that contains answers to questions about diabetes. In addition to educational features, there is a forum feature that can be used by users to share questions or experiences about diabetes. In addition, in this forum feature users can provide feedback from forum reviews that have been made by other users. On the homepage there is also an article feature that contains review information and tips. These reviews can be shared and tagged according to user wishes. Then there is a bookmark feature that contains the tagging results of the forum and article features made by the user,

In the Teman Diabetes there is also a recording feature that is used by users to manage blood sugar and health records. In managing blood sugar, users can add blood sugar through the DNurse medical device or input it manually. The addition of this blood sugar record can only be input once a day for each type of blood sugar examination. Besides being able to add blood sugar, users can also view blood sugar history and charts. In managing health records, users can add and edit exercise data, then weight data, Hb1Ac, blood pressure, and medication reminders. Users can also share blood sugar data with family, friends, or medical experts.

Also in this application, users can purchase drugs or medical devices related to diabetes in the shopping feature. This diabetes companion application collaborates with GoApotik, which is one of the

business fields of PT. Global Urban Essentials which has collaborated with almost all pharmacies in Indonesia and is engaged in the sale of drugs and medical devices [24].

Mean while, users can manage profiles. In managing profiles, users can change profiles, then view forum reviews that have been made, view user activity, users can also manage records through managing this profile, then there are features that invite friends, choice of forum topics, FAQ, about the Teman diabetes, privacy policy , and log out.

There is an inner circle feature which is like a friendship feature in managing this profile, so that users can add friends or family, accept inner circle requests, or delete inner circles. However, the type of user who can add an inner circle is only diabetics and experts, non-diabetic users cannot add an inner circle but can only accept and delete. For more details, below shown in Figure 4, with the name the Level-1 use case diagram of the inner circle feature. The detailed features that will be carried out by automatic regression testing include: Onboarding, Registration, Login, Forget Password, Education, Forum, Article, Bookmark, Blood sugar records, Health records, Medication reminder, Shares blood sugar data, Shopping, Personal account data, Manage forums in profile feature, Inner circle, Adding records via profile, About the teman diabetes application, Logout, Notification.

4.1.1.2 Manual Regression Testing

Table 1. Report Manual Regression Testing

Things to do	Time	Number of Test Cases	Passed	No.
Download and install the APK	15 minutes	89	89	0
The test was carried out on 89 test cases	53 minutes			

Table 1 shows the report time needed when generated after manual regression testing, where the total duration of time required is 1 hour 8 minutes with a time division of 15 minutes used to download and install APK and 53 minutes used for testing. The number of test cases tested in this manual regression testing is 89 test cases with the number passed 89, so there are no bugs in this test.

4.1.2. Test Planning

Before performing automatic testing using Katalon Studio, there are several steps that must be taken, namely:

1. *Install Node.js*

   Node.js is software used to develop web-based applications using the JavaScript programming language. Because JavaScript runs on the client or browser side only, therefore Node.js was created to complement the role of JavaScript so that it can act as a programming language that can run on the server side as well. In addition, Node.js can run on the Windows Operating System, Mac OS X and Linux without updating the program code.

2. *Setup Appium*

   Appium is needed as an automatic tool for native (native), mobile web, or hybrid applications for IOS and Android platforms.

3. *Setup Devices*

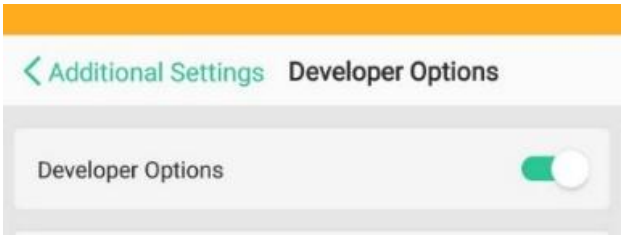

Figure 6. Developer Mode in Mobile Settings

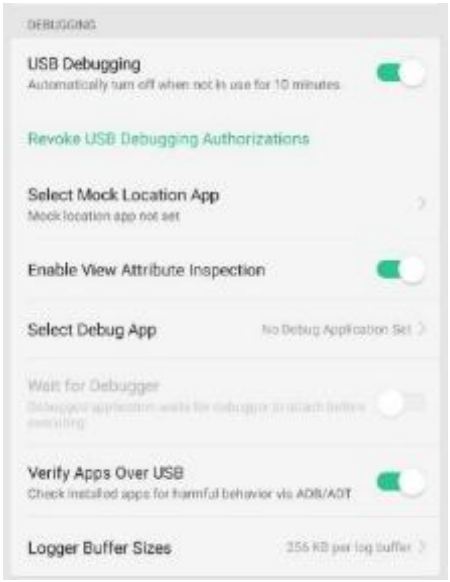

Figure 7. USB Debugging Mode

Things that need to be set in the mobile android device are:

- Turn on developer mode as in Figure 6, then in Figure 7 turn on USB debugging so that the smartphone is in debug mode by allowing permission to be granted and simulating input via USB debugging, Install via USB so that the smartphone can install applications via USB.
- *Install* Android SDK

Android SDK (Software Development Kit) is a kit that can be used by developers to develop Android-based applications. It includes several tools such as a debugger, software library, emulator, documentation, sample code and tutorials.

- Connect your Android phone to your laptop or computer using a USB cable.

4.1.3. Test Case Development

In the third phase of this STLC, the QA team makes reference in testing, creates test cases, test data, and automatic test scripts based on the test cases that have been made.

4.1.3.1. Registration Test Case

Figure 8. Test case for user registration with Valid Data

```
19 import com.github.javafaker.Faker as Faker
20
21 Faker faker = new Faker()
22
23 String name = faker.name().fullName()
24
25 String firstName = faker.name().firstName()
26
27 String email = firstName + '@yopmail.com'
28
29 println(firstName)
```

Figure 9. Faker email script code

In Figure 8 is a test case design to test the registration function with the condition "user registers with valid data", this condition is included in the normal / positive test case type. The Expected Result of the test case in Figure 8 is a register that is carried out by a successful user and displays an education page, and the system stores new user data and the user has a user name and password to log in as a user with diabetes. In this test case, the email faker is used, namely yopmail, as arbitrary data used for testing data. The e-mail faker used for test data generation in automated testing for allow tester to generated random, and realistic-looking emial. The email faker useful for preventing data duplication, enhancing test automation, avoiding spam or real email usage and testing registrations and login workflow. In this study the task done by writing a code script on Katalon Studio as shown in Figure 9.

4.1.4. Environment Setup

This phase, the QA Team ensures the environment test. Environment test is performed to determine software requirements.

No.	ID	Description	Run
1	Test Cases/TD-1 Registrasi/TD 1.0 Registrasi - User registrasi input field password tanpa angka		
2	Test Cases/TD-1 Registrasi/TD 1.1. Register - User registrasi input field nama blank		
3	Test Cases/TD-1 Registrasi/TD 1.2 Registrasi - User klik Sudah Punya Akun dan user klik Buat Akun		
4	Test Cases/TD-1 Registrasi/TD 1.3 Registrasi - User klik button back		
5	Test Cases/TD-1 Registrasi/TD 1.4 Registrasi - User klik button simpan		
6	Test Cases/TD-1 Registrasi/TD 1.5 Registrasi -User registrasi input field email blank		
7	Test Cases/TD-1 Registrasi/TD 1.6 Registrasi - User registrasi input field email selain email		
8	Test Cases/TD-1 Registrasi/TD 1.7 Registrasi - User registrasi input field email yang sudah terdaftar		
9	Test Cases/TD-1 Registrasi/TD 1.8 Registrasi - User registrasi input field nama kurang dari 3 character		
10	Test Cases/TD-1 Registrasi/TD 1.9 Registrasi - User registrasi input field password kurang dari 6 karakter		
11	Test Cases/TD-1 Registrasi/TD 1.10 Registrasi - User registrasi input field password tanpa huruf kapital		
12	Test Cases/TD-1 Registrasi/TD 1.11 Registrasi - User registrasi input field password blank		
13	Test Cases/TD-1 Registrasi/TD 1.12 Registrasi - User registrasi input field nama special character		
14	Test Cases/TD-1 Registrasi/TD -1.13 Register - User registrasi dengan valid data		

Figure 10. Registration test suite

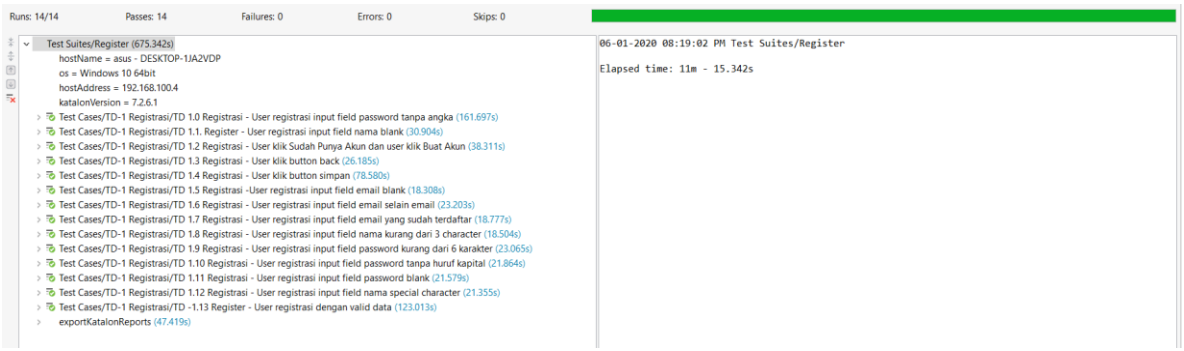

Figure 11. Test suite result registration

4.1.4.1. Registration Test Suite

In Figure 10. the registration feature test suite design that functions to run the registration test cases that have been previously made. The number of registration test cases that will be run is 14 test cases. Whereas in Figure 11. are the results obtained after running the registration test suite, namely the number of 14 test cases passed without any bugs with a total duration of 11 minutes 15 seconds.

4.1.5. Test Execution

No.	ID	Run with
1	Test Suites/Register	Android
2	Test Suites/Login	Android
3	Test Suites/Forget Password	Android
4	Test Suites/Main Menu - Home	Android
5	Test Suites/Event	Android
6	Test Suites/Main Menu - Rekaman	Android
7	Test Suites/Main Menu - Belanja	Android
8	Test Suites/Main Menu - Profile	Android
9	Test Suites/Notifikasi	Android

Figure 12. Test suite collection for Automatic Regression Testing

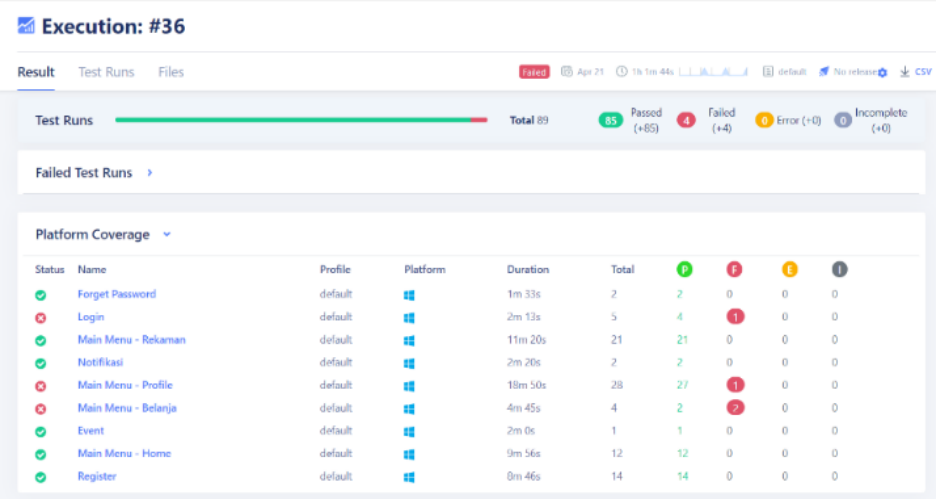

Figure 13. report of automatic regression testing results

Figure 12. is a test suite collection that contains test suite registers, logins, homepages, forget passwords, shopping, records, events, notifications, and profiles that have been previously created and run. Automatic regression testing is performed by running the test suite collection in Figure 12.

The report provided by Katalon Studio tools after automatic regression testing for the menu kelola feature using APK is in Figure 13. Reports will be generated automatically by Katalon Studio as in Figure 13. In this report the number of test cases run is 89 test cases with 2 forget password test suites with a test duration of 1 minute 33 seconds, the Login test suite is 5 test cases with a test duration of 2 minutes 13 seconds with a duration of time, the test suite Main Menu-Recording as many as 21 test cases with a test duration of 11 minutes 20 seconds, the Notification test suite as many as 2 test cases with a test duration of 2 minutes 20 seconds, the Main Menu-Profile test suite as many as 28 test cases with a testing duration of 18 minutes 50 seconds,1 hour 1 minute 44 seconds. The number of passed in automatic regression testing is 85 test cases with the number of failed 4 test cases that occur in the login test suite, Main Menu-Profile, and Main Menu-Shopping.

4.2. Discussion

Before doing automatic regression testing on this Teman diabetes, manual regression testing was performed. Manual regression testing requires a total testing time of 1 hour 8 minutes, manual regression testing includes downloading the APK provided via Google Drive by the developer, then installing an APK that has new features on a smartphone, then testing all old features. For more details, it can be seen in table 2. below.

Table 2. Time Range in Manual Regression Testing

Activity	Time
Download and install the APK	15 minutes
The test was carried out on 89 test cases	53 minutes

The total number of test cases on the menu kelola feature is 89, besides that this test can only be done after the feature development stage has been completed by the developer team, and after functional testing is carried out on new features. In manual regression testing, no bugs were found, for more details as in table 3.

Table 3. Results of Manual Regression Testing

Number of Test Cases	Passed	No.
89	89	0

However, after automatic regression testing was carried out with a duration of 1 hour 1 minute 44 seconds, the results were that there was a bug in the old feature, namely shopping, this means that the

bug that occurs is a side effect given from the new feature to the old feature. This bug occurs when all "Order Now" buttons cannot be tapped.

This impact can occur because there is a custom library method used by developers for the educational video feature which is the same as tapping "order now" in the shopping feature. This feature is still under development by the development team, because in developing the menu kelola features and educational videos it is carried out in parallel by the developer team, but the menu kelola feature will be released first. Meanwhile, another bug occurred in the login feature where the user logged in manually with a blank email field, then the forum feature with a tap reply forum question occurred due to an unstable signal, because Katalon Studio's tools really depend on the stability of the connection from smartphones and laptops.

Table 4. Comparison of the duration of manual regression testing with automatic regression testing

Testing type	Number of Test Cases	Test Case Execution Duration	*Bug* that was found
Manual Regression Testing	89	1 hour 8 minutes 26 seconds	0
Automatic Regression Testing	89	1 hour 1 minute 44 seconds	4

In table 4 it can be seen that manual regression testing to test 89 test cases takes 1 hour 8 minutes 26 seconds and there are no bugs found. Meanwhile, automatic regression testing takes 1 hour 1 minute 44 minutes to execute 89 test cases with 4 bugs found. From the above discussion, it can be concluded that testing by means of automated in the use of test execution time is faster than manual testing which takes a long time to execute test cases. This time ratio can be seen from the percentage increase in speed [25], with equation (1):

$$Time\ Ratio = \frac{Automated\ Testing\ Time}{Manual\ Testting\ Time} \quad (1)$$

$$Speed\ Increase\ (\%) = (1 - \frac{Time\ for\ Automated\ Testing}{Time\ for\ Manual\ Testing}) \times 100\% \quad (2)$$

Based on equation (1) that has been done, the time ratio we have is 0.9023 this means automated testing takes 90.23% of the time required for manual testing. In the speed increase case equation (2), we have the result is 9.77%. So the result is that automatic regression testing is 9.77%. more faster and efficient in testing time compared to manual regression testing. In addition, automatic regression testing is more effective in finding bugs that are the impact of new features on old features.

There are obstacles in automatic regression testing using Katalon Studio tools, namely the Appium server which sometimes experiences errors which can cause the testing process to be hampered so that a force close occurs on the application, and signal stability affects the testing process using Katalon Studio, because if the signal device used is not stable, so when a tool is executing a command it can be too late, such as when a tool performs a verify element command in the toast message, the smartphone device has not yet appeared an element, even though the tools have moved to the next step. This can happen because the toast message element on the smartphone device is still loading while the tools are running.

5. Conclusions and Suggestions

5.1. Conclusion

In this study, it can be concluded that to improve the performance of the regression testing time on the Teman diabetes, you can use testing by means of automatic, because the automatic test is 9.77 % faster than manual regression testing. This research also found a bug in the shopping feature which is the impact of new features that will be released. This bug was discovered after running automatic regression testing.

So it can be concluded that automatic regression testing is efficient in using testing time and is effective in finding bugs because when manual regression testing is carried out, the testers missed the bugs which are the impact of new features on old features. However, the problem with automatic regression testing is that the Katalon Studio tools used depend on the stability of the signal, resulting in an error when the test is run. While automation testing provide a efficiency improvement, the time recuction is not drastic but automatic testing can reduce the labor and operational cost.

This study recommends implementing automated regression testing for the Teman Diabetes application. Manual testing, while useful, has inherent limitations, including the risk of human error, which may lead to undetected bugs. By transitioning to automated regression testing, the process can be more reliable, consistent, and efficient, ensuring that defects are thoroughly identified and addressed. This approach enhances the application's quality and stability, ultimately improving user experience and confidence in the system. Therefore, incorporating automation in regression testing is essential to maintaining the integrity and usability of the Teman Diabetes application as it continues to develop.

5.2. Suggestion

Based on the conclusions of this study, the suggestions given in relation to further research are:

1. Automated regression testing should be conducted using multiple tools to facilitate a comparative analysis of their advantages and disadvantages relative to Katalon Studio. This will provide deeper insights into the effectiveness and efficiency of different automation tools.
2. This study focused solely on the Android operating system; therefore, future research should extend the analysis to the iOS platform to ensure broader applicability and cross-platform consistency in testing.
3. Future research should develop a structured, automated regression testing process flow, adhering to industry standards such as ISO/IEC 29119, which provides standardized testing policies and strategies guidelines. Implementing such standards will enhance the automated testing process's reliability, repeatability, and overall effectiveness.

6. References

[1] Yenny Yusra, “Layanan Healthtech di Asia Berkembang Pesat, di Indonesia Belum Signifikan,” Dailysocial.id. [Online]. Available: https://dailysocial.id/post/healthtech-di-asia

[2] Andriansyah Agustian, “Menerka Potensi Perkembangan Startup di Bidang Kesehatan,” Dailysocial.id. [Online]. Available: https://dailysocial.id/post/menerka-potensi-perkembangan-startup-di-bidang-kesehatan

[3] R. A. Wijaya, N. Ilhama, and B. Paramastri, “Pentingnya Pengelolaan Inovasi Dalam Era Persaingan,” vol. 5, no. 2, pp. 217–227, 2019.

[4] Mohamad Nur Asikin, “ACIS 2018: Pentingnya Inovasi Hadapi Ketatnya Persaingan Merebut Pasar,” JawaPos.com. [Online]. Available: https://www.jawapos.com/ekonomi/bisnis/16/11/2018/acis-2018-pentingnya-inovasi-hadapi-ketatnya-persaingan-merebut-pasar/

[5] S. Torniainen, “Improving Effectiveness of Regression Testing of Telecommunications System Software,” 2008.

[6] A. Ngah, M. Munro, and M. Abdallah, “An overview of regression testing,” Journal of Telecommunication, Electronic and Computer Engineering, vol. 9, no. 3-5 Special Issue, pp. 45–49, 2017, doi: 10.1145/308769.308790.

[7] A. Permana, “Gojek Rugi Puluhan Milyar Gara-gara Bug GPC,” Trentech.id. [Online]. Available: https://www.trentech.id/gojek-rugi-puluhan-milyar-gara-gara-bug-gpc/

[8] R. Cohane, “Financial Cost Of Software Bug,” Medium.com. [Online]. Available: https://medium.com/@ryancohane/financial-cost-of-software-bugs-51b4d193f107

[9] K. Akromunnisa, "Manual and Automated Testing," Medium.com. [Online]. Available: https://medium.com/@kitamiakr/manual-and-automated-testing-908d184bf0be
[10] K. Suryaprabha and P. Sudha, "Measuring Effectiveness of software Quality by Comparing Manual Testing and Selenium," no. February, 2019.
[11] I. Dobles, A. Martinez, and C. Quesada-Lopez, "Comparing the effort and effectiveness of automated and manual tests: An industrial case study," Iberian Conference on Information Systems and Technologies, CISTI, vol. 2019-June, no. June, pp. 19–22, 2019, doi: 10.23919/CISTI.2019.8760848.
[12] F. Zahro, F. Pradana, and A. Arwan, "Kakas Bantu untuk Penentuan Prioritas Test Scenario Berdasarkan UML Activity Diagram," vol. 3, no. 6, pp. 5376–5382, 2019.
[13] G. J. Myers, T. Badgett, and C. Sandler, Eds., The Art of Software Testing. Wiley, 2012. doi: 10.1002/9781119202486.
[14] B. Beizer, Black-Box Testing: Techniques for Functional Testing of Software and Systems. Wiley, 1995.
[15] C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing: A Context-Driven Approach. Wiley, 2001.
[16] A. Sharma and S. Jain, "Efficiency Analysis of Automated Versus Manual Regression Testing in Agile Development," Journal of Software Engineering Research and Development, vol. 7, no. 1, pp. 45–62, 2019.
[17] L. Mariani, M. Pezze, and O. Riganelli, "Automating Regression Testing for Evolving Software," IEEE Transactions on Software Engineering, vol. 43, no. 4, pp. 299–319, 2017.
[18] A. Bertolino and A. Polini, "Software Testing Automation: Advances and Challenges," Software Quality Journal, vol. 21, no. 4, pp. 627–648, 2013.
[19] S. W. G. AbuSalim, R. Ibrahim, and J. A. Wahab, "Comparative Analysis of Software Testing Techniques for Mobile Applications," J Phys Conf Ser, vol. 1793, no. 1, p. 012036, Feb. 2021, doi: 10.1088/1742-6596/1793/1/012036.
[20] J. A. Whittaker, Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design. Addison-Wesley Professional, 2009.
[21] Krishna, "black box testing," guru99.com. [Online]. Available: https://www.guru99.com/black-box-testing.html
[22] "Software Testing Life Cycle," guru99.com. [Online]. Available: https://www.guru99.com/software-testing-life-cycle.html
[23] M. K. Hutauruk, "UML Diagram : Use Case Diagram," Binus University. [Online]. Available: https://socs.binus.ac.id/2019/11/26/uml-diagram-use-case-diagram/
[24] GueSehat, "Goapotik," goapotik.com. [Online]. Available: https://www.goapotik.com/pages/tentang-kami
[25] Yoel, "Cara Menghitung Persentase Kenaikan," Statmat. [Online]. Available: https://statmat.id/cara-menghitung-persentase-kenaikan