An Implementation of Automatic Dart Code Verification for Mobile Application Programming Learning Assistance System Using Flutter

Yan Watequlis Syaifudin
Dept. of Information Technology
State Polytechnic of Malang
Malang, Indonesia
qulis@polinema.ac.id

Agus Salim Hadjrianto
Dept. of Information Technology
State Polytechnic of Malang
Malang, Indonesia
gusalim04@gmail.com

Nobuo Funabiki
Dept. of Elec. and Comm. Engineering
Okayama University
Okayama, Japan
funabiki@okayama-u.ac.jp

Dewi Yanti Liliana
Dept. of Informatics Engineering
State Polytechnic of Jakarta
Jakarta, Indonesia
dewiyanti.liliana@tik.pnj.ac.id

Andi Baso Kaswar
Dept. of Computer Engineering
State University of Makassar
Makassar, Indonesia
a.baso.kaswar@unm.ac.id

Usman Nurhasan
Dept. of Information Technology
State Polytechnic of Malang
Malang, Indonesia
usmannurhasan@polinema.ac.id

***Abstract*—The popularity of smartphone devices has rapidly increased in recent years and many people utilize smartphones for various needs. The development of mobile applications has been aimed at various fields that make the demand for mobile application programmers increase. Recently, *Flutter* has become a software development kit for cross-platform applications development, including *Android* and *iOS*, so many software developers have adopted it. To provide a self-learning system for studying mobile programming with *Flutter*, we propose a learning assistance system with an automatic Dart code verification feature. Based on our previous studies in *Android Programming Learning Assistance System (APLAS)*, the automatic code verification process can adopt software testing process for Android applications. The learning model provides learning materials for studying and practicing by solving an assignment. A learning topic of developing a simple application is prepared for the proposed system evaluation. 40 university students in IT department have been appointed to study *Flutter* and solve the assignment. Finally, they can solve the assignment correctly and give positive opinions about using this system.**

***Index Terms*—self-learning system, automatic code verification, Flutter, Dart, Android, iOS**

I. Introduction

The popularity of smartphone devices around the world has rapidly increased in recent years. Many people utilize smartphones for various needs, so the development of smartphone-based applications or mobile applications has been aimed at various fields, ranging from personal tools, office, education, health, hobbies, and lifestyle [1]. Currently, the operating systems for smartphones are dominated by *Android* by Google and iOS by Apple. Based on Statista, by two thousand nineteen, they have been installed on more than "ninety-nine%" of smartphones worldwide [2]. Due to this fact, the demand for mobile application programmers in the job market has increased significantly in recent years, especially on Android and iOS-based application development projects [3]. Most companies released mobile applications in both versions of the operating systems to connect with their customers.

Previously, to provide a mobile application for the two operating systems, programmers had to develop two applications with different programming languages, namely *Java/Kotlin* for Android and Swift for iOS [4]. In two thousand seventeen, Google has released an open-source UI software development kit for cross-platform applications development, including Android and iOS, namely *Flutter* [5]. By using *Dart* programming language, programmer can build mobile applications for both operating systems using a single Flutter project [6]. So that many developers use it for mobile application development. Also, professional schools and universities have adopted it for mobile programming courses.

To provide students' self-learning in mobile programming using Flutter, we propose a model to automate the *Dart code verification* that adopts *test-driven development method (TDD)* [7]. Based on our previous studies in *Android Programming Learning Assistance System (APLAS)* [8], the automatic verification process can be implemented by adopting software testing process for Android applications that consists of unit testing and integration testing. For automated testing, Flutter provides three levels of testing including unit testing, widget testing, and integration testing. A *unit testing* is used to test a single function, method, or class. A *widget testing* (in other UI frameworks referred to as component test) is used to test a single widget of UI. An *integration testing* is used to test a complete application or a large part of an application.

The implementation of this learning system considers applying the learning process based on a real Flutter programming environment that commonly uses an *IDE (Integrated Development Environment)* for Flutter, such as *VS Code* and *Android*

Studio. For practicing the concept, the student has to solve an assignment by writing source code on the IDE and the provided test code(s) that can be used to verify the written source code and give feedback to the student if some fails or errors occur. A learning topic to study a simple application development has been prepared for implementation that requires developing an application by utilizing *TextSpan*, *TextRich*, *TextStyle*, and *ButtonText*. To finish the whole application step-by-step, the learning materials are divided into tasks to make learning process easier.

To evaluate the implemented system, we asked forty students in an IT department in Indonesia to study the simple appli- cation topic, including solving the assignment. The results show that all students are successfully completing the given four tasks and creating a simple Android application. The students' feedback also deliver positive comments and notice that the system is helpful for studying Flutter. It confirms the effectiveness of a learning system with automatic Dart code verification for learning mobile application development usingFlutter.

The organization of this paper consists of: Section II lists some studies relating to this paper, Section III presents the concept of automated testing in Flutter, Section IV explains the learning system implementation with automatic code verification, Section V evaluates the implemented system, lastly Section VI presents a comprehensive conclusion with future works.

II. Related Works

This section presents some research and works in the literature related to this study.

In "two thousand fifteen", Kang et al. [9] presented a study in mobile programming education with a case study for Android ap- plications. They adopt Multi Android Development Tools on MIT App Inventor and Eclipse for programming IDE. Byusing puzzle models, novice students feel easy to simulate an Android application. However, this model requires manual checking to confirm source code correctness.

In "two thousand nineteen", Arif et. al. [10] analyzed the challenge results of software testing for mobile applications using some testing tools. The testing types include unit, integration, regression, and system testing.

In "two thousand nineteen", Arb et. al. [11] develop a mobile application that can be run on both iOS and Android devices using the Flutter framework. The superiority and efficiency of Flutter framework were confirmed by applying a simple procedure.

In "twenty twenty one", Syaifudin et. at. [12] proposed a self-learning system for developing Android applications, especially for Ad- vanced Widgets topic. It presents an automatic code validationusing *JUnit* and *Robolectric*.

III. Automated Testing in Flutter

This section explains the overview of automated testing model in Flutter and the three levels of automated testing.

A. Overview of Flutter

Since the first release of Android application, *Android Studio* became the main IDE for Android application development by combining *Java*, *Kotlin*, and *XML* as programming languages. In addition, Apple has also released *Apple XCode* as a platform for iOS application development. However, in "two thousand seventeen", Google released an open-source UI software de- velopment kit for cross-platform applications development, including Android and iOS [13]. By using *Dart* programming language, programmers can build mobile applications for both operating systems using a single Flutter project [14]. With Flutter, both Android and iOS applications can be developed using some adopted IDEs, such as *VS Code*, *Android Studio*, and *IntelliJ IDE* [15]. So that many developers use it formobile application development.

In Flutter, automated testing is used to help ensure that the applications work correctly before publishing, while maintaining features and speed at fixing bugs [16]. As shown in Figure 1, Flutter provides three levels of automated testing, including unit testing, widget testing, and integration testing [17].

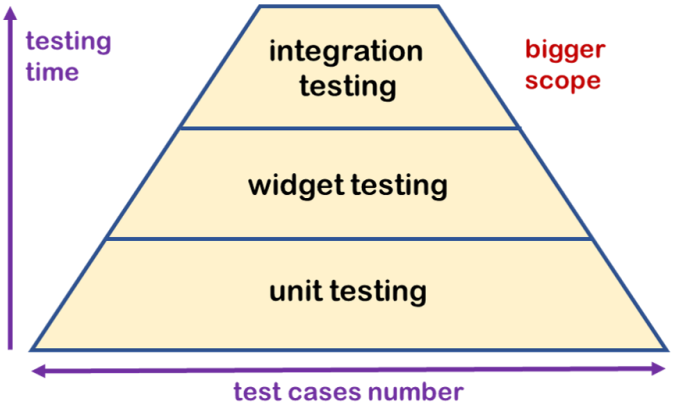

Fig. 1. Three levels of automated testing in Flutter.

B. Unit Testing

Unit testing is a type of testing that is used to verify the behavior of a single function, method, or class. It can confirm the behavior of a unit of logic under a variety of conditions. In Flutter framework, the test package contains *the core framework* for writing unit testing, and *the flutter_test package* for additional utilities of widget testing. Because of its simplicity, unit testing is not intended to test the behaviors or functions for disk reading or writing, rendering to screen, and receiving user actions from other applications or services. Also, it performs a faster testing process compared to others.

C. Widget Testing

Widget testing is a level of testing that in other UI frameworks is referred to as component testing. It is useful for testing components that exist in Flutter, such as in Mockito or JUnit, the testing widget can be said to be similar to the Mock function, in Flutter it is called Pump, where the tester

will visualize the components to be tested. The goal of this testing is to confirm that the widget components and UI screen are looked at and can interact as expected.

To test a widget, the process applies numerous classes and requires a test environment that supplies a suitable widget lifecycle context. For example, the widget testing verifies a widget that should be able to receive and respond to some actions or events from the user. It also confirms the widget's precise placement on the screen. A widget testing should be more complex and comprehensive than unit testing, but this process doesn't require running the application on a real device or emulator.

D. Integration Testing

The top level of testing, integration testing, provides the biggest scope of testing that simulates the application to a real device or emulator. It tests a complete application or a large portion of an application. The goal is to confirm that the application with all the widgets and services being tested working together and running properly as expected. Moreover, integration testing can be used to verify the application's performance.

IV. System Implementation

This section explains the implementation of the proposed system, including software specifications, learning materials, and learning process by students.

A. System Environment

The implementation of this learning system considers applying the learning process based on a real Flutter programming environment that commonly uses an IDE (Integrated Development Environment) for Flutter [18], such as *VS Code* and *Android Studio*. As shown in Figure 2, the system provides a guide document for the student to learn Flutter. The guide document is a PDF format document that contains concepts of a specific subject on Flutter and step-by-step guidance for practicing writing source code based on the concept. For practicing the concept, the student has to solve an assignment by writing source code on an IDE for Flutter, as a copy of the real environment of Flutter programming. The system also provides test code(s) that can be executed to verify the written source code and give feedback to the student if some fails or errors occur.

B. Learning Materials

Referring to the Flutter books, in the first step, students learn how to use Dart language to develop a simple application that shows plain text, formatted text, and images. To implement a learning process with automatic Dart code verification, a learning topic for developing a simple application is prepared. The learning topic provides an assignment to develop an application by utilizing *TextSpan*, *TextRich*, *TextStyle*, and *ButtonText*. The expected result after a student solving the assignment is shown in Figure 3. To finish the whole application,

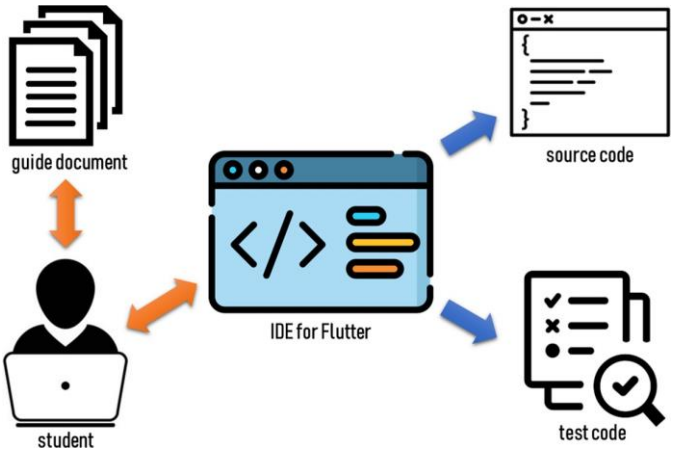

Fig. 2. System environment for learning Flutter.

the guide document provides four tasks for learning step-by-step. Each task guides the student to define the specific widget and give styling to it.

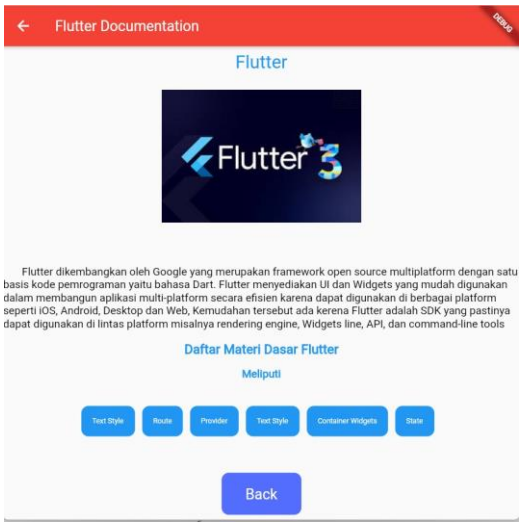

Fig. 3. The interface of a simple application for assignment.

C. Learning Process by Student

To apply the learning process with automatic code verification [19], the student has to follow a procedure as shown in Figure 1.

1) Student preparation: student prepares a unit of PC or laptop that already installed a PDF reader software and an IDE for Flutter.

2) Receiving learning materials: student receives learning materials from the teacher that contain guide document (PDF format) and testing code(s).

3) Studying: student starts to learn Flutter from the guide document for a specific topic. In this step, the student studies the concepts of a specific subject in Flutter.

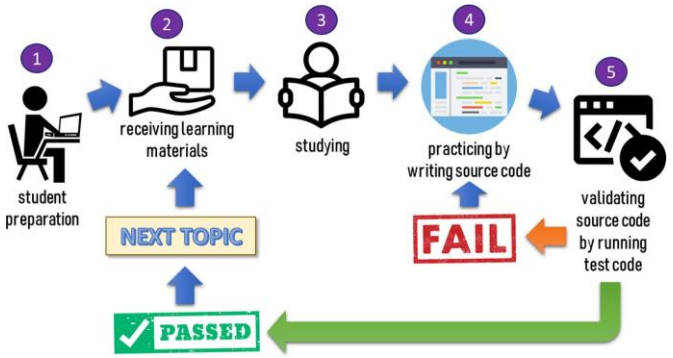

Fig. 4. Procedure of learning process by student.

4) Practicing: student writes source code on the IDE by following instructions in the guide document. In this step, the student practices the concepts by developing an application.

5) Validating source codes: student verifies the written source codes by executing the given test code(s). In this step, the student will get a result consisting of *Passed* or *Failed*. If the result is passed, he/she can continue studying the next topic. If the result is failed, he/she gets messages and suggestions to fix the source codes, then he/she can verify again the source codes.

V. Evaluation

This section explains the evaluation results of the implemented system.

A. Evaluation Setup

For a comprehensive evaluation, 40 students of Android programming class have been appointed to study the simple application topic. They are students in IT department in a university in Indonesia. The learning materials are prepared with an assignment to create a simple application that must be solved in a day. Android Studio with some required plugins is adopted as IDE for developing the Flutter application. To expedite the evaluation process, students are guided in setting up the required software until they are ready to be used to run Flutter applications. They use their own PCs and also the laboratory's PCs that meet minimum hardware requirements. To measure the learning effectiveness by using automated assistance, the solving successfulness and students' feedback will be collected.

B. Students' Solving Activities

Table I summarizes the results on assignment solving activities by "forty" students. It reveals that all students can solve the given assignments with support from the automated code verification feature. The solving time of each student varies depending on his/her experience in Dart coding, ability in programming, and the number of source code verification processes. The first task relatively needs a longer time dueto it requires some programming environment configuration and students have to adapt to Dart programming and Flutter framework. Then, students need less time to solve the subsequent tasks except for the last task due to it needs to define many buttons on the UI.

TABLE I
Solving activity results on simple application assignment

task no.	**1**	**2**	**3**	**4**
# of passed	40	40	40	40
# of failed	0	0	0	0
average time (minutes)	22	twelve	ten	fifteen
shortest time (minutes)	fourteen	10	7	eleven
longest time (minutes)	38	32	30	32

C. Disscussion on Student's Feedback

The results of students' feedback can be divided into three categories, including:

- **positive** opinion ("eighty seven point five"%) contains the system is helpful, easy to create a mobile application with Flutter, and the guidance is clear;
- **neutral** opinion ("five"%) contains no comment, start pro- gramming, and need mind concentration;
- **negative** opinion ("seven point five"%) contains many configurations, PC problem, and error compiling.

All feedback are dominantly positive opinion that confirm the learning system's effectiveness. The feature of automatic code validation is very useful for students to achieve the correct results in programming practice. Only a maximum of three students noticed negative feedback related to some kind of difficulties in using Flutter framework. They can be solved by giving minimum hardware requirements, step-by-step detail configuration, and a troubleshooting model. The average time to solve in task as shown in Table I is relatively fast, even the students do not have experience in Dart programming. It reveals the increase of their self motivation and being familiar with this system.

VI. Conclusion

This paper presented the implementation of automatic Dart code verification for mobile application learning assistance system using Flutter. This system is important to support students for self-learning mobile application development. Currently, Flutter promises the ability to build native applications on both iOS and Android platforms and can achieve native performance [20]. The results show that all students are successful in solving the given assignment. All students' feedback are dominated by positive comments and notices that the system is helpful to studying Flutter. It confirms the effectiveness of a learning system with automatic Dart code verification for learning mobile application development using Flutter. Future works will focus on the development of subsequent learning topics, web-based learning platform, and evaluation in blended learning.

REFERENCES

[1] S. Hsiao, "Reviewing the Global Smartphone Industry Strategic Implication in Response to COVID-19 Situation," Int. J. Interactive Mob. Tech., vol. 15, no. 14, pp. 169-192, 2021.

[2] Statista GmbH. 2021. "Mobile operating systems' market share worldwide," available at: *https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/* (accessed on 12 June 2022).

[3] K. McCullen, "An Android application development class, " J. Comput. Sci. Coll., vol. 31, no. 6, pp.11–17, 2016.

[4] R. N. Sansour, N. Kafri, and M. N. Sabha, "A survey on mobile multimedia application development frameworks," 2014 Int. Conf. Multimedia Comput. Systems (ICMCS), 2014, pp. 967-972.

[5] Google. 2022. "Build apps for any screen," available at: *https://flutter.dev/* (accessed on 12 June 2022).

[6] S. Y. Ameen and D. Y. Mohammed. "Developing cross-platform library using flutter," European J. Eng. & Tech. Research, vol. 7, no. 2, pp.18–21, 2022.

[7] V. Farcic and A. Garcia, "Test-driven Java development," Packt Pub., 2015.

[8] Y. W. Syaifudin, N. Funabiki, M. Kuribayashi, and W.-C. Kao, "A proposal of Android programming learning assistant system with implementation of basic application learning," Int. J. Web Inform. Sys., vol. 16, no. 1, pp. 115-135, 2019.

[9] H. Kang and J. Cho, "Case study on efficient Android programming education using multi Android development tools," Indian J. Sci. Tech., vol. 8, no. 19, pp. 1-5, 2015.

[10] K. S. Arif and U. Ali, "Mobile application testing tools and their challenges: A comparative study," Proc. in 2nd Int. Conf. Comput. Math. Eng. Tech., pp. 1-6, Sukkur, Pakistan, 2019.

[11] G. I. Arb and K. Al-Majdi, "A freights status management system based on Dart and Flutter programming language," Proc. in Int. Conf. Modern Appl. Infor. Comm. Tech., Baghdad, Iraq, 2019.

[12] Y. W. Syaifudin, N. Funabiki, M. Kuribayashi, and W.-C. Kao, "A proposal of advanced widgets learning topic for interactive application in Android programming learning assistance system," SN Comput. Sci., vol. 2, no. 172, pp. 1-13, 2021.

[13] L. Heimann and O. Veliz, "Mobile Application Development in Flutter," in Proc. 53rd ACM Tech. Symp. Comp. Sci. Edu., New York, 2022.

[14] S. Boukhary and E. Colmenares, "AcClean approach to Flutter development through the Flutter clean architecture package," 2019 Int. Conf. Comput. Sci. & Comput. Intell., pp. 1115-1120, Las Vegas, 2019.

[15] J. Christoph, D. Rösch, T. Schuster, and L. Waidelich, "Current Progress in Cross-Platform Application Development - Evaluation of Frameworks for Mobile Application Development," Int. J. Adv. Softw., vol. 12, no. 1, pp. 30-45, 2019.

[16] A. Hussain, H.A. Razak, and E.O. Mkpojiogu, "The perceived usability of automated testing tools for mobile applications," J. Eng. Sci. and Tech., Special Issue on ISSC 2016, pp. 86-93, 2017.

[17] Google. 2022. "Testing Flutter apps," available at: *https://docs.flutter.dev/testing* (accessed on 12 June 2022).

[18] F. W. Zammetti, "Practical Flutter : Improve your Mobile Development with Google's Latest Open-Source SDK," Apress, California, 2019.

[19] Y. W. Syaifudin, S. Rohani, N. Funabiki and P. Y. Saputra, "Blending Android Programming Learning Assistance System into Online Android Programming Course,"in Proc. 9th Int. Conf. Infor. Edu. Tech., pp. 26-33, 2021.

[20] N. Kuzmin, K. Ignatiev, D. Grafov, "Experience of Developing a Mobile Application Using Flutter," Infor. Sci. Appl., vol 621, pp. 571–575, 2020.