A Web-based Online Platform of Distribution, Collection, and Validation for Assignments in Android Programming Learning Assistance System

Yan Watequlis Syaifudin, Nobuo Funabiki, Mustika Mentari, Habibie Ed Dien, Ikhlaashul Mu'aasyiqiin, Minoru Kuribayashi, and Wen-Chung Kao

***Abstract*—Currently, *Android smartphones* have dominated the distribution of mobile devices around the world. Then, with high demands for *Android programming* jobs, large numbers of schools, universities, and professional training institutions offer related courses. The *Android Programming Learning Assistance System (APLAS)* has been developed to accommodate independent and automatic assisting learning for students to learn Android programming by adopting *test-driven development (TDD) method*. APLAS offers learning materials to systematically study Android programming through solving topics with four stages and provides features for automatic validation of the student's answers by running given test codes. However, a platform to support various activities of teachers for handling a lot of students in the courses is not implemented yet, and it made their load become very high. This paper presents the proposal of online platform in APLAS for distributing the learning materials, collecting the student's answers, and validating them, based on a web application system. By using *Gradle*, a *validator program* was developed for running the test codes automatically in the server when a new answer is submitted. A comprehensive evaluation has been applied by asking 60 undergraduate students in an Indonesian university to use the online platform, solve three APLAS learning topics, and submit the answers. The results showed that the three functions worked properly, and the validation process of 183 submissions from all students delivered correct results on the server. The stability, robustness, effectiveness, performance, and security of the online platform were confirmed by analyzing the data of users' activities and validation results. The significant reduction in processing time on all three functions also confirmed its effectiveness and improvement. Finally, the students' feedback proved its usability.**

***Index Terms*—APLAS, Android programming, automatic validation, web application, online platform, unit testing**

Manuscript received Nov. 17, 2020; revised June. 28, 2021.

Yan Watequlis Syaifudin is a PhD candidate in Graduate School of Natural Science and Engineering, Okayama University, Okayama, Japan, e-mail: p52m5jeb@s.okayama-u.ac.jp. He is also with Department of Information Technology, State Polytechnic of Malang, Indonesia, email: qulis@polinema.ac.id.

Nobuo Funabiki is a professor in Department of Electrical and Communication Engineering, Okayama University, Japan, e-mail: funabiki@okayama-u.ac.jp.

Mustika Mentari is a lecturer in Department of Information Technology, State Polytechnic of Malang, Indonesia, email: must.mentari@polinema.ac.id.

Habibie Ed Dien is a lecturer in Department of Information Technology, State Polytechnic of Malang, Indonesia, email: habibie@polinema.ac.id.

Ikhlaasul Mu'aasyiqiin is a bachelor candidate in the Department of Information Technology, State Polytechnic of Malang, Indonesia, email: ikhlaashul.ti@gmail.com.

Minoru Kuribayashi is an associate professor in Department of Electrical and Communication Engineering, Okayama University, Japan, e-mail: kminoru@okayama-u.ac.jp.

Wen-Chung Kao is a professor in Department of Electrical Engineering, National Taiwan Normal University, Taipei, Taiwan, e-mail: jungkao@ntnu.edu.tw.

I. Introduction

In this era of mobility, smartphones have become more popular than personal computers (PCs) [1][2]. In 2019, more than 1.5 billions smartphones were sold to end-users while around 260 millions PCs were sold [3]. This amount is estimated to increase in the following years continously.

Among operating systems for smartphone devices, *Android* has become at the top compared to other operating systems. Latest evidence shows that over 70% of smartphone devices in the entire world used Android as the main operating system[4]. Besides, *Google Play* as the main distribution service site for Android applications becomes the most demanding store of mobile applications. It actually covers at least 2.5 million Android apps at the beginning of 2020 [5].

In connection with these facts, the demand for Android application programmers is continuing to increase. Currently, Android application programmer becomes one of the most wanted jobs in various industrial sectors, especially in IT [6]. With this demand, large numbers of schools, universities, and professional training institutions have carried out some courses specific to *Android programming*, despite the fact that programming skills are very important in IT field [7].

To master a programming language for students, a tool or a system to study it by themselves is useful and often indispensable. Therefore, to assist students studying Android programming independently, the *Android Programming Learning Assistance Systems (APLAS)* has been developed [8] that provides assignments to students in several learning topics and accommodates them to validate the answers instantly and automatically.

In APLAS, student has to use *Android Studio* to accomplish the assignment answers. The adoption of *test-driven development (TDD) method* [9][10] enables students to confirm the correctness by running the given test codes without manual checking by teachers. APLAS uses together *JUnit* [11] and *Robolectric* [12] to apply *unit testing* for the source codes in student's answers that are written by *Java* and *XML*.

However, we did not implement the platform to support various activities of teachers to handle a lot of students in programming courses which made their load becomes very high. A teacher needs to distribute several files of learning materials which include *guide documents*, *supplement files*, and *test codes* for each of 13 learning topics to the students manually [8]. The teacher also needs to collect and validate

answers from students. Each learning topic in APLAS consists of 8 to 12 tasks, where each of them consists of some test codes. Thus, in the case of a class that has 50 students, the teacher must run more than 400 test codes to confirm the correctness of all students' answers for only one topic. This circumstance causes difficulties for a teacher to apply APLAS in his/her class [13].

In this paper, we propose an *online platform* based on *web application system* for APLAS which offers *three functions* including distributing the learning materials, collecting the student's answers, and validating them automatically. It offers interactive user interfaces on web browser, adopts *database system* using *MySQL* in the server to store learning data and source codes from students, and provides a *validator program* to validate the source codes by running test codes automatically in the server.

As an essential part of this platform, the validator program is implemented by Java as a thread program. It utilizes an advanced build tool, *Gradle* [14], to handle the compiling and building process of the source codes in the form of an Android project and perform unit testing by running the test codes on *Java Virtual Machine* with *Android SDK* library [15]. As a thread program, the validator program runs as a background process because it always runs as long as the server is on. When it detects a new answer submission from a student which contains several source codes, it will validate them and automatically store the results in the database.

To apply a comprehensive evaluation, we asked 60 undergraduate students in an Indonesian university to use it by solving three APLAS learning topics[16][17][18] and submitting the answers, then analyzed their activities and the validation results of their answers. The results show that the online platform delivered correct results in validating all students' answers. The students properly obtained the learning materials and submitted their answers using web browsers, and the *validator program* correctly completed the validations of 183 students' answers on the server. The time for students to solve the three topics was improved compared to the former studies. This online platform was also compatible with various web browsers that are used by students. The effectiveness of the three functions was also confirmed that this platform could reduce the teacher's load and eliminate the main problem in the learning process with APLAS. Performance on answer validation is better than the manual validation process which produces longer times. The system security in three aspects is also confirmed, including system protection, information authorization, and data protection. Finally, the 60 students gave mostly positive opinions in using this platform.

The sections of this paper are arranged as follows: Section I explains the background and the brief introduction of this research. Section II presents published research related to this paper. Section III shows the review of APLAS as a self-learning system for Android programming learning. Section IV presents the proposal of online platform for APLAS. Section V explains the implementation of online platform based on web applications with the three functions. Section VI shows the automation of the validation process by validator program. Section VII evaluates the effectiveness of the proposal. Finally, Section VIII concludes the results of this research and presents our future works.

II. Related Works in Literature

This section presents the discussion on related research in literature to this paper.

A. Automatic Validation

Several papers reported automatic validations of student answer codes for independent programming studies [19].

Funabiki et. al. presented a web-based *Java programming learning assistant system* or *JPLAS* in 2013 for assisting students learning Java programming [20]. *JPLAS* utilizes JUnit as a unit testing framework to implement answer validation on source codes in Java from students. Some improvements for student's learning method for Java programming were also implemented [21].

In 2014, Yulianto et al. presented a source code analyzer for automatic grading of student's programming assignments namely *SCAGrader* [22]. This application consists of a Java-based auto grader engine and a web application for user interface.

Staubitz et. al. proposed a platform based on web application in 2016, called *CodeOcean* [23]. It is used to support practices in any programming courses and must be applied in MOOCs to support teachers in preparing practices for students in learning programming. By using unit testing, the student's results can be graded instantly.

In 2019, a tool has been proposed by Khan et. al., called *AUTOGRADER*, to determine the validity of student's answers on programming tasks automatically[24]. It can measure semantical differences between the code and the reference, and decrease the teacher's efforts in defining test suites to make the grading impeccable.

B. Online Platform

Several papers reported online platforms to manage programming learning materials and results for teachers and students.

In 2010, a web-based system for learning web programming and design with a programming language has been introduced by Köse [25]. It creates a prototype of the learning management system (LMS) for programming and can integrate the roles of teachers and students. Additionally, Daehnhardt et. al. also presented with semantic web [26].

Bosnić et. al. presented *ORVViS*, a feature for *Moodle* as the most popular LMS. It is intended to confirm source codes as assignment answers from students [27]. The function for automatic validation of source codes is not provided by this feature.

In 2014, a system to provide an interactive examination for programming courses has been developed by Yang et. al. [28].

In 2015, Hayashi et. al. introduced a model for collaborative learning on programming courses by adopting the flipped classroom [29].

Hundt et. al. presented an interactive web application system called *SAUCE (System for AUtomated Code Evaluation)* to learn parallel programming [30].

In 2017, Su et. al. introduced an application based on the web application to assist learning on programming concepts using scratch tool [31]. The material for object-oriented programming concept is also provided by this application [32].

C. Android Programming Learning Platform

Some papers reported implementations of Android programming learning platforms.

In 2017, Kang et. al presented Android programming education using on *MIT App Inventor* that based on scratch [33].

Rekhawi et al. developed an intelligent tutoring system based on web application. Similar to MOOC, it offers lessons on various topics of Android application development [34].

Amalfitano et. al. introduced *jugular*, a technique to detect explored GUI components during the utilization of application [35].

In 2020, Zhao et. al. proposed *APIHelper*, a tool that enables junior programmers to learn Android API by managing various APIs in the Android application [36]. They focused on APIs related to permission, energy-consuming, and advertising.

III. Review of Android Programming Learning Assistance System

This section reviews APLAS as self-learning system for Android programming and its drawbacks on the previous studies[8][18].

A. Overview

APLAS is an automatic assisting learning and self-learning system that focuses on Android programming learning using Android Studio. By using a practical learning approach, it provides assignments that must be solved by students. By adopting the *test-driven development (TDD)* method [10][37], any *assignment's answers* from students can be validated automatically without manual checking by teacher. The validation of *assignment's answers* that consist of *source codes* in Java and XML is performed by applying the testing process. The answer codes are checked by running *test codes* that were granted for the specific assignment. APLAS performs two types of testing, namely, *unit testing* and *integration testing*, based on Model View Presenter (MVP) model [38]. The *JUnit*[39] is adopted for unit testing, and *Robolectric* is used for integration testing [12][40].

Android Studio, the main Integrated Development Environment (IDE) for developing Android applications[36], is used by students to write the source codes. By default, it is integrated with Android SDK [41], a package of libraries developed by Google to compile source code become Android application. It requires writing source codes by combinations of *Java* codes and *XML* codes. By running on Java Virtual Machine with JDK 8, as presented in Figure 1, APLAS supports major operating systems such as Windows, Linux, and Mac OS.

B. Learning Materials

APLAS provides 13 *learning topics* to cover a wide range of Android programming subjects. As illustrated in Figure 2, one *learning topic* has an application for assignment and several *tasks* to solve the assignment step by step. One task represents a primitive assignment with a specific learning objective.

For each learning topic, the teacher provides a package of learning materials that consist of *guide documents*, *test codes*, and *supplement files*. The guide document is intended to direct the student in solving one task of the learning topic. It contains learning objectives, required specifications of hardware and software required for solving the learning topic and guidance on how to achieve the task. The test codes are provided to confirm the correctness of source codes by running them on Android Studio. The supplement files cover the additionally required files for the assignment, such as fonts, images, video, and styles resources.

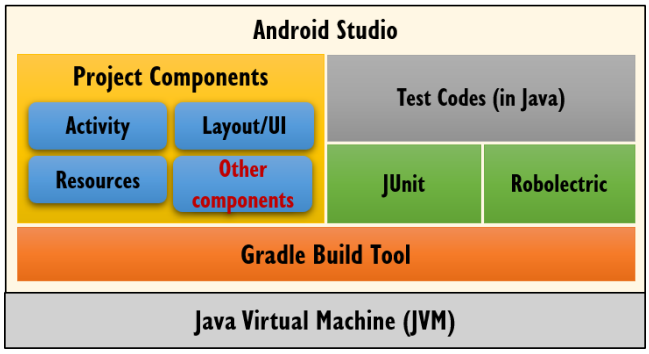

Fig. 1. Software architecture of APLAS.

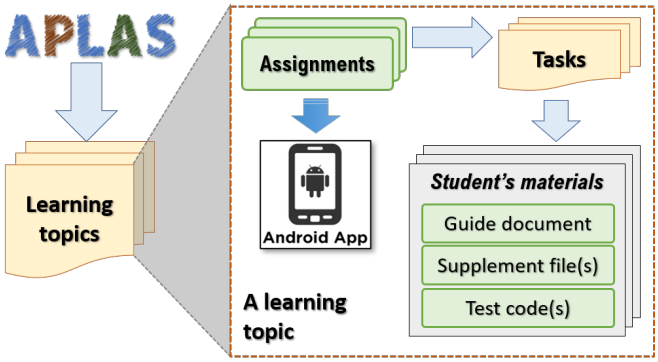

Fig. 2. Structure of learning materials in APLAS.

C. Learning Process by Teacher and Students

The learning process of APLAS contains three stages:

1) Distribution: In the classroom, the teacher will distribute the learning materials by using a flash disk or sending emails to his/her students. Then, students will read the guide documents using a PDF reader software, and develop source codes for the assignments using Android Studio, as shown in Figure 3.

2) Collection: After solving the assignments, the teacher will collect their answers by asking students to send the zipped Android project folder by using a flash disk or sending an email. Then, he/she must copy or download them to his/her PC and unzip them.

3) Validation: To confirm the correctness of students' answers, the teacher needs to read and check the source codes. Here using Android Studio, he/she must open the Android project folder, and verify the source codes using the test codes.

D. Unit Testing for Validation

In developing software or application, unit testing performs an essential role to verify the achievement of the expected specifications in the source code [42][43]. A study[40]

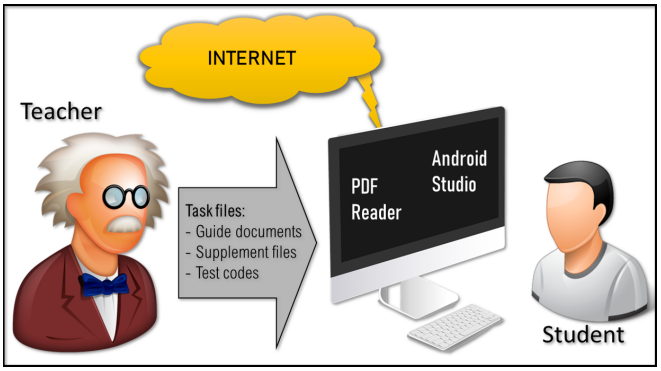

Fig. 3. Learning environment in APLAS.

confirmed JUnit and Roboletric as the main Java-based testing frameworks for Android applications. Another study[44] shows similar results that confirm the highly adopted of *JUnit*[45] and *Robolectric* as the testing tools for Android programming. Robolectric is used to simulate an Android application using shadow objects and cause the unit testing is possible to implement on the Android environment.

As shown in Figure 4, Java source codes can be directly tested by JUnit using *assertion methods*. However, the components that are specific to Android applications, such as UI components, the *Activity* class, the *Content Provider*, and the application's resources, cannot be directly tested by JUnit. The *Gradle*, a build tool, must build and integrate them as Java classes, then Robolectric will generate *Shadow objects*, the Java objects as the simulation of the Android application running on Java Virtual Machine environment, instead of running on a smartphone device or emulator. Then, on the *Shadow objects*, JUnit performs testing that makes Android application testing becomes faster [46].

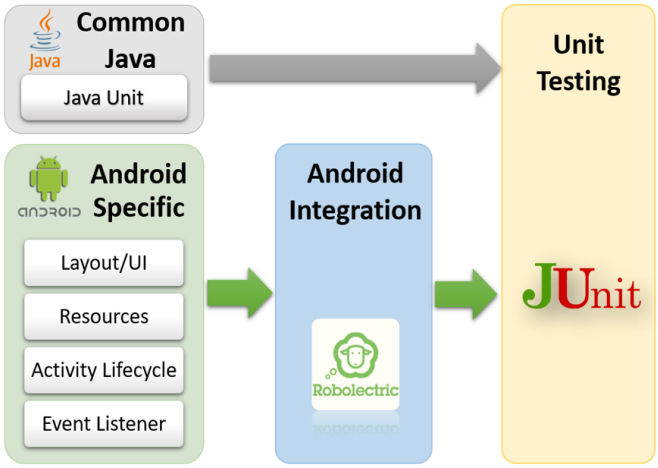

Fig. 4. Validation process of Android project.

E. Encountered Problems

In this subsection, we describe problems encountered in the previous implementations of APLAS.

1) Interactions between Teachers and Students: In the previous implementation of APLAS, there is no platform for assisting the interactions between teachers and students. Thus, several procedures in the learning process were performed manually, which drastically increases the teacher's load to deal with a lot of students. Then, a platform is essential to integrate the functions that can be used by teachers and students.

2) Problem at Learning Material Distributions: The manual distribution of learning materials of 13 learning topics to students becomes more difficult for teacher. For 32 files in each topic, the teacher must deliver more than 400 files with sizes of more than 100 MB. He/she also has to prepare them in structured folders. The distribution by email produced a size limitation problem and by flash disk affected students' queue in the copying process. Additionally, distribution by flash disk does not support online courses.

3) Problem at Answer Collection: To collect assignment's answers, the teacher asks his/her students to zip them and submit them by using a flash disk or sending emails. Collecting by flash disk produced a queue in the classroom at the copying process and prone to mistakes. Collecting by sending emails also caused a problem in the limitation of attachment file size and the rejection of source code files by its antivirus. After receiving the files of assignment's answers, the teacher must copy or download them to PC and unzip them one by one to become an Android project.

4) Problem at Answer Validation: To validate the student's answers, the teacher must copy and open first them on Android Studio one by one. Then, he/she must run the test codes one by one, and check the results manually. These processes take a long time due to synchronizing and building processes on Android Studio. The process of validating a student's answer using 12 test codes needs at least 32 mouse clicks and 18 minutes. If there are 50 students, 1,600 mouse clicks must be performed, and 15 hours must be spent to finish all the validation procedures for a topic. Additionally, the teacher must save the validation results by writing them in a file.

IV. Proposal of Online Platform for APLAS

This section explains the proposal of online platform of APLAS to fix the encountered problems in the previous version.

A. Overview

To integrate the functions in distribution, collection, and validation process by teachers and students, we propose an online platform to make the learning process with APLAS easier. This online platform can simplify the distribution of learning materials to students by teacher, collection of answers and source codes from students, and validation process of source codes. Using the web application, students can efficiently acquire specific learning materials and use them for learning on their PCs. Then, they can submit the answers using the web application and get the validation results instantly. Moreover, the teacher can monitor his/her students' learning progresses and students can check their learning achievements.

B. Client-server Model

The client-server model is applied for the online platform, as presented in Figure 5. The *server-side* provides the web application system that can be accessed by teachers and students. For students, a *client-side* is a PC that is used to

access the web application by using a web browser and solve the assignments by using Android Studio.

C. Operation Flows of Online Platform

Figure 5 shows the operation flows of the online platform consisting of the *client-side* for student's PC and the *server-side*. The following procedure is conducted for each learning topic:

- Teacher uploads the learning materials for the topic to distribute them to the students.
- Each student downloads the learning materials and solves the assignments in them by writing Java, XML, or DSL of Gradle source codes using Android Studio on his/her PC.
- Student validates the source codes using the test codes in the materials on Android Studio.
- Student submits the answer codes and the learning reports to teacher using the web application.
- The answer codes are validated automatically on the server-side and the results are stored in the database.
- The teacher and the student access the results using the web browser.

D. Functions of Online Platform

As the main part of online platform, the server-side has three functions.

1) Online Distribution: The online platform is used to distribute learning materials to students through the web application system. Teacher can upload the packages of the learning materials to the server, which contain guide documents, supplement files, and the test codes that are necessary to solve the assignments. Then, students can download them. This function is essential to deliver a lot of files from teacher to students safely and easily.

2) Online Collection: The online platform is used to collect the answers on any assignments from students through the web application system. The assignment's answers for a learning topic include several source codes of Java codes, XML codes, or DSL for Gradle configuration. The students can upload them and the teacher can directly access them. The teacher does not need to download them, where he/she can see them through the web application. This function is used to simplify the collection process of student's answers.

3) Online Validation: The online platform is used to automate the validation process of assignment's answers on the server by *validator program*, which can save the loads of validating assignment's answers from a lot of students manually. After the validation, both the teacher and the student can get validation results directly through the web application.

V. Implementation of Web-based Online Platform

This section presents the implementation of web application, validator program, and database system of the online platform for APLAS.

A. Software Architecture

The client-server architecture of the online platform for APLAS is presented in Figure 6. The client-side is a student's PC that is used to solve the assignments or tasks using Android Studio and access the server using a web browser. The server-side is a server system that provides a web application server, runs the validator program, and implements the database system to manage any data. Table I shows the software specification for the implemented online platform.

TABLE I
SOFTWARE SPECIFICATIONS FOR ONLINE PLATFORM.

section	specification
client-side:	
IDE	Android Studio 4.0, Gradle 6.0.2, Java OpenJDK 1.8, JUnit 4.13, Robolectric 4.2.1
server-side:	
Web application server	Apache server 2.4.29, Laravel 6, PHP 7.4.1
database	MySQL 5.7
validator	JUnit 4.13, Robolectric 4.2.1, Java OpenJDK 1.8, Android SDK 26, Gradle 6.0

B. Web Application Server

The web application server offers the interface for the users. The user authentication function is implemented for the authorization of three user types, namely, the student, the teacher, and the administrator. Figure 7 illustrates the top page, where a Youtube video is prepared to explain how to start this application, how to start learning with APLAS, and how to submit the answers.

1) Features for Student: The student has three features as described in Table II, *Start Learning*, *Submit Answers*, and *Validation Results* to provide the three functions of online platform. He/she can get learning materials, submit the answer codes, and get the validation results of the submitted answers.

TABLE II
STUDENT'S FEATURES ON THE WEB APPLICATION.

no	feature	description	function
1	Start Learning	the feature to explore and download learning materials	Distribution
2	Submit Answers	the feature to submit the answer codes	Collection
3	Validation Results	the feature to check results of answer's validation	Validation

2) Features for Teacher: The teacher has five features as described in Table III, *Classrooms*, *Student Member*, *Learning Topic*, *Assignment Results*, and *Completeness* to provide the three functions and classroom management. He/she can get manage classroom, submit the answer codes, and get the validation results of the submitted answers.

C. Database System

MySQL, as a reliable relational database system [26], is adopted to store and manage the data in APLAS. The entities related to learning management involve *Users*, *Classroom*,

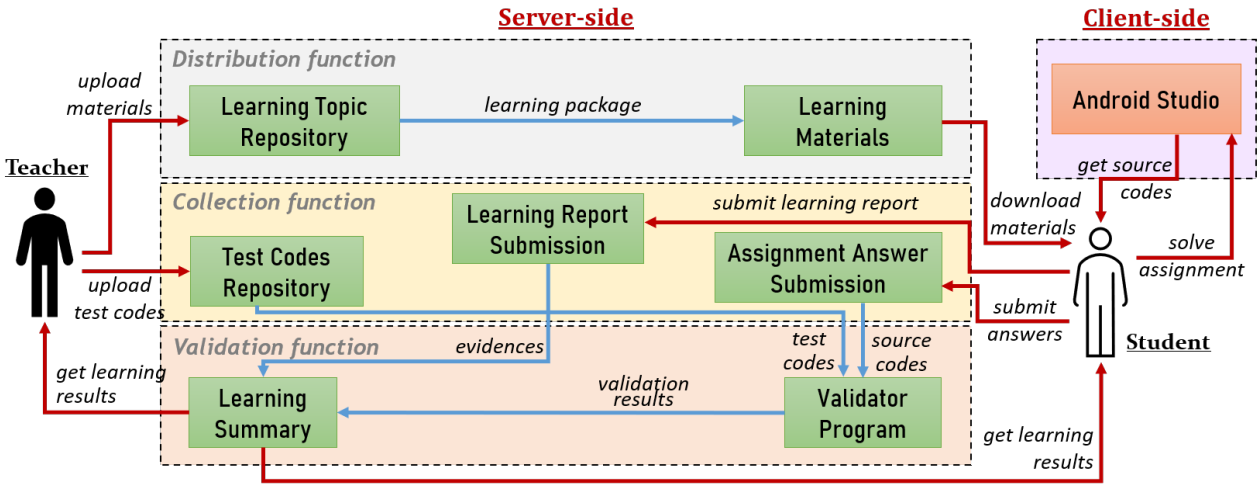

Fig. 5. Operation flows of online platform on client-side and server-side.

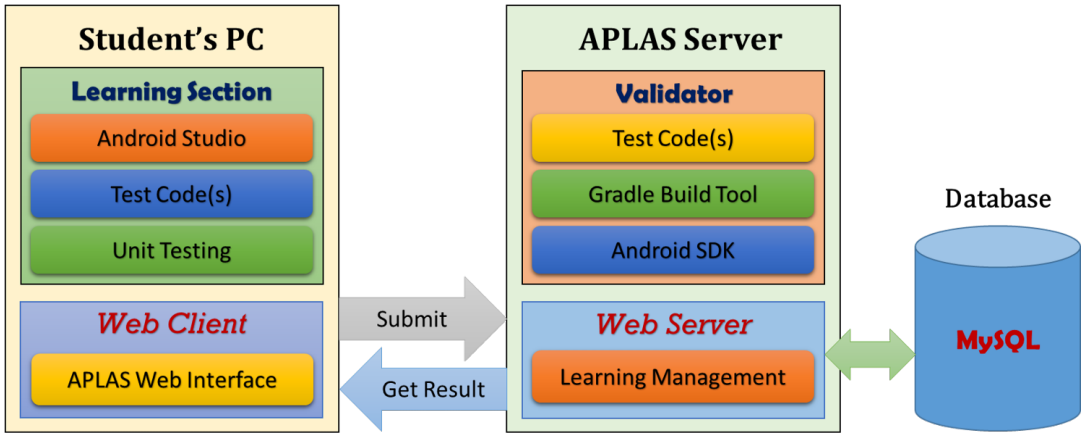

Fig. 6. Software architecture.

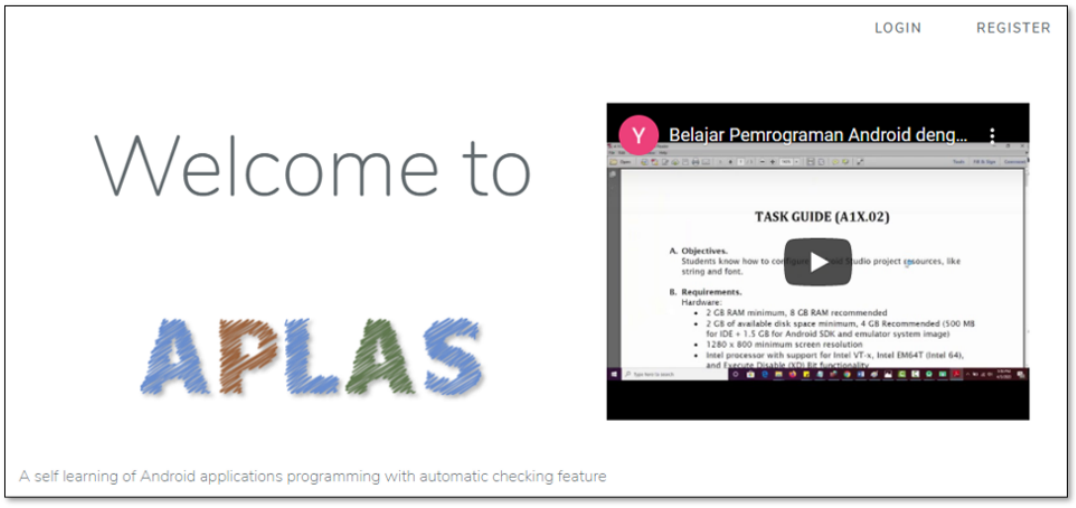

Fig. 7. Top page of online platform.

Learning topics, *Learning tasks*, *Learning stages*, *Topic files*, and *Test codes*. The entities related to learning process include *Student submissions*, *Learning evidence*, *Submission files*, and *Validation results*. In addition, there are default tables from *Laravel* used for the web application management.

In APLAS, a student needs to submit related files to the assignment, such as the pictures for learning evidence and the files in the developed Android project folder including the Java source codes, the project manifest file, the resource files with XML codes, and the Gradle configuration. These files are uniquely renamed and are stored in the folder in the server that is accessible to the web application and the validator program.

TABLE III
TEACHER'S FEATURES ON THE WEB APPLICATION.

no	feature	description	function
1	Classrooms	the feature to create a new classroom for students	Distribution
2	Student Member	the web page to show the list of the students and their achievements in solve the assignments	Distribution
3	Learning Topic	the feature to create and manage learning materials which can be used by students	Distribution
4	Assignment Results	the feature to see the submission of answers from each student and check the source codes	Collection
5	Completeness	the page to see the completeness of each student in solving the given topics in APLAS	Validation

D. Validator Program

Validator program is the thread program written with Java and runs as a background job on Linux operating system [48]. It is an important part of the server-side to perform automatic validation of student's answers. It always runs to detect a new answer submission and validate it by calling Gradle to compile the folder containing the Android project and running every test code to confirm the correctness of developed source codes in the folder. Gradle will run as a command-line program and produce output as text messages.

In the validation process, Gradle sets up the configuration of Android SDK used in the Android project. Then, it compiles the project by integrating the required components defined in the 'build.gradle' file. To run the test codes, Gradle compiles JUnit and Robolectric libraries, where this validation process runs in the background without using Android Studio. The validator program records the test results, the execution time, the error messages (if occur), and the cause error messages in the database. They can be checked by both students and teachers using the web interface.

E. Implementation of Three Functions

In this subsection, we explain the implementation of three functions in the web-based online platform.

1) Distribution Procedure: By web application, the teacher can create learning topics for students and upload the necessary files of guide documents, test codes, supplement files, and other files if needed. Figure 8 shows an example for a teacher creating *'B1 - Basic Activity'* topic with uploading its package of learning materials. Then, the teacher has to configure tasks in the learning topic by uploading test code(s) for each task. After that, students can access the new learning topic and can download its learning materials through the web application. Figure 9 shows a web page for students to download the learning materials of 'B1 - Basic Activity' topic. Then, they can start learning process by developing source codes for each task that is described in each guide document, and validates them using the test codes on the client-side.

2) Collection Procedure: After solving a learning topic, a student needs to submit the learning report and the assignment answers. The learning report includes validation status (passed or failed), duration time for the task completion (in min.), learning evidence (image files for test code results), and comments on solving tasks if any. He/she also needs to submit source codes in the developed Android project folder. They are Java source code, XML source code, and DSL of Gradle configuration. Figure 10 shows the interface to submit the learning report and the assignment answers of 'B1 - Basic Activity' topic.

3) Validation Procedure: When a student submitted the assignment answers, the validator program will process them. It executes the related test codes to validate the answers automatically. The validation results can be presented to the student and his/her teacher on a web interface. Figure 11 shows the web interface to check the validation results of *'B1 - Basic Activity'* topic with a failure result. There are passed results from 10 test codes and one failed result from the test code *'TestB1BasicActivity051.java'* with the reason for the failure.

VI. ONLINE VALIDATION PROCESS

This section presents the details of automated validation process by the validator program on server-side.

A. Building Process in Android Application

To test an Android application, a programmer needs to compile and build the Android project that contains source codes, related resources, and some dependency libraries. As illustrated in Figure 12, the building process contains several steps to produce an APK file, an executable Android application on Android device.

1) Gradle as Manager: By default, Android Studio utilizes Gradle [49] as an advanced build tool to manage the automation of building process of Android applications. Actually, Gradle is not a real compiler. It manages the process of source code compilations, file links, test codes running, and packaging for an Android project [42], by utilizing Android SDK Build Tools [51] as the component of Android SDK. The Android SDK Build Tools contains *Android Asset Packaging Tool* (AAPT) [52] and *d8* tool [53].

2) Compilation Process: In the compilation process, the source codes are transformed into Java classes by the compiler of JDK. Then, the d8 tool converts the Java classes into a *DEX (Dalvik Executable)* [54] file. By AAPT, the DEX file and the compiled application's resources are packaged into an *APK* (Android Package) [55] file. In this process, Gradle and Android SDK can run independently on Android Studio. Moreover, Gradle can run from command-line to build or test an Android project.

B. Unit Testing Process

Gradle also conducts unit testing process of Android project by using Robolectric and JUnit as shown in Figure 13. It needs a building process before creating tested objects by Java compiler or Robolectric. Then, JUnit will be used to perform unit testing on the tested objects. The process will generate text messages describing the status of test results. Thus, the unit testing is executed from command-line by the validator program.

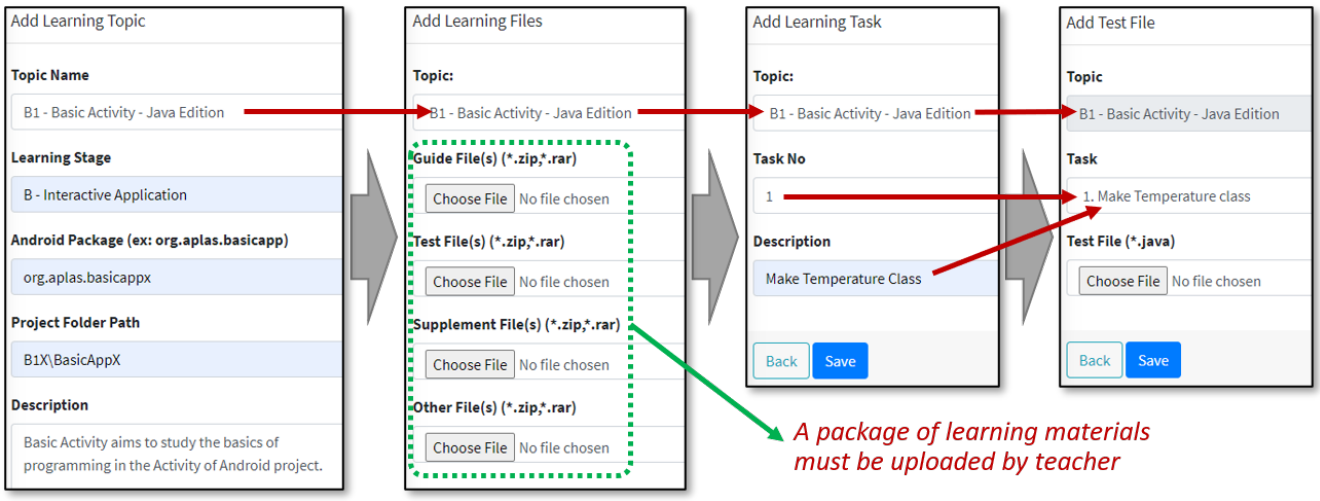

Fig. 8. Web interface to upload learning materials for distributions by teacher.

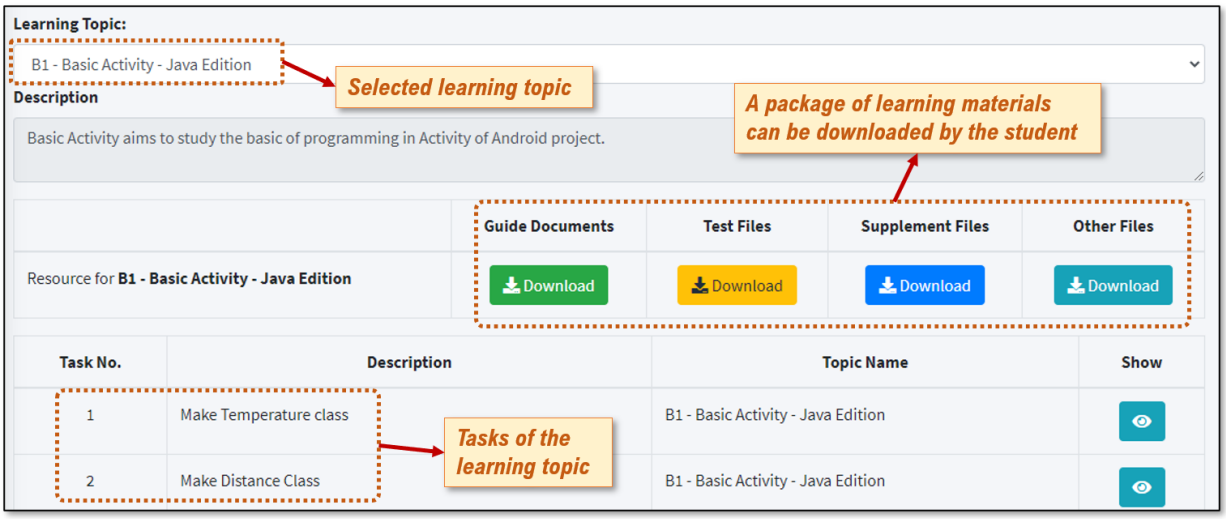

Fig. 9. Web interface to access and download learning materials by student.

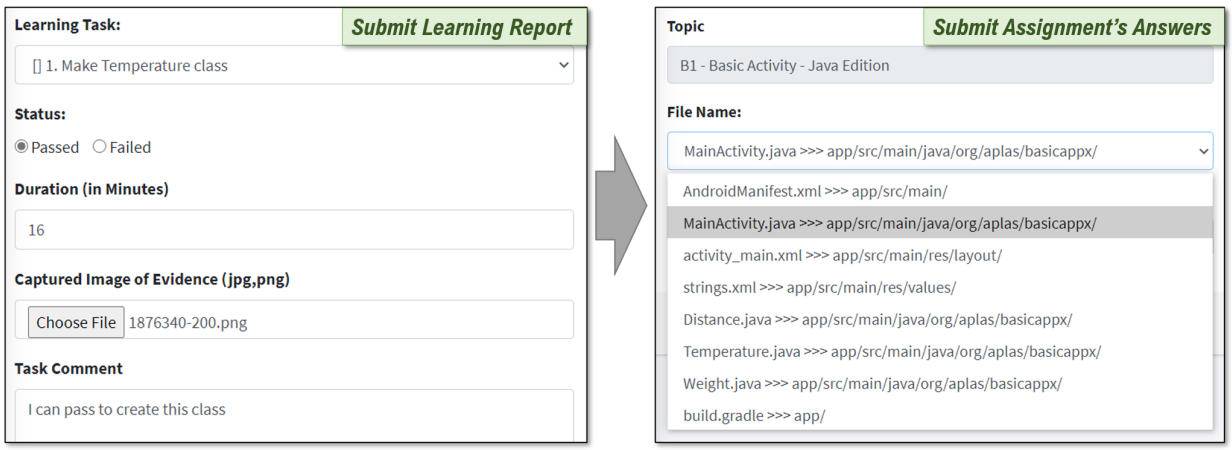

Fig. 10. Web interface for answer submission.

C. Modules in Validator Program

To automate the validation process of each student's answer, four modules in Figure 14 are implemented in validator program.

1) Submission Detector Module: The *submission detector module* always checks the data of student's answer submissions in the database. If a new submission is detected, this module will create a new folder containing an Android project that matches its topic. It will set the folder ready to be tested by copying the associated test codes, copying the submitted source codes, and making the executable Gradle script file. Then, the module will inform the path of generated

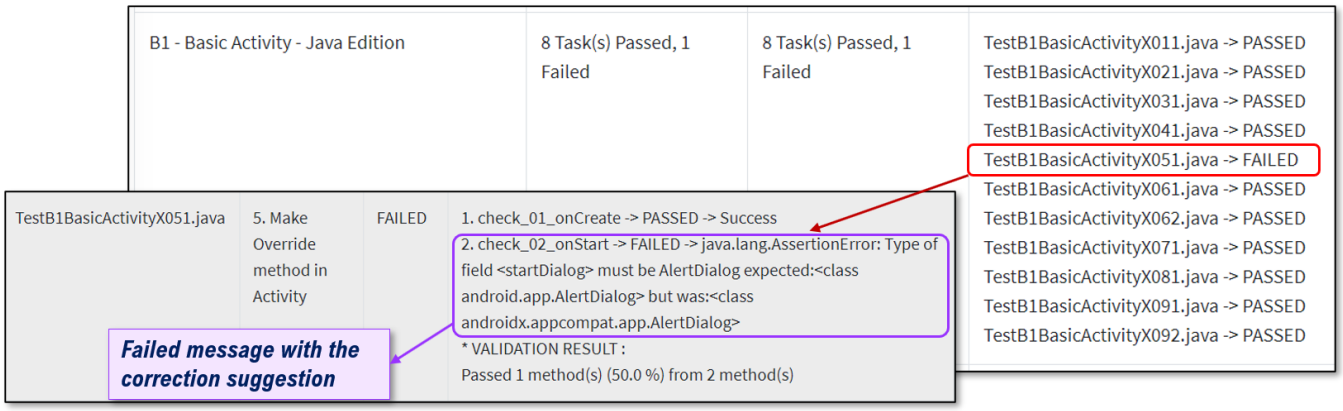

Fig. 11. Web interface for checking the validation results.

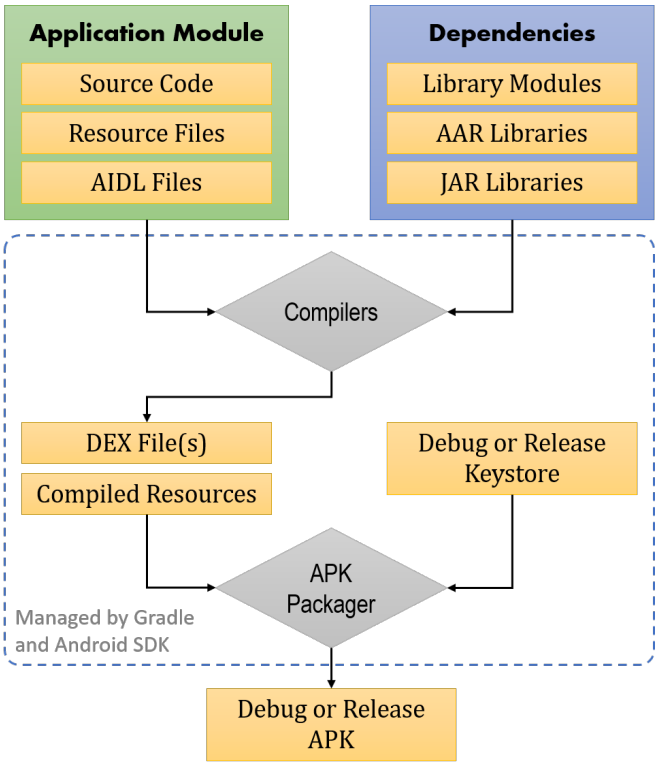

Fig. 12. Building process of Android project.

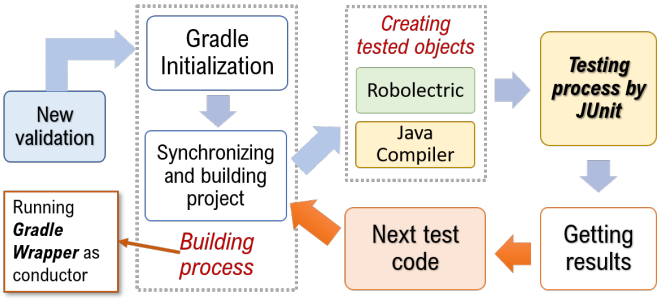

Fig. 13. Unit testing process for validation.

Android project folder to *executor module*.

2) Executor Module: Before the *executor module* runs the test codes from *submission detector* module one by one, it will first check the configuration of Android project. If it is found to be incomplete, this module will abort the validation process and produce an error message for the submission. Otherwise, it will check the server load. If the current server load is found to be too high for validation, the process will halt for one minute and then repeat the checking process. To execute the test codes one by one, the executor module will run Gradle command-line. Figure 15 shows a case of a test code execution with its results that will be sent to *parser module* as plain text.

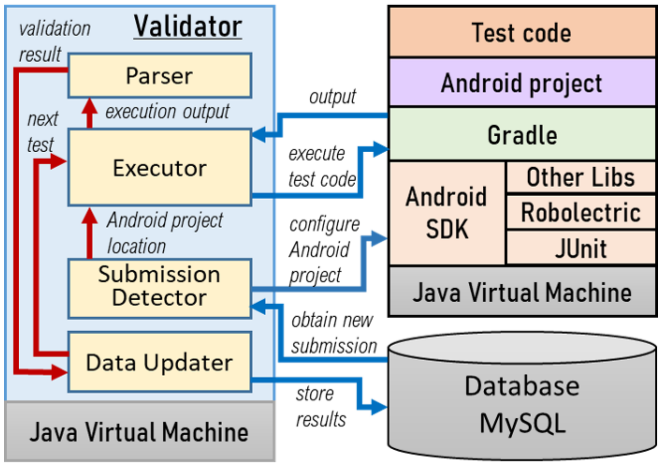

Fig. 14. Software architecture of validator program.

3) Android Project Configuration Check: First, the use of Android application library in the source codes is checked. Currently, two types of libraries released by Google as part of Android SDK are available, namely, *Android Support* library and *AndroidX* library [56]. If library misuses are found in the source codes, the executor module will correct them. Then, the attributes of Java source codes, including package name, import class, and application name, are checked. If incorrectness is found, the executor module will correct them. Any incorrectness will be reported as a part of validation messages.

4) Server Load Check: To compile the Android project, sufficient resources in server are required. Thus, the executor module checks the available memory size and CPU rate. For this purpose, the server administrator must set up the thresholds on them to run the testing process. In general, the available CPU rate should be at least 80% and the memory size is at least 1.5 GB to run a test code by Gradle in command-line.

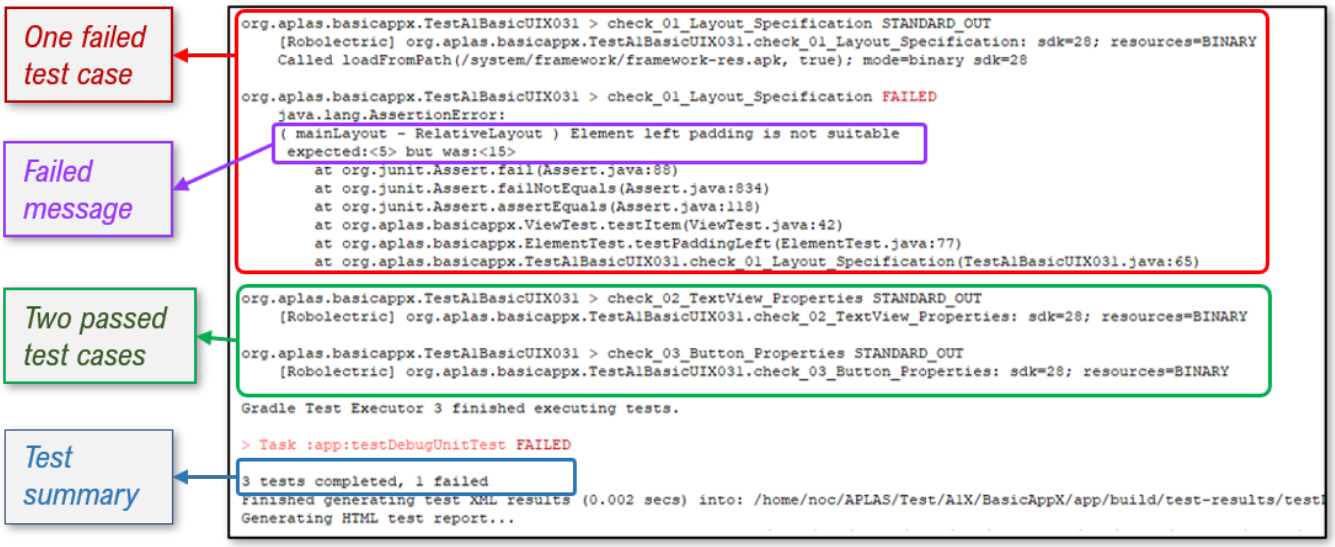

Fig. 15. Results of a test code execution by command-line.

5) Parser Module: The output from the *executor module*, as shown in Figure 15, contains four important information on the validation results including passed test cases, failed test cases, failed test information, and test results summary. The *parser module* extracts the following information in this figure:

- the *red box* designates one failed test case whose method name is *'check_01_Layout_Specification'*,
- the *purple box* shows messages related to failed information produced by JUnit which the attribute of *TextView* with id *'myTextView1'* contains an incorrect height value,
- the *green box* indicates two passed test cases whose method names are *'check_02_TextView_Properties'* and *'check_03_Button_Properties'*,
- the *blue box* summarizes the testing results which contain the test cases number and the failed tests number.

This module will parse the messages to identify each information and send them to the *data updater module*.

6) Data Updater Module: The *data updater module* will convert the validation results for each test code sent from the *parser module* into SQL format to be stored in the MySQL database. They include test results for each test case, messages on failed test cases, a summary of validation results, and duration of the validation process. After that, as explained in Figure 13, *executor module* will run the next test code. This cycle will be continued until all test codes in the Android project was executed. Then, both the student and his/her teacher can access these data of validation results using the web application immediately.

D. Validation Result Classification

For each test, the *TestLogEvent* class in Gradle[57] will output one of six states in Table IV.

Then, they are classified into three statuses in Table V to simplify the validation results. The *Failed* status means that the source code is not correct but can be compiled by Gradle where it has at least one failed test case. The *Passed* status means that there is no failed test case. The *Error* status means that the Android project has an incorrect code and Gradle failed to compile the source codes.

TABLE IV
OUTPUT FROM *TestLogEvent* CLASS.

State	Message
FAILED	The test has failed.
PASSED	The test has completed successfully.
SKIPPED	The test has been skipped.
STANDARD_ERROR	The test has written a message to standard error.
STANDARD_OUT	The test has written a message to standard out.
STARTED	The test has started.

TABLE V
STATUS OF VALIDATION RESULT.

Status	Condition
Failed	The test case has *FAILED* in *Gradle* execution.
Passed	The test case has *PASSED* or *STANDARD_OUT* without *FAILED* in *Gradle* execution.
Error	The test case has other state.

VII. EVALUATION

In this section, we evaluate the online platform for APLAS by conducting preliminary testing and implementing it through applications to 60 undergraduate students who are taking the Android programming course in an IT department in Indonesia.

A. Evaluation Setup

First, we describe the setups of evaluations.

1) Server Specification: Table VI presents the hardware and software specifications of the server in the implemented platform.

2) Learning Topics and Materials: The three learning topics of *Basic UI*, *Basic Activity*, and *Advanced Widgets* were selected for evaluations. For each topic, the learning

TABLE VI
SERVER SPECIFICATIONS.

Item	Specification
Processor	AMD Opteron(TM) Processor 6238 4 Core 2.6 GHz.
RAM	11 GB.
Storage	100 GB.
Operating system	Linux Ubuntu 18.04.4 LTS.
JVM	Java OpenJDK 1.8.
SDK	Android SDK Tools 26.
Web server	Apache server 2.4.29, Laravel 6, PHP 7.4.1.
Domain	aplas.polinema.ac.id

materials using Android Support version and AndroidX version were prepared to let students learn Android programming using different libraries. Table VII shows the technical specifications of the three topics.

TABLE VII
TECHNICAL SPECIFICATIONS OF THREE LEARING TOPICS.

aspects	Basic UI	Basic Activity	Advanced Widgets
# of tasks	9	9	8
# of *JUnit-based* test codes	0	4	0
# of *Robolectric-based* test codes	12	7	9
total # of test codes	12	11	9

To help students solve each topic, a *Youtube* video was provided to explain how to use APLAS for learning Android programming. Besides, APLAS provides extended guide documents were written in the native language to improve readability.

3) Student Solving Activity: The 60 students accessed learning materials from a web browser on their PCs and download the necessary files. Then, they solved each task in three learning topics using Android Studio, where only two days were available for each topic. At the *first chance*, a student could submit the developed source codes after completing all the tasks of a learning topic. If he/she cannot solve some tasks or gets a Failed or Error result, he/she can solve them and submit the answers again one week later, which is called the *second chance*.

B. System Stability

Second, to evaluate the robustness of the online platform, we asked 60 students to solve three learning topics using it at the same time. They submitted a total of 183 answers using web browsers on their PCs. The validation results of these answers in Table VIII were obtained correctly, where it shows the number of passed and failed students at the first and second chances, and the maximum, minimum, and average solving time by each student.

1) Answer Results: As shown in Table VIII, all of the 60 students successfully solved all the tasks in the three learning topics before the due date through the two submission chances. For *Basic UI* and *Basic Activity*, they solved all the tasks at the first chance. For *Advanced Widgets*, only two students failed to solve some tasks at the first chance but passed them at the second chance.

TABLE VIII
STUDENT'S SOLUTION RESULTS.

Topic name	Basic UI	Basic Activity	Advanced Widgets
# of passed / failed students at first chance	60 / 0	60 / 0	58 / 2
# of passed / failed students at second chance	0 / 0	0 / 0	2 / 0
min. / max. solving time (min.)	18 / 275	25 / 422	25 / 290
average solving time (min.)	106.46	174.78	121.47

2) Feedback for Failed Results: The online platform correctly produces the message that describes the reason of the *error* result or the *failed* result at the validation of each task. Figure 16 shows their examples. The error result notes that the validator program cannot find the definitions of 'distList' and 'weightList' in the source codes of the *Basic Activity* topic. The failed result notes that the 'TextView' with the 'clrText' id displays a wrong text in four test cases, 'check_01_Before_Game', 'check_02_Right_Answer', 'check_03_Wrong_Answer', and 'check_04_No_Answer', with their incorrectness for the *Advanced Widgets* topic with *Color Game* application. The student can correct the answer by referring to them.

3) Solving Time: Table VIII shows that the required time to solve each topic is much different among the students. The time becomes shorter than the time using the previous system in [16][17][18].

C. Web Browser Compatibility

Third, we evaluated the web browser compatibility for use of the online platform. At each access of a student to the platform server, the accessing time, the web browser name, the IP address, and the error case were recorded, where a total of 10,304 accesses were observed. Table IX shows the browser names, the access percentage, and the failed access percentage. It shows that *Chrome* was mostly used by the students and the seven browsers caused no error.

TABLE IX
WEB BROWSERS HAD BEEN USED BY TEACHERS AND STUDENTS.

Web Browser	User Access	Error Case
Chrome Desktop	62%	0%
Chrome Mobile	11.55%	0%
Firefox	14.07%	0%
Safari	7.27%	0%
Microsoft Edge	2.62%	0%
Safari Mobile	1.5%	0%
Opera	0.99%	0%

D. Processing Time Reduction

Fourth, we evaluated the reduction of estimated processing time on the three functions by teacher in the online platform by comparing them with the previous methods. Table X compares the estimated processing time required to distribute the learning materials in three learning topics to 60 students, collecting their answers, and validating them by using emails or flash disks in the previous methods and by using this

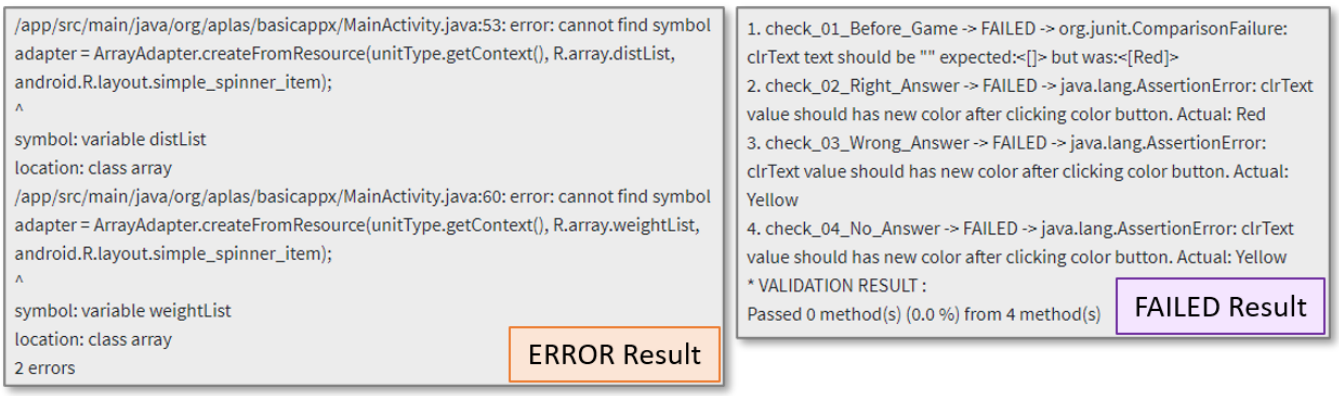

Fig. 16. Error and Failed status of validation result

online platform in this proposal. For the online platform, we counted the waiting time for the teacher to complete each operation after taking an action as the processing time.

TABLE X
PROCESSING TIME COMPARISONS BETWEEN PREVIOUS AND PROPOSAL.

	previous		proposal
media	**email**	**flash disk**	**online platform**
distribution	3 hours	5 hours	1 min.
collection	9 hours	18 hours	0 min.
validation	50 hours		0 min.

1) Improvements in Distribution Process: The processing time for distributing learning materials to students is only 1 minute by using the online platform. On the other hand, it was 3 hours by emails and 5 hours by flash disks. The use of emails caused a lot of loads to teacher in sending learning materials to students and collecting their answers, in addition to the file size limitation problem at file attachments. The use of flash disks made students queue in the classroom to copy their answers one by one to the teacher's PC.

2) Improvements in Collection Process: The processing (waiting) time by teacher for collecting answers from students becomes zero, because the teacher does not need to download them from server. The students' answers will be validated on server automatically. On the other hand, it was 9-18 hours by using emails or flash disks.

3) Improvements in Validation Process: The processing time for validating the answers also becomes zero, because they are automatically validated at the server before the access by the teacher. On the other hand, it was 50 hours by manually validating them on Android Studio.

E. Answer Validation Performance

Fifth, we evaluated the processing time to validate the student's answers on the server of online platform.

1) Test Code Execution Process: At the answer validation, running the test codes with *Gradle* can take long CPU times by synchronizing, integrating, compiling, and running test code in the Android project. As shown in Figure 12, *Application module* and *Dependencies* are compiled together to produce the Java class files [58]. Then, *Robolectric* intercepts the execution process [59] and generates the *Shadow objects* as a simulation of the Android application on JVM. Finally, the unit testing by JUnit is performed by running the test codes.

2) Validation Time of Three Topics: Table XI shows the test codes number, the test cases number in the test codes, the minimum, maximum, and average validation time for each learning topic. The average validation time by the proposal is shorter than the time by the manual one.

TABLE XI
VALIDATION TIME RESULTS.

Topic name	**Basic UI**	**Basic Activity**	**Advanced Widgets**
# of test codes for *JUnit*	0	4	0
# of test codes for *Robolectric*	11	7	9
# of test cases	35	12	39
min / max validation time (sec.)	319/1346	182/1217	243/1298
average validation time (sec.)	434.78	293.37	310.92

3) Waiting Time in Getting Validation Results: Figure 17 shows the distribution of the waiting time to obtain the validation results after submitting the answers among the 183 submissions. To measure the waiting time for each submission, we calculated the difference between submitting time and time on getting validation results. The results of waiting time are relatively long due to the complexity of testing process of Android application, as illustrated in Figure 13. In each testing process, Gradle will synchronize, integrate, compile, and execute the test code in the Android project [60].

Figure 17 indicates that the waiting time is distributed unevenly with 374.01 seconds for the average and 244.77 seconds for the standard deviation. 90% of the submissions spent less than 580 seconds and 10% of them did the time from 590 seconds to 1368 seconds. Using the *k-means* method, they can be clustered into three groups in Table XII.

4) Analysis on Waiting Time Results: To understand the behavior of validation process, based on results in Table XII, we investigated the detail of submission data in each cluster. From the submission data, some attributes could be generated which comprise the submission time, the time difference with previous submission, the learning topic of previous submission, CPU load, and RAM usage during the validation

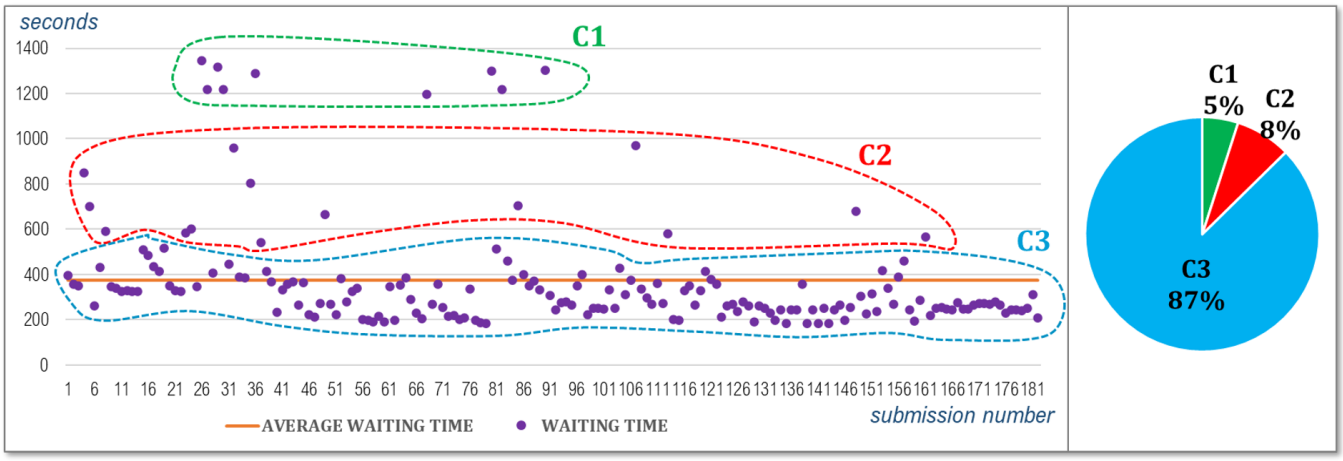

Fig. 17. Distribution of validation time of 183 answers from 60 students (seconds)

TABLE XII
CLUSTERING RESULTS OF WAITING TIME DATA USING *k-means*.

No	Range	# of Submission.	Ave. Time
C1	1190-1368 sec.	9 (5%)	1287.91
C2	573-986 sec.	14 (8%)	752.32
C3	181-557 sec.	160 (87%)	365.19

process. As shown in Figure 18, several characteristics of each cluster can be obtained as follow:

- **C1** represents the cluster where *Gradle Wrapper* daemon was initiated running for the validation, because it was terminated from the long idling due to no answer submissions from students for more than five hours. The initiation of *Gradle Wrapper* daemon caused the long waiting time.
- **C2** represents the cluster where *Gradle Wrapper* daemon continued running but new shadow objects were generated for the validation.
- **C3** represents the cluster where *Gradle Wrapper* daemon continued running and no new shadow objects were generated, which made the short time.

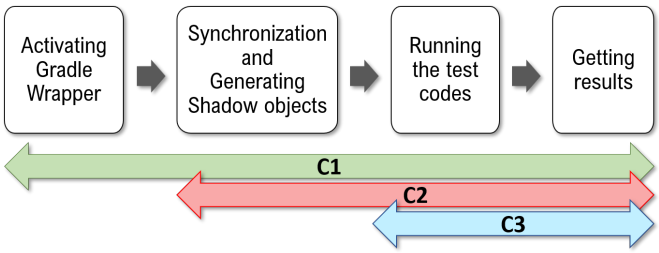

Fig. 18. Four steps in validation process

F. System Security

Sixth, we evaluated the system security in three aspects, including system protection, information authorization, and data protection, as shown in Table XIII.

1) System Protection: The implemented online platform allows the public access to the Internet using the HTTP protocol as the web application system. To avoid access from unauthorized persons or *bot* programs[61], it implemented

TABLE XIII
FEATURES TO IMPLEMENT SYSTEM SECURITY.

security aspect	features in charge
system protection	- user authentication, - *CAPTCHA* tool, - registration verification.
information authorization	- web page authorization, - data authorization
data protection	- internal server access - secured web access - access rights protection

the *user authentication* function with the 'Login' page as the gate to the system [62]. This login page adopts *CAPTCHA* [63] to differentiate the real users and the automated bot users. All the new students and teachers who want to use the system need to complete the registration using 'Registration' page, which also uses *captcha*. At the registration, the *registration verification*[64] is applied to avoid registering any unknown user in the registration process. The teacher's registration must be verified by administrator and the student's registration must be verified by his/her teacher.

2) Information Authorization: The online platform of APLAS has three user types of the student, the teacher, and the administrator. Each of them is allowed to take the defined role in the system. The *user authorization* is applied to avoid access to each web page from any unauthorized user. The *data authorization* is also applied to avoid access to the specific data from any unauthorized user. For instance, a student can submit his/her answers and check their validation results specifically, instead of his/her friends' results.

3) Data Protection: In the online platform, the database system must be secured at the public access from the Internet [65]. The database system is set up to prohibit external access. It permits access from application programs in the server system. To update and maintain the database system, the *secured web protocol* is adopted for exclusive access. Each program in the online platform has its own right to access the database to protect it from unauthorized access.

G. User's Opinions

Last, we evaluated students' opinions on the online platform. The students were asked to give opinions on the uti-

lization of the online platform and the learning process using it for all learning topics. Table XIV shows the percentages of positive, neutral, and negative opinions from them. A lot of positive opinions said that this online platform is helpful and useful for self-learning, especially for developing Android applications. Negative opinions occupied lower than 10% in any topic. Thus, the online platform is helpful for students in learning Android programming.

TABLE XIV
USER'S OPINIONS ON THE THREE TOPICS.

learning topic	positive	neutral	negative
Basic UI	55.4%	40.2%	4.5%
Basic Activity	59.8%	31.5%	8.7%
Advanced Widgets	54.7%	37.9%	7.4%

VIII. CONCLUSION

This paper presented the proposal, implementation, and comprehensive evaluation of the online platform of APLAS for distributing the learning materials, collecting and validating the student's answers, based on the web application system. Using the validator program on the server, the test codes can be run automatically for validation when a new answer is submitted from a student. By storing data in the database system, both teachers and students can access the validation results on the web browser. The evaluation is conducted by applying this online platform to 60 students in an Indonesian university. The evaluation results confirmed the correctness of implemented functions in the web application. The results also confirmed the stability, compatibility, effectiveness, performance, robustness, accessibility, security, and usefulness of this online platform. It could reduce the teacher's load and eliminate the main problem in learning process with APLAS. For future works, the research will continue to implement other answer submission methods, develop the automatic grading function of the student's answers, develop an interactive Android UI learning feature, and provide various interfaces to help teachers in using APLAS.

ACKNOWLEDGMENT

The authors say a great appreciation to Mr. Ahmadi Yuli Ananta, Mr. Rudy Ariyanto, and Mr. Faizal Abroni from State Polytechnic of Malang for supporting us in implementing our web-based online platform by providing an online server. Also, great thanks to Ms. Permata and Ms. Farida for proofreading the paper. We are very grateful to have students in State Polytechnic of Malang with great cooperation to use this platform.

REFERENCES

[1] D. Gladden, *The effects of smartphones on social lives: how they affect our social interactions and attitudes*, OTS Master's Level Projects & Paper, Old Dominion University, 2018.

[2] K. Hsiao, "Android smartphone adoption and intention to pay for mobile internet: perspectives from software, hardware, design, and value," Library Hi Tech, vol. 31, no. 2, pp216-235, 2013.

[3] Statista GmbH. Forecast number of mobile devices worldwide from 2019 to 2023 (in billions)* [Online]. Available: *https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/.*

[4] Statcounter Global Stats. Mobile operating system market share worldwide, June 2019 - June 2020 [Online]. Available: *https://gs.statcounter.com/os-market-share/mobile/worldwide*.

[5] J. Clement. Google Play: number of available apps 2009-2020 [Online]. Available: *https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.*

[6] D. K. Kim, "Towards performance-enhancing programming for Android application development," International Journal of Contents, vol. 13, no. 4, pp39-46, 2017.

[7] T. Wu, "Understanding programming language semantics for the sophisticated world," IAENG International Journal of Computer Science, vol. 36, no. 1, pp53-60, 2009.

[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," International Journal of Web Information Systems, vol. 16, no. 1, pp115-135, 2019.

[9] A. Tort, A. Olivé, and M. -R. Sancho, "An approach to test-driven development of conceptual schemas," Data & Knowledge Engineering, vol. 70, no. 12, pp1088-1111, 2011.

[10] W. Bissi, A. G. S. S. Neto, and M. C. F. P. Emer, "The effects of test driven development on internal quality, external quality and productivity: a systematic review," Information and Software Technology, vol. 74, no. 1, pp45-54, 2016.

[11] JUnit. JUnit: a simple framework to write repeatable tests [Online]. Available: *https://junit.org/junit4/.*

[12] Robolectric. Robolectric: a framework that brings fast and reliable unit tests to Android [Online]. Available: *http://robolectric.org/.*

[13] Y. W. Syaifudin, N. Funabiki, M. Mentari, P. Y. Saputra, Y. Yunhasnawa, F. Ulfa, and M. Kuribayashi, "Web application implementation of Android programming learning assistance system and its evaluations," in 3rd Annual Advanced Technology, Applied Science, and Engineering Conference, Malang, Indonesia, 2020.

[14] K. Marsicano, B. Gardner, B. Phillips, and C. Stewart, *Android programming: the big nerd ranch guide*, Big Nerd Ranch Guides, 2019.

[15] A. Sarkar, A. Goyal, D. Hicks, D. Sarkar, and S. Hazra, "Android application development: a brief overview of Android platforms and evolution of security systems," in 3rd International conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud), pp73-79, Palladam, India, 2019.

[16] Y. W. Syaifudin, N. Funabiki, and M. Kuribayashi, "Learning model for Android programming learning assistant system," in IEICE General Conference, ppS89-S90, Tokyo, Japan, 2019.

[17] Y. W. Syaifudin, N. Funabiki, and M. Kuribayashi, "An implementation and evaluation of basic activity topic for interactive application stage in Android programming learning assistance system," in Forum of Information Technology, pp305-306, Okayama, Japan, 2019.

[18] Y. W. Syaifudin, N. Funabiki, and M. Kuribayashi, "An implementation and evaluation of advanced widgets topic for interactive application stage in Android programming learning assistance system," in 8th International Conference on Information and Education Technology, pp88–93, Okayama, Japan, 2020.

[19] C. A. Usener, T. A. Majchrzak, and H. Kuchen, "E-assessment and software testing," Interactive Technology and Smart Education, vol. 9, no. 1, pp45-56, 2012.

[20] N. Funabiki, Y. Matsushima, T. Nakanishi, K. Watanabe, and N. Amano, "A Java programming learning assistant system using test-driven development method," IAENG International Journal of Computer Science, vol. 40, no. 1, pp38-46, 2013.

[21] N. Funabiki, Tana, K. K. Zaw, N. Ishihara, and W. -C. Kao, "A graph-based blank element selection algorithm for fill-in-blank problems in Java programming learning assistant system," IAENG International Journal of Computer Science, vol. 44, no. 2, pp247-260, 2017.

[22] S. V. Yulianto and I. Liem, "Automatic grader for programming assignment using source code analyzer," in International Conference on Data and Software Engineering, pp. 1-4, Bandung, Indonesia, 2014.

[23] T. Staubitz, H. Klement, R. Teusner, J. Renz, and C. Meinel, "CodeOcean - a versatile platform for practical programming exercises in online environments," in IEEE Global Engineering Education Conference, pp314-323, Abu Dhabi, UAE, 2016.

[24] 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, pp1-21, 2020.

[25] 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, pp1174-1184, 2010.

[26] E. Daehnhardt and Y. Jing, "An approach to software selection using semantic Web," IAENG International Journal of Computer Science, vol. 40, no. 4, pp238-249, 2013.

[27] I. Bosnić, B. Mihaljević, M. Orlić, and M. Žagar, "Source code validation and plagiarism detection - technology-rich course experiences,"

in 4th International Conference on Computer Supported Education, vol. 1, pp149-154, Porto, Portugal, 2012.
[28] 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 IEEE 14th International Conference on Advanced Learning Technologies, pp637-639, Athens, Greece, 2014.
[29] Y. Hayashi, K. -I. Fukamachi, and H. Komatsugawa, "Collaborative learning in computer programming courses that adopted the flipped classroom," in International Conference on Learning and Teaching in Computing and Engineering, pp209-212, Taipei, Taiwan, 2015.
[30] C. Hundt, M. Schlarb, and B. Schmidt, "SAUCE: A web application for interactive teaching and learning of parallel programming,"Journal of Parallel and Distributed Computing, vol. 105, pp163-173, 2017.
[31] J. -M. Su and S. -J. Wang, "A web-based learning activity integrated with scratch tool to support programming learning," in 10th International Conference on Ubi-media Computing and Workshops, pp1-4, Pattaya, Thailand, 2017.
[32] J. -M. Su and F. -Y. Hsu, "Building a visualized learning tool to facilitate the concept learning of object-oriented programming," in 6th IIAI International Congress on Advanced Applied Informatics, pp516-520, Hamamatsu, Japan, 2017.
[33] H. Kang and J. Cho, "Case study on efficient android programming education using multi android development tools," Indian Journal of Science and Technology, vol. 8, no. 19, pp1-5, 2015.
[34] H. A. A. Rekhawi and S. S. A. Naser, "Android applications UI development intelligent tutoring system," International Journal of Engineering and Information Systems, vol. 2, no. 1, pp1-14, 2018.
[35] D. Amalfitano, V. Riccio, N. Amatucci, V. D. Simone, and A. R. Fasolino, "Combining automated GUI exploration of Android apps with capture and replay through machine learning," Information and Software Technology, vol. 105, pp95-116, 2019.
[36] J. Zhao, T. Song, and Y. Sun, "APIHelper: helping junior Android programmers learn API usage," IAENG International Journal of Computer Science, vol. 47, no. 1, pp92-97, 2020.
[37] H. K. Kim, "Test-driven mobile applications development," in World Congress on Engineering and Computer Science, San Fransisco, vol. 2, pp785-789, USA, 2013.
[38] M. Aljamea and M. Alkandari, "MMVMi: a validation model for MVC and MVVM design patterns in iOS applications," IAENG International Journal of Computer Science, vol. 45, no. 3, pp377-389, 2018.
[39] M. Wahid and A. Almalaise, "JUnit framework: an interactive approach for basic unit testing learning in software engineering," in 3rd International Congress on Engineering Education, pp159-164, Kuala Lumpur, Malaysia, 2011.
[40] M. Linares-Vásquez, C. Bernal-Cardenas, K. Moran, and D. Poshyvanyk, "How do developers test android applications?," in IEEE International Conference on Software Maintenance and Evolution, pp613-622, Shanghai, China, 2017.
[41] Google Developers. (2020, Jun 15). SDK Tools release notes [Online]. Available: *https://developer.android.com/studio/releases/sdk-tools*.
[42] M. Beller, G. Gousios, A. Panichella, S. Proksch, S. Amann, and A. Zaidman, "Developer testing in the IDE: patterns, beliefs, and behavior", IEEE Transactions on Software Engineering, vol. 45, no. 3, pp261-284, 2019.
[43] L. Lun, X. Chi, and H. Xu, "Testing approach of component interaction for software architecture," IAENG International Journal of Computer Science, vol. 45, no. 2, pp353-363, 2018.
[44] D. R. Almeida, P. D. L. Machado, and W. L. Andrade, "Testing tools for Android context-aware applications: a systematic mapping," Journal of the Brazilian Computer Society, vol. 25, no. 12, pp1-22, 2019.
[45] G. K. Mostefaoui and F. Tariq, *Mobile apps engineering: design, development, security, and testing*, Boca Raton: CRC Press, 2019.
[46] P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyandé, and J. Klein, "Automated testing of Android apps: a systematic literature review," IEEE Transactions on Reliability, vol. 68, no. 1, pp45-66, 2019.
[47] 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, vol. 1, no. 4, pp926-941, 2011.
[48] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts*, 9th ed., New York: Wiley Pub., 2013.
[49] Gradle Inc., Gradle user manual version 6.7, Gradle Inc. [Online]. Available: *https://docs.gradle.org/6.7/release-notes.html*.
[50] Google Developers. Configure your build [Online]. Available: *https://developer.android.com/studio/build*.
[51] Google Developers. SDK Build Tools release notes [Online]. Available: *https://developer.android.com/studio/releases/build-tools*.
[52] Google Developers. AAPT2 [Online]. Available: *https://developer.android.com/studio/command-line/aapt2*.
[53] Google Developers. d8 [Online]. Available: *https://developer.android.com/studio/command-line/d8*.
[54] P. Gilski and J. Stefanski, "Android OS: a review," TEM Journal, vol. 4, no. 1, pp116-120, 2015.
[55] R. Meier and I. Lake, *Professional Android*, 4th ed., Indianapolis: John Wiley & Sons, 2018.
[56] Google Developers. AndroidX overview [Online]. Available: *https://developer.android.com/jetpack/androidx*.
[57] Gradle Inc. Enum TestLogEvent [Online]. Available: *https://docs.gradle.org/current/javadoc/org/gradle/api/tasks/testing/logging/TestLogEvent.html*.
[58] Y. Zhaoa, Z. Tanga, G. Yea, D. Penga, D. Fanga, X. Chena, and Z. Wang, "Compile-time code virtualization for android applications," Computers & Security, vol. 94, pp1-16, 2020.
[59] E. R. Wognsen, H. S. Karlsen, M. C. Olesen, and R. R. Hansen, "Formalisation and analysis of Dalvik bytecode," Science of Computer Programming, vol. 92, no. A, pp25-55, 2014.
[60] G. Maudoux and K. Mens, "Correct, efficient, and tailored: the future of build systems," IEEE Software, vol. 35, no. 2, pp32-37, 2018.
[61] S. Vaithyasubramanian, D. Lalitha, and C. K. Kirubhashankar, "Enhancing website security against bots, spam and web attacks using /CAPTCHA," International Journal of Computers and Applications, vol. 44, no. 1, pp1-7, 2019.
[62] M. Li, W. Susilo, and J. Tonien, "The code for securing web applications," Journal of Information and Optimization Sciences, vol. 40, no. 4, pp905-917, 2019.
[63] N. Roshanbin and J. Miller, "A survey and analysis of current CAPTCHA approaches," Journal of Web Engineering, vol. 12, no. 1-2, pp1-40, 2013.
[64] Z. Wang and Y. Sun, "How to design the registration and login function of APP," Journal of Software Engineering and Applications, vol.11, no. 5, pp223-234, 2018.
[65] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," Journal of Computer and System Sciences, vol. 80, no. 5, pp973-993, 2014.

Yan Watequlis Syaifudin received the B.S. degree in Informatics from Bandung Institute of Technology, Indonesia, in 2003, and the M.S. degree in Information Technology from Sepuluh Nopember Institute of Technology, Surabaya, Indonesia, in 2011, respectively. In 2005, he joined State Polytechnic of Malang, Indonesia, as a lecturer. He is currently a Ph.D. candidate in Graduate School of Natural Science and Technology at Okayama University, Japan. His research interests include educational technology, multimedia processing, and database systems. He is a student member of IEICE.

Nobuo Funabiki received the B.S. and Ph.D. degrees in mathematical engineering and information physics from the University of Tokyo, Japan, in 1984 and 1993, respectively. He received the M.S. degree in electrical engineering from Case Western Reserve University, USA, in 1991. From 1984 to 1994, he was with the System Engineering Division, Sumitomo Metal Industries, Ltd., Japan. In 1994, he joined the Department of Information and Computer Sciences at Osaka University, Japan, as an assistant professor, and became an associate professor in 1995. He stayed at University of Illinois, Urbana-Champaign, in 1998, and at University of California, Santa Barbara, in 2000-2001, as a visiting researcher. In 2001, he moved to the Department of Communication Network Engineering (currently, Department of Electrical and Communication Engineering) at Okayama University as a professor. His research interests include computer networks, optimization algorithms, educational technology, and Web technology. He is a member of IEEE, IEICE, and IPSJ.

Mustika Mentari received the B.S. degree in Computer Science from Brawijaya University in 2011, and the M.S. degree in Informatics from Sepuluh Nopember Institute of Technology, Surabaya, Indonesia in 2014, respectively. Since 2015, she has been a lecturer at State Polytechnic of Malang. Her research interests include image processing, intelligent systems, and computer vision.

Habibie Ed Dien received the B.S. degree in Informatics Engineering from Trunojoyo Madura University in 2014, and the M.S. degree in Informatics from Bandung Institute of Technology, Indonesia, in 2018, respectively. In 2017, he worked at the Agency National Disaster Management (BNPB) as an ICT support personnel. Since 2019, he has been a lecturer at State Polytechnic of Malang. His research interests include media technology and mobile.

Ikhlaashul Mu'aasyiqiin is currently pursuing the B.S. degree in Computer Science at State Polytechnic of Malang, Indonesia, and at Shenyang Aerospace University, China, through the double degree program. His research interests include educational technology.

Minoru Kuribayashi received the B.E., M.E., and D.E degrees from Kobe University, Kobe, Japan, in 1999, 2001, and 2004, respectively. From 2002 to 2007, he was a Research Associate in the Department of Electrical and Electronic Engineering, Kobe University. In 2007, he was appointed as an Assistant Professor at the Division of Electrical and Electronic Engineering, Kobe University. Since 2015, he has been an Associate Professor in the Graduate School of Natural Science and Technology, Okayama University. His research interests include digital watermarking, information security, cryptography, and coding theory. He received the Young Professionals Award from IEEE, Kansai Section, in 2014.

Wen-Chung Kao received the M.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taiwan, in 1992 and 1996, respectively. From 1996 to 2000, he was a Department Manager at SoC Technology Center, ERSO, ITRI, Taiwan. From 2000 to 2004, he was an Assistant Vice President at NuCam Corporation in Foxlink Group, Taiwan, where he was responsible for leading embedded software team to develop digital still/video cameras. In 2002, he was also invited to form SiPix Technology Inc., Taipei, Taiwan, where he was in charge of setting up the research team of the company and studying flexible electrophoretic display. Since 2004, he has been with National Taiwan Normal University (NTNU), Taipei, Taiwan, where he is currently the Research Chair Professor at Department of Electrical Engineering and the Dean of College of Technology and Engineering. His current research interests include system-on-a-chip (SoC) as well as embedded software design, flexible electrophoretic display, machine vision system, digital camera system, and color imaging science.