An Implementation of Solution Progress Monitoring Function in Android Programming Learning Assistance System

Abdul Rahman Patta*, Nobuo Funabiki, Minoru Kuribayashi, and Yan Watequlis Syaifudin

***Abstract*—Nowadays, *Android* is the most popular operating system for mobile devices. To enhance the education of mobile application developments, *Android Programming Learning Assistance System (APLAS)* has been proposed to help students for its self-learning. However, in the current implementation, the teacher can only check the final answers of students for APLAS assignments at the end of the course. To save at-risk students, it is better to check the progress of solving them using *Android Studio* on their PCs. This paper presents an implementation of the *solution progress monitoring function* in APLAS. It periodically collects log files of *Android Studio* and shows the progress at the interface. For evaluations, we asked 32 students taking the Android programming course in the Department of Computer Engineering, Makassar State University, Indonesia, to solve nine tasks in APLAS with this function. Since using this function, the results showed a reduction in the number of dropped-out students. The teacher could identify and assist students who are experiencing difficulties at the early stage of the course.**

Index Terms*—APLAS, Android programming, solution progress monitoring, *Android Studio

I. Introduction

Android is the most popular operating system among mobile platforms throughout the globe. It is used in hundreds of millions of devices in more than 190 countries. The share is 71.59% for the mobile market as of April 2022. This trend is growing bigger every day [1]. With this trend, the demand for Android application programmers is among the highest in the IT fields [2]. Therefore, education of mobile application developments to novice students has become essential in universities and professional schools. A significant number of educational institutes have provided Android programming classes. The computer science curriculum by *Association for Computing Machinery (ACM)* recommends including mobile computing-related topics in the computer science curriculum [3].

To improve Android programming education and assist students in its self-study, an *Android programming learning assistance system*, named *APLAS*, has been developed as a web-based application [4, 5]. *APLAS* assumes using the *Android Studio* as *Integrated Development Environment* to implement an Android application program using *Java* and *XML*. In this programming, the logic functions are implemented by Java, and the user interfaces are by XML. *APLAS* uses the *test-driven development (TDD)* method in the learning process. The answer from a student is automatically marked by *unit testing* with the *test code*, and the result is returned to the student. Thus, it is expected to guide the student to continue the study without a teacher's help [6]. In *APLAS*, *JUnit* [7] and *Robolectric* [8] are actually used for *unit testing* of the student answer codes [9]. *JUnit* tests the logic functions, and *Robolectric* tests the user interfaces. The students can validate their answers by executing the test codes in *Android Studio* by themselves. Then, they can correct the answers based on the validation results until all the tests are passed.

However, in the current implementation of *APLAS*, the teacher can only check the final answers of the students to the assignments at the end of the course. They cannot check the students' progress in solving the assignments using *Android Studio*. To assist students experiencing difficulties early in the course, a function that can monitor the progress on *Android Studio* is required to identify such students. This early action by the teacher will improve the learning outcomes and ensure the students' accountability in Android programming.

This paper presents an implementation of the *solution progress monitoring function* in *APLAS* to solve the abovementioned issue. For this function, the program on the student's PC sends the testing log file to the *APLAS* server every time a student runs the test code to check the answer code on *Android Studio*. Then, the web interface shows the number of currently completed assignments, the time required to complete them, and the number of validation requests that have been submitted by each student. These information helps the teacher to monitor the progress of solving assignments by the student. It is expected that the teacher can recognize and support the students who have difficulties solving assignments in *APLAS* and may need help from the teacher.

For evaluations, we asked 32 students taking the Android programming course in Department of Computer Engineering, Makassar State University, Indonesia, to solve tasks in *APLAS* with this function. The results confirmed the reduction of the number of dropped-out students using this function, where the teacher could identify and assist students experiencing difficulties at the early stage of the course.

The rest of this paper is organized as follows: Section II discusses technical terms for this study. Section III introduces related works in literature. Section IV literature reviews our previous works on *APLAS*. Section V presents the implementation of *solution progress monitoring function*. Section VI discusses the evaluation. Finally, Section VII concludes this paper with future works.

Manuscript received December 19, 2022; revised February 20, 2023; accepted March 9, 2023.

A. R. Patta, N. Funabiki, and M. Kuribayashi are with the Department of Information and Communication Systems, Okayama University, Okayama, 700-8530, Japan.

Y. W. Syaifudin is with the Department of Information Technology, State Polytechnic of Malang, Malang, Indonesia.

*Correspondence: p9wj5q3w@s.okayama-u.ac.jp (A.R.P.)

II. Technical Terms

This section briefly explains the technical terms related to *APLAS* that will be discussed in this paper.

A. Android Application

Android applications are designed to operate on Android devices or emulators. Several components are key building elements based on the *Android SDK library* [10]. Each component serves as an entry point into the program for the system or the user. There are four components to the application: *Activity*, *Services*, *Broadcast receiver*, *and Content provider*.

B. Android Library

Android library has the same structure as an Android application module. It can contain everything required to create an application, such as the source code, the resource files, and the Android manifest [11]. Every day, new libraries and resources will be launched to expedite developments. The goal of a library developer is to simplify the complexity of the code and to package the codes for future reuse.

C. Unit Testing

As said, to insert images in Word, position the cursor at the insertion point and either use Insert | Picture | From File or copy the image to the Windows clipboard and then Edit | Paste Special | Picture (with "Float over text" unchecked).

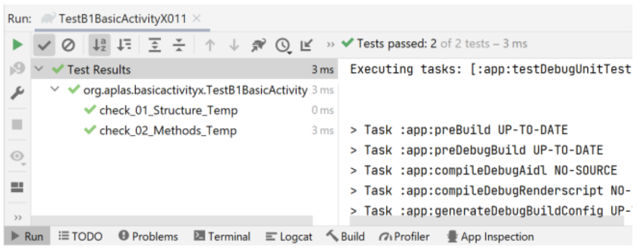
The authors of the accepted manuscripts will be given a copyright form and the form should accompany your final submission. Unit testing is an essential part of the software development lifecycle for ensuring that the source code meets the required software standards [12]. The study conducted by Linares-Vasquez *et al*. [13] revealed the development of the most successful tool in Android application developments, followed by *Robolectric*. In 2011, Sadeh and Gopalakrishnan [14] presented and demonstrated an instrumented approach to testing Android applications using *unit testing*. It makes the testing process fast by providing relevant instruments to check user interface code test coverage of GUI test cases to be written with less source code refactoring. *Junit* has been widely adopted as a Java unit testing framework [15]. An automatic testing tool is essential to implement the automated assessment or validation of source codes. Almedia *et al*. showed that *JUnit* and *Robolectric* are widely used testing tools in Android programming [16].

D. Test in Android Studio

Android Studio is designed to make tests easier by containing features to simplify the way of creating, running, and analyzing tests on a local computer. Test results can be displayed in Android Studio [17].

1) Test Result View: The test results in Android Studio appear in the *Run* window. Fig. 1 shows an example of a successful test result. Two successful tests are displayed on the left side, and the results and the messages are displayed on the right side.

2) Test Failure Analysis: The resulting window also displays the alert and the number of failed tests if they appear. Fig. 2 shows an example of a failed test result. One failed the test, and one successful test are displayed on the left side, and the messages on the failed test are displayed on the right side.



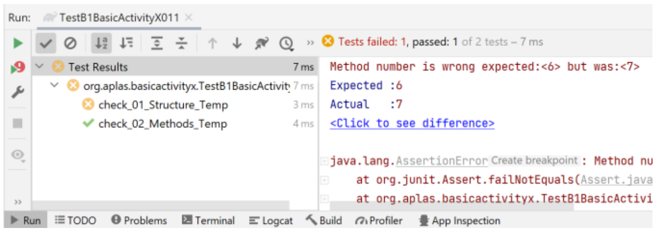
```
Run: TestB1BasicActivityX011
✓ Tests passed: 2 of 2 tests – 3 ms

✓ Test Results                                   3 ms
  ✓ org.aplas.basicactivityx.TestB1BasicActivity 3 ms
    ✓ check_01_Structure_Temp                    0 ms
    ✓ check_02_Methods_Temp                      3 ms

Executing tasks: [:app:testDebugUnitTest


> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:compileDebugAidl NO-SOURCE
> Task :app:compileDebugRenderscript NO-
> Task :app:generateDebugBuildConfig UP-
```

Fig. 1. Successful test result.



```
Run: TestB1BasicActivityX011
Tests failed: 1, passed: 1 of 2 tests – 7 ms

Test Results                                    7 ms
  org.aplas.basicactivityx.TestB1BasicActivit   7 ms
    check_01_Structure_Temp                     3 ms
    ✓ check_02_Methods_Temp                     4 ms

Method number is wrong expected:<6> but was:<7>
Expected :6
Actual   :7
<Click to see difference>

java.lang.AssertionError Create breakpoint : Method nu
    at org.junit.Assert.failNotEquals(Assert.java
    at org.aplas.basicactivityx.TestB1BasicActivi
```

Fig. 2. Failed test result.

III. Literature Review

Hanafi *et al*. [18] described their use of a mobile app to aid undergraduate students in Malaysia in their study of Android programming. They provided details on the key principles involved in creating the Android-based educational platform. The majority of students were attracted to the platform as a means of learning Android programming because it was convenient, cost-effective, and engaging.

Kose [19] introduced a web-based system that was specifically created to aid in project-based learning for courses in web design and programming. This system offers students efficient and advanced learning environments that can help them to effectively design and implement websites.

Yang *et al*. [20] developed an interactive testing system that can be utilized in computer programming classes. This system enables students to identify their programming misconceptions and enhance their programming skills by evaluating different code statements and validating them. The authors conducted an experiment using this system in a course that involved the development of Android applications.

Hayashi *et al*. [21] introduced a *collaborative learning* approach for teaching computer programming using the flipped classroom instructional strategy. To evaluate the effectiveness of this approach, the authors compared the examination results of students who were taught using the proposed method with those who were not.

Hundt *et al*. [22] introduced the *System for Automated Code Evaluation (SAUCE)* as an interactive online tool that can be used to teach parallel programming. *SAUCE* is capable of identifying common parallel algorithms that use *C++11*, *Open Multi-Processing (OpenMP)*, *Message Passing Interface (MPI)*, and *Compute Unified Device Architecture (CUDA)* threads, which can then be interactively incorporated into courses on high-performance computing or parallel computing.

Su and Wang [23] presented a web-based program that teaches the basics of programming, which includes using the Scratch tool. The aim of the program is to assist students in their online lessons and provide them with diagnostic reports based on their learning portfolios.

Rekhawi and Naser [24] designed a web-based intelligent tutoring system that can be used for teaching Android programming. This system provides lessons on the basics of Android programming, offers adaptable demonstrations, and has been shown to have positive effects on evaluators.

Amalfitano *et al*. [25] introduced a hybrid GUI exploration approach called *juGULAR*, which can be used to automatically identify gate GUIs during application explorations. This approach was developed within the modular software architecture for the Android mobile platform. The authors conducted experiments that showed an improvement in the ability to investigate covered activities and covered lines of code.

Domenach and Portides [26] put forward an online system known as the *Programming Assignment Submission System (PASS)*. This system is an interactive web-based application that allows students to submit their programming assignment answers and receive real-time feedback. *PASS* was tested in the context of C++ programming assignments.

Chaudhari and Uke [27] suggested a system for online accommodations and evaluations of Java programming assignments. The aim of this work is to investigate existing tools, design and develop intelligent web programs that can allow students to submit programming assignments and receive constant feedback.

Madeja and Poruban [28] conducted studies to analyze the errors made by students when attempting to complete Android programming projects. In order to perform the testing successfully, the students used Android testing tools such as unit testing, integration testing, and user interface testing.

Khan *et al*. [29] presented a software tool named *AUTOGRADER* that can automatically assess the correctness of programming tasks. This tool compares the answer codes to the reference implementations provided by the instructor to ensure that they follow the same logic. Instructors can utilize this tool for various purposes, such as creating test cases and administering tests.

IV. Review of Android Programming Learning Assistance System

This section reviews our previous studies [4, 5].

A. *Overview*

APLAS is an automated and self-learning system for Java-based Android programming. The source codes and packages for the Android project of the task, which are answers from students to assignments, can be validated using a test-driven development process. It is expected to repeat the process that the students access to the unit testing results for their answers and correct them if there is some error. *Junit* is adopted to test Java-based source codes as the prominent Java testing framework. *Robolectric* is adopted to generate Java objects from XML files to be tested as Java codes. The *Java Virtual Machine (JVM)* is used to power this underlying architecture.

B. *System Architecture*

In *APLAS*, a web-based online platform is adopted to distribute the materials for assignments to the students, collect the assignment answers from the students, and validate them using *JUnit* and *Robolectric* on the server. Fig. 3 illustrates the software architecture of *APLAS*. In this architecture,

1) *Learning Platform* is used by the students to complete assignments using *Android Studio* on PCs.
2) *Web Application* offers the interfaces on web-browsers to the students for downloading course materials, uploading assignment answer files, and obtaining validation results.
3) *Validator* detects new submissions of answers from students and validates them using *Gradle* links the Android project to the *Android SDK* library and the testing tools. This program runs on the background and the related data is stored in the database.

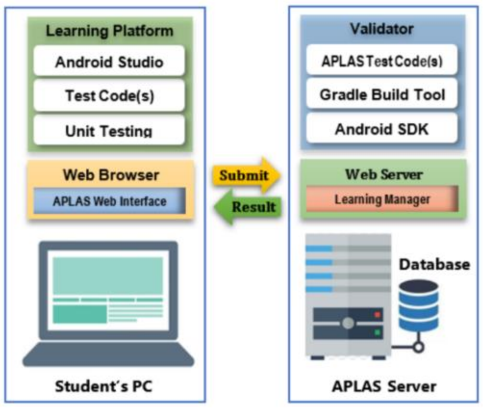


Fig. 3. Software architecture of *APLAS*.

C. *Learning Process*

The learning materials in *APLAS* consist of four stages: *user interface*, *interactive application*, *content provider*, *and service interaction*}. Each stage includes several *topics*. Then, each topic consists of several *tasks* that the students must complete in the given order. Each task contains guide documents, supplement files, and test codes. The students follow the four steps to complete each task:

1) Downloading the necessary files for each task from the server,
2) Configuring and synchronizing the project with *Gradle* [30] by integrating the necessary Android components and libraries [31],
3) Implementing and validating the source codes for the project in *Android Studio* by referring to the guidance documents and running the test codes, and
4) Submitting them to the server where the source code is automatically validated.

D. *Validation in APLAS*

One learning topic in *APLAS* contains several tasks that must be solved by the students one by one sequentially. The answer source code for each task must be validated using the provided test codes. On the client side, the unit testing using the test codes is initiated by the students in *Android Studio* on their PCs. On the server side, it is automatically performed by the validator program in the system. It is requested that the students be in the *passed* status for the current task by passing all the test cases in every test code.

Fig. 4 shows the unit testing process for an Android project, which is also carried out with the assistance of *Gradle*. When

Gradle is idling, it will execute the initialization by launching the *Gradle Wrapper runtime* in the background as a separate process to reduce the execution time. After that, *Gradle* will bring the Android project up to execute the unit testing.

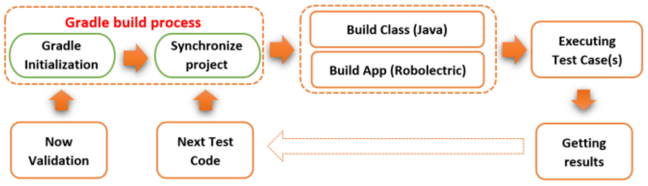


Fig. 4. Unit testing process.

It is noted that unit testing with *Robolectric* needs complicated processing. Here, *Gradle* generates Java classes by compiling the source codes. *Robolectric* interrupts this stage and creates Android application objects to be captured as *Activity* objects in the test code. Then, unit testing is applied to them [32].

V. Implementation of Progress Monitoring Function

The *progress monitoring function* receives the reports on the results of the *test-code* runs by the students from *Android Studio*. Besides, it takes the identities of the students and their computer specifications. The add-on program for realizing them is made so that the students can easily install and add it into *Android Studio*. As a result, the teacher can monitor the progress of solving the tasks using *Android Studio* by every student and can assist the students who may have difficulties in solving them.

A. Flow of Function

As shown in Fig. 3, *APLAS* consists of the *Student's PC* and the *APLAS server*. On the *Student's PC*, a student will complete the tasks using *Android Studio*, including the validations with the given *test codes*. After completing all the tasks, the student will submit them to the *APLAS server*. Then, the teacher can know the student's progress at the first time, which can be too late for poor students.

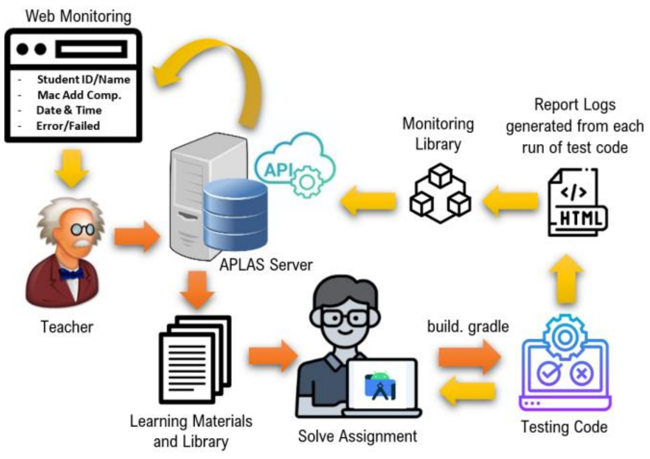


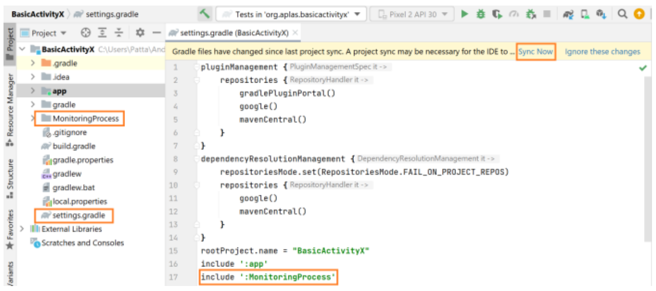
Fig. 5. Flow of process monitoring function.

To solve this problem, this paper presents the *process monitoring function*. Fig. 5 illustrates the flow of this function. For this function, the students need to download the client-side program with the learning materials from the server. *Gradle* is used here so that every time a student runs a test code, it will generate the report of the test results in the HTML file. This report will be automatically sent to the server. The server-side program will receive the reports and store them in the database. The web interface is also implemented to display them to the student.

B. Design and Implementation

Groovy is a programming language created by *Apache*. It is used in the implementation of the *process monitoring function*. *Groovy* is a dynamic programming language that operates efficiently on *Gradle*, and the multi-faceted language for the Java platform [33].

Fig. 6 shows the project layout to use this function. The students will *copy the module* and *paste* it into *Android Studio* to include the program in their projects. After that, they will refer to the *setting.gradle* section and add *including ':monitoringProcess'* before clicking *Sync Now*.



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to ... Sync Now Ignore these changes

```
pluginManagement { PluginManagementSpec it ->
    repositories { RepositoryHandler it ->
        gradlePluginPortal()
        google()
        mavenCentral()
    }
}
dependencyResolutionManagement { DependencyResolutionManagement it ->
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories { RepositoryHandler it ->
        google()
        mavenCentral()
    }
}
rootProject.name = "BasicActivityX"
include ':app'
include ':MonitoringProcess'
```

Fig. 6. Client-side program in *Android Studio*.

Then, the client-side program for the *process monitoring function* will be executed automatically after running the test code. As shown in Fig. 7, the corresponding script will be executed as a dependent of the *Gradle* task. Their roles are described as follows:

1) The student completing the task needs to build and execute the test code in the *src/test/java* directory.
2) Gradle builds the test code by running the *testDebugUnitTest* task.
3) *Android Studio* receives the results after running the test code.
4) The report on the test results will be produced by *Gradle* and be saved as an HTML file in the *build/reports/tests* directory.
5) *Gradle* will run the client-side program to copy the HTML report file, extract the student ID from *identity.txt*, and upload them to the server.

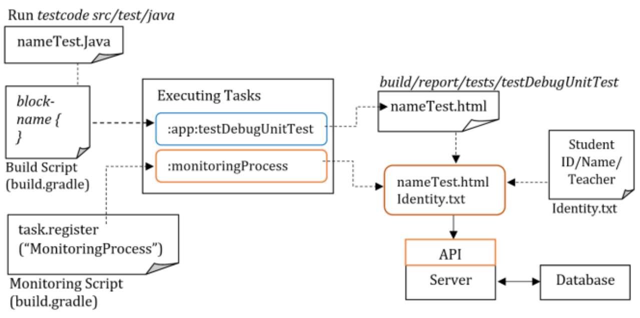
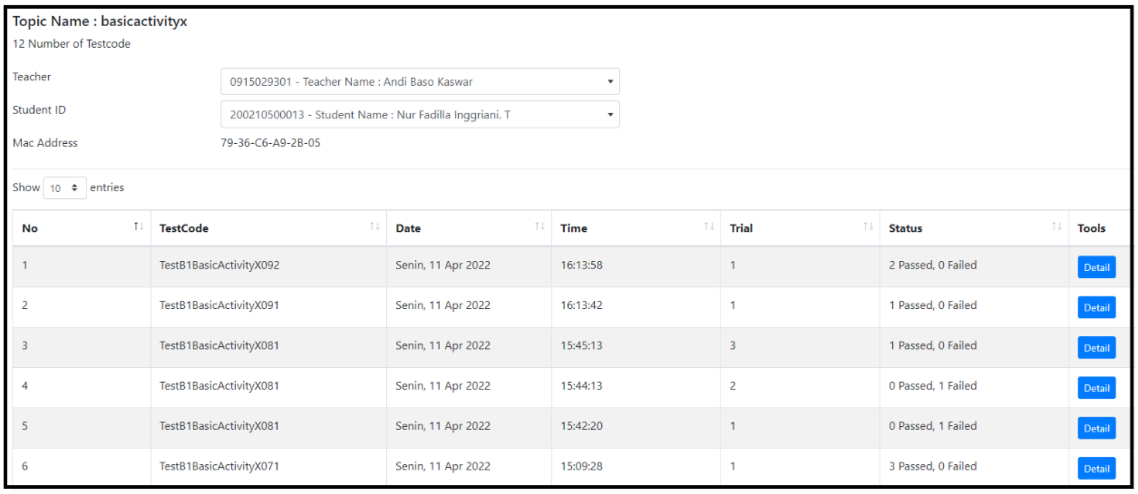


Fig. 7. Flow of processes.

C. Web Interface

The program for the web-based monitoring interface receives the progress reports from the database on a regular basis. Fig. 8 shows the user interface. The interface shows the current state of each task or tests code for a single learning topic. The teacher can know how far the student is in completing the tasks. This web interface shows the *MAC address* of the student's PC, the file names containing the *test codes*, the *date and time* for running the test in *Android Studio*, and the number of answers that have been submitted for validations.



Topic Name : basicactivityx
12 Number of Testcode

Teacher	0915029301 - Teacher Name : Andi Baso Kaswar
Student ID	200210500013 - Student Name : Nur Fadilla Inggriani. T
Mac Address	79-36-C6-A9-2B-05

Show 10 entries

No	TestCode	Date	Time	Trial	Status	Tools
1	TestB1BasicActivityX092	Senin, 11 Apr 2022	16:13:58	1	2 Passed, 0 Failed	Detail
2	TestB1BasicActivityX091	Senin, 11 Apr 2022	16:13:42	1	1 Passed, 0 Failed	Detail
3	TestB1BasicActivityX081	Senin, 11 Apr 2022	15:45:13	3	1 Passed, 0 Failed	Detail
4	TestB1BasicActivityX081	Senin, 11 Apr 2022	15:44:13	2	0 Passed, 1 Failed	Detail
5	TestB1BasicActivityX081	Senin, 11 Apr 2022	15:42:20	1	0 Passed, 1 Failed	Detail
6	TestB1BasicActivityX071	Senin, 11 Apr 2022	15:09:28	1	3 Passed, 0 Failed	Detail

Fig. 8. Interface for task status in one topic.

VI. Evaluation

This section evaluates the *process monitoring function* in *APLAS* through its application to students.

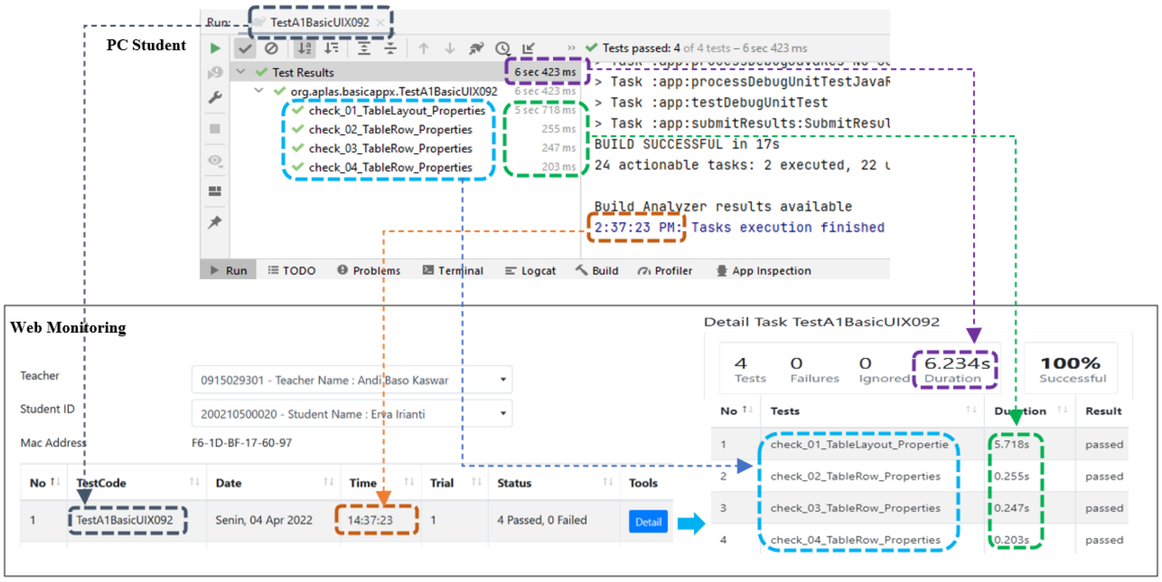
A. Evaluation Setup

For this evaluation, we prepared nine tasks with 11 test codes in the *Basic Activity* topic. Then, we asked 32 undergraduate students enrolled in Android programming courses at the IT department in Makassar State University, Indonesia, to solve them.

The students first downloaded the necessary files, including the programs for the *process monitoring function*, before solving the nine tasks using *Android Studio*. To help them in the installation of the programs, we provided the documents in Indonesian.

B. Process Monitoring Function Results

To ensure that the proposed function can operate without manual interventions, the students are requested to run the test codes independently in *Android Studio*. Then, the function's performance in transmitting and receiving data in real time is evaluated. As shown in Fig. 7, the validation results of the tasks in *Android Studio* were successfully shown on the web interface in real-time. Besides, as shown in Fig. 9, the details of one test were shown on the web interface. By checking the progress of the students using them, the teacher can find the students who may need care.



```
Tests passed: 4 of 4 tests – 6 sec 423 ms
Test Results                                  6 sec 423 ms
  org.aplas.basicappx.TestA1BasicUIX092       6 sec 423 ms
    check_01_TableLayout_Properties           5 sec 718 ms
    check_02_TableRow_Properties                    255 ms
    check_03_TableRow_Properties                    247 ms
    check_04_TableRow_Properties                    203 ms

> Task :app:processDebugUnitTestJavaR
> Task :app:testDebugUnitTest
> Task :app:submitResults:SubmitResul
BUILD SUCCESSFUL in 17s
24 actionable tasks: 2 executed, 22 u

Build Analyzer results available
2:37:23 PM: Tasks execution finished
```

Teacher	0915029301 - Teacher Name : Andi Baso Kaswar
Student ID	200210500020 - Student Name : Erva Irianti
Mac Address	F6-1D-BF-17-60-97

No	TestCode	Date	Time	Trial	Status	Tools
1	TestA1BasicUIX092	Senin, 04 Apr 2022	14:37:23	1	4 Passed, 0 Failed	Detail

Detail Task TestA1BasicUIX092

4 Tests, 0 Failures, 0 Ignored, 6.234s Duration, 100% Successful

No	Tests	Duration	Result
1	check_01_TableLayout_Propertie	5.718s	passed
2	check_02_TableRow_Properties	0.255s	passed
3	check_03_TableRow_Properties	0.247s	passed
4	check_04_TableRow_Properties	0.203s	passed

Fig. 9. Interface for status details in one task.

C. Solving Activity Result

Fig. 10 shows the solving performances of the students who completed nine tasks in the *Basic Activity* topic. It was discovered that the 32 students could finish all the tasks successfully. From the web interface, it was revealed that the average time required to complete the tasks among the students was 262min. The shortest time was 213min with running the test codes by 15 times, and the longest time was 311min with running the test codes by 25 times.

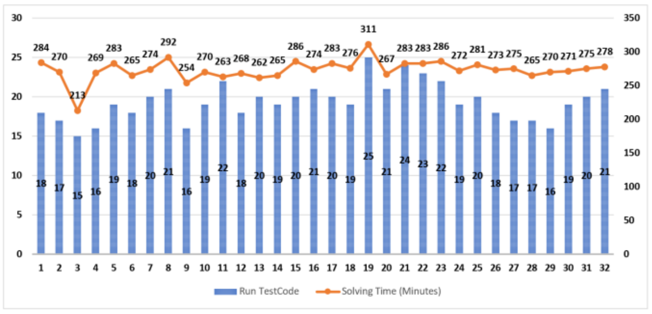


Fig. 10. Solving performances of 32 students.

Fig. 11 shows the rate of the students who run the test code only one time. The graph indicates that this rate is generally improved as the students solved more tasks more with the proper guidance from the teacher, especially for the students who may have difficulty solving the tasks in the *process monitoring function*.

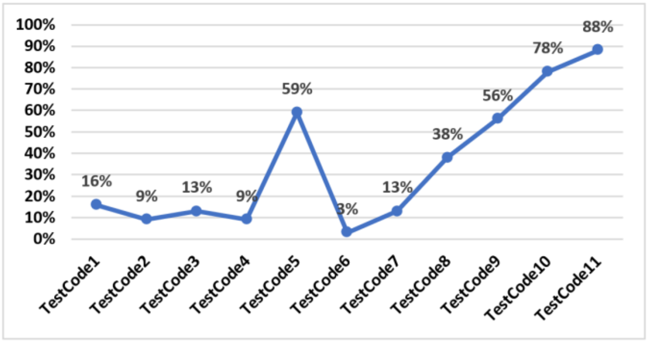


Fig 11. Students correct when first running test code.

D. Comparison with and without Proposal

Table I compares the number of students who failed each task with and without the *process monitoring function* [14]. Without the proposal, five students (12.5%) could not complete the task as a whole and were dropped-out. On the other hand, with the proposal, by allowing the teacher to monitor the progresses of the students, all the students successfully completed all the assignments. Thus, with the proposal, the reduction of the number of dropped-out students was observed.

TABLE I: COMPARISON OF FAILED STUDENTS WITH AND WITHOUT PROPOSAL

Task no.	1	2	3	4	5	6	7	8	9
Without Monitoring Function									
# of passed	39	40	40	39	38	40	40	40	39
# of dropped-out	1	0	0	1	2	0	0	0	1
With Monitoring Function									
# of passed	32	32	32	32	32	32	32	32	32
# of dropped-out	0	0	0	0	0	0	0	0	0

VII. CONCLUSION

This paper presented the implementation of the *solution progress monitoring function* in *Android Programming Learning Assistance System (APLAS)* that has been developed for self-learning by students. The function periodically collects log files of *Android Studio* and shows the progress at the web interfaces. For evaluations, we asked 32 students taking the Android programming course in the Department of Computer Engineering, Makassar State University, Indonesia, to solve nine tasks in *Android Programming Learning Assistant System* with this function. The results showed that using this function, the number of dropped-out students was reduced, because the teacher could identify and assist the student's experiencing difficulties at the early stage of the course. In future works, we will improve the function by including other aspects, such as hidden test codes and automatic grading, and use them to support programming education in Android programming courses.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

A. R. Patta mainly conducted the research and wrote the paper. N. Funabiki, M. Kuribayashi and Y.W. Syaifudin reviewed and finalized the paper. All the authors had approved of the final version.

ACKNOWLEDGMENT

The authors are very grateful to the students in State Polytechnic of Malang and Makassar State University in Indonesia who participated in the evaluation.

REFERENCES

[1] (May 14, 2022). Mobile operating system market share worldwide access. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide
[2] Statista GmbH. (May 14, 2022). App developers—statistics and facts. [Online]. Available: www.statista.com/topics/1694/app-developers/
[3] CC2020 Task Force, *Computing Curricula 2020: Paradigms for Global Computing Education*, Association for Computing Machinery, New York, NY, USA, June 2021
[4] 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. Syst.*, vol. 16, no. 1, pp. 115–135, 2020.
[5] Y. W. Syaifudin, N. Funabiki, M. Mentari, H. E. Dien, I. Mu'aasyiqiin, M. Kuribayashi, and W.-C. Kao, "A web-based online platform of distribution, collection, and validation for assignments in Android programming learning assistance system," *Eng. Lett.*, vol. 29, no. 3, pp. 1178–1193, Aug. 2021.
[6] P. G. F. Garcia and F.D. Rosa, "RoBlock—web app for programming learning," *Int. Journal of Emerging Technologies in Learning (IJET)*, vol. 11, no. 12, pp. 45–53, 2016.
[7] (May 14, 2022). JUnit: A simple framework to write repeatable tests. [Online]. Available: https://junit.org/junit4/
[8] (May 14, 2022). Robolectric: A framework that brings fast and reliable unit tests to Android. [Online]. Available: http://robolectric.org/
[9] L. Koskela, "Test driven: Practical TDD and acceptance TDD for Java developer, manning publications," *Shelter Island*, New York, NY 2008.
[10] N. Smyth, *Android Studio 3.0 Development Essentials* 8th ed., CreateSpace Independent Publishing Platform, 2017.
[11] (November 29, 2022). *Android Library: Google Developers*. [Online]. Available: https://developer.android.com/studio/projects/android-library/

[12] M. Wahid and A. S. Almalaise, "JUnit framework: An interactive approach for basic unit testing learning in software engineering," in *Proc. 3rd International Congress on Engineering Education (ICEED)*, pp. 159–164, 2011.

[13] M. L. Vásquez, C. B. Cardenas, K. Moran, and D. Poshyvanyk, "How do developers test Android applications?" in *Proc. IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 613–622, 2017.

[14] B. Sadeh and S. Gopalakrishnan, "A study on the evaluation of unit testing for Android systems," *International Journal on New Computer Architectures and Their Applications (IJNCAA)*, vol. 1, no. 4, pp. 926–941, 2011.

[15] C. A. Usener, T. A. Majchrzak, and H. Kuchen, "E-assessment and software testing," *Interactive Technology and Smart Education*, vol. 9, pp. 46–56, 2012.

[16] D. R. Almeida, P. D. L. Machado, and W. L. Andrade, "Testing tools for Android context-aware applications: a systematic mapping," *J Braz Comput. Soc*., vol. 25, no. 12, pp. 1–22, 2019.

[17] (Nov. 29, 2022). Test in Android: Google Developers, test in android studio. [Online]. Available: https://developer.android.com/studio/build

[18] H. F. Hanafi and K. Samsudin, "Mobile learning environment system (MLES): The case of Android-based learning application on undergraduates' learning," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 3, no. 3, pp. 1–5, 2012.

[19] U. Köse, "A web-based system for project-based learning activities in web design and programming course," *Procedia-Social and Behavioral Sciences*, vol. 2, no. 2, pp. 1174–1184, 2010.

[20] T. C. Yang, S. J. Yang, and G. J. Hwang, "Development of an interactive test system for students' improving learning outcomes in a computer programming course," in *Proc. IEEE 14th International Conference on Advance Learning Technology*, Athens, Greece, pp. 637–639, 2014.

[21] Y. Hayashi, K. I. Fukamachi, and H. Komatsugawa, "Collaborative learning in computer programming courses that adopted the flipped classroom," in *Proc. International Conference on Learning Technology and Computer Engineering*, Taipei, Taiwan, pp. 209–212, 2015.

[22] C. Hundt, M. Schlarb, and B. Schmidt, "SAUCE: A web application for interactive teaching and learning of parallel programming," *J Parallel Distributed Computer*, vol. 105, pp. 163–173, July 2017.

[23] J. M. Su and S. J. Wang, "A Web-based learning activity integrated with scratch tool to support programming learning," in *Proc. the 10th International Conference on Ubi-media Computer and Workshops (Ubi-Media)*, Pattaya, Thailand, pp. 1–4, 2017.

[24] H. A. A. Rekhawi and S. S. A. Naser, "Android applications UI development intelligent tutoring system," *International Journal of Engineering and Information Systems (IJEAIS)*, vol. 2, no. 1, pp. 1–14, 2018.

[25] D. Amalfitano *et al*., "Combining automated GUI exploration of Android apps with capture and replay through machine learning," *Information and Software Technology*, vol. 105, pp. 95–116, January 2019.

[26] F. Domenach and G. Portides, "PASS—a programming assignment submission system," in *Proc. the International Conference on Interactive Mobile Communication Technologies and Learning (IMCL)*, pp. 195–199, November 2015.

[27] S. Chaudhari and N. J. Uke, "A method for submitting programming language assignment for effective evaluations," in *Proc. the Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–4, 2018.

[28] M. Madeja and J. Poruban, "Automatic assessment of assignments for Android application programming courses," in *Proc. the IEEE 14th International Scientific Conference on Informatics*, pp. 232–237, 2017.

[29] I. A. Khan, M. Iftikhar, S. S. Hussain, A. Rehman, N. Gul, W. Jadoon, and B. Nazir, "Redesign and validation of a computer programming course using inductive teaching method," *PLoS ONE*, vol. 15, no. 6, pp. 1–21, 2020.

[30] (May 14, 2022). [Online]. Available: Gradle user manual version 6.7. https://docs.gradle.org/6.7/release-notes.html

[31] G. Maudoux and K. Mens, "Correct, efficient and tailored: The future of build systems," *IEEE Software*, pp. 1–6, 2018.

[32] Y. W. Syaifudin, N. Funabiki, and M. Kuribayashi, "Implementation and performance evaluation of unit testing for student's answer validation in Android programming learning assistant system," *Technical Program of Life Intelligence and Office Information Systems (LOIS) - IEICE*, pp. 1–6, 2021.

[33] (Dec. 5, 2022). *Apache Groovy*. [Online]. Available: https://groovy-lang.org/