Three Implementations for Improvements of Android and Flutter Programming Learning Assistance System

March, 2024

Abdul Rahman Patta

Graduate School of
Natural Science and Technology

(Doctor's Course)
Okayama University

Dissertation submitted to
Graduate School of Natural Science and Technology
of
Okayama University
for
partial fulfillment of the requirements
for the degree of
Doctor of Philosophy.

Written under the supervision of

Professor Nobuo Funabiki

and co-supervised by
Professor Satoshi Denno
and
Professor Yasuyuki Nogami

Okayama University, March 2024.

To Whom It May Concern

We hereby certify that this is a typical copy of the original doctor thesis of
Abdul Rahman Patta

Signature of	Seal of
the Supervisor	
	Graduate School of
Prof. Nobuo Funabiki	Natural Science and Technology

Abstract

Nowadays, numerous studies are dedicated to programming learning assistant systems due to the pivotal role that basic computer literacy and programming education hold in computer science education. Additionally, the curriculum guidelines endorsed by the *Association for Computing Machinery (ACM)* emphasize the integration of mobile programming-related topics into computer science education. Therefore, the significance of educating novice students in mobile application development has grown substantially in universities and professional schools.

To enhance the education of mobile programming, particularly in Android programming, the *Android programming learning assistance system* called APLAS has been developed as a web-based application as a self-study tool. APLAS assumes the use of Android Studio as the *Integrated Development Environment (IDE)* for the implementation of Android application programs. The code marking process using APLAS follows the *test-driven development (TDD)* method. Specifically, *JUnit* and *Robolectric* are employed for unit testing of student answer codes. *JUnit* assesses logic functions, while *Robolectric* evaluates user interfaces. Students have the capability to verify their answers by executing the provided test code within *Android Studio*.

APLAS provides guidance documents, supplementary files, and test codes to guide students in the learning process. The *guide document* includes step-by-step instructions, covering the creation of a new project application, configuration, writing code for Android components, tutorial on accomplishing the task, and guidance on validating the developed source codes as answers to the assignment. The *test codes* are used to validate the student's answer codes. The *supplement files* cover the additional required files to complete the task, such as fonts, images, videos, animations, and style resources.

However, there are three drawbacks to be addressed in APLAS. First, in practical programming, students often need to write source code or build projects without the assistance of a guide document. Instead, they should rely on an application specification document. Second, 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 at the early stage of the course, a function that can monitor the progress on *Android Studio* is required to identify such students. Third, *Flutter* has become popular instead of *Android* as the uniform platform for different environments including Android, iOS, and browsers. The programming on *Flutter* should be mastered by students.

In this thesis, as the first contribution, I investigate learning outcomes in APLAS using assignments without *guidance documents*. The results confirm the effectiveness of learning with APLAS without them, demonstrating that acquiring Android programming skills through APLAS is particularly beneficial for beginner students.

As the second contribution of the thesis, I implement of the *solution progress monitoring function* in APLAS. This function operates that the student's PC sends the test log file to the APLAS

server, each time the student executes the test code to evaluate the answer code in *Android Studio*. Subsequently, the web interface displays the number of completed assignments, the time spent on them, and the number of validations submitted by each student. Through the utilization of this function, teachers can identify and support the students facing difficulties in completing assignments in APLAS. Real-time data facilitates timely teacher interventions and provides supports for students in need of additional assistances.

As the third contribution of this thesis, I implement the *Grammar Concept Understanding Problem (GUP)* for *Flutter* cross-platform mobile programming learning. With the advent of cross-platform mobile programming, a paradigm shift has occurred in the mobile development landscape. Therefore, it is important for students to learn cross-platform application development. *Flutter* is gaining popularity as a software development framework for creating cross-platform applications compatible with *Android* and *iOS*. Based on our previous study of *Programming Learning Assistance Systems (PLAS)*, *Grammar-concept Understanding Problems (GUP)* is adopted as initial learning tasks for novice students in cross-platform mobile programming, with a special focus on *Flutter*.

In future works, we aim to integrate the problem model from the *programming learning assistant system* into cross-platform mobile programming learning, with a particular focus on *Flutter*. This involves exploring the creation of *Code Modification Questions (CMP), Code Writing Problems (CWP)*, and the utilization of test code for validating program code. Additionally, we plan to develop a completion activity monitoring function in *Flutter*.

Acknowledgements

I would like to express my gratitude to God and the individuals who supported my Ph.D. studies at Okayama University, Japan. Completing this thesis would not have been possible without their generous support, encouragement, and efforts. I am deeply indebted to them. Although I wish to convey many things to them, words often fail to express my feelings adequately. This is my sincere expression of immense gratitude.

First and foremost, I extend my sincere thanks to my supervisor, Professor Nobuo Funabiki, for his continuous support throughout my Ph.D. studies and research. I appreciate his encouragement, patience, motivation, enthusiasm, and vast knowledge. Professor Nobuo Funabiki guided me in various aspects, including directing research activities, writing exceptional papers, and creating outstanding presentations. His countless valuable suggestions and advice have contributed to the significant achievements in this study and will undoubtedly lead to more accomplishments in my future endeavours.

I am also grateful to my two co-supervisors, Professor Satoshi Denno and Professor Yasuyuki Nogami, for their unwavering support, encouragement, suggestions, and proofreading of this thesis. I extend my sincere thanks for their lessons and enlightening knowledge, as well as to Associate Professor Minoru Kuribayashi and the course teachers at Okayama University during my Ph.D. program.

I want to express my appreciation to the members of the Funabiki Lab at Okayama University. Ms. Keiko Kawabata, Dr. Yan Watequlis Syaifudin, Dr. Hendy Briantoro Dr. Pradini Puspitaningayu, Dr. Hein Hteit, Dr. Md. Mahbubur Rahman, and Dr. Roy Sujan Chandra who helped me a lot, especially in the beginning of my study; Mr. Yohanes Panduman, Ms. Irin Anggraini, and all lab members in general who have shared strength during our three years; Sharing time with these individuals in Okayama has provided me with great experiences and unforgettable moments. Thank you for your significant support and helpfulness in both my academic endeavours and daily life.

I also extend my gratitude to my colleagues at the State University of Makassar, Indonesia has been a continuous source of support throughout my study. Thank you for your invaluable assistance during this period.

Finally, my deepest gratitude goes to my mother, father, wife, daughter, son, sister, parents-in-law, and all my family members. Their support in various ways over the years has been instrumental. They have consistently provided me with the energy and motivation to complete this study on time. Being far away from my family in Indonesia presented a significant challenge, and their unwavering support helped me overcome it. I am truly blessed to have each and every one of them in my life.

Abdul Rahman Patta
Okayama, Japan
March 2024

List of Publications

Journal Papers

1. **Abdul Rahman Patta**, Nobuo Funabiki, Minoru Kuribayashi, and Yan Watequlis Syaifudin, "An Implementation of Solution Progress Monitoring Function in Android Programming Learning Assistance System," International Journal of Information and Education Technology (IJIET), Vol. 13, No.10, pp. 1597-1603 (October 2023).

International Conference Papers

2. **Abdul Rahman Patta**, Nobuo Funabiki, Yan Watequlis Syaifudin, and Wen-Chung Kao, "An Investigation of Learning Outcomes Using Assignment without Guide Documents in Android Programming Learning Assistance System," 2022 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW 2022), pp. 195-196 (Online, Taipei, Taiwan, 2022).

3. **Abdul Rahman Patta**, Nobuo Funabiki, Yan Watequlis Syaifudin, and Wen-Chung Kao, "An Implementation of Solving Activity Monitoring Function in Android Programming Learning Assistance System," 5th International Conference on Vocational Education and Electrical Engineering (ICVEE 2022), pp. 84-88 (Online, Surabaya, Indonesia, 2022).

4. **Abdul Rahman Patta**, Nobuo Funabiki, Xiqin Lu, and Yan Watequlis Syaifudin, "A Study of Grammar-concept Understanding Problem for Flutter Cross-platform Mobile Programming Learning," 6th International Conference on Vocational Education and Electrical Engineering (ICVEE 2023), pp. 249-254 (Online, Surabaya, Indonesia, 2023).

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Background

Nowadays, mobile devices such as cellular phones, smartphones, and tablets has experienced a substantial surge in popularity over the years. Concurrently, the proliferation of applications tailored for these devices has also seen a notable increase [1]. In computer science education, the study of mobile application development holds significant importance. Numerous studies are dedicated to programming learning assistant systems due to the pivotal role that basic computer literacy and programming education hold in computer science education[2]. Additionally, the curriculum guidelines endorsed by the *Association for Computing Machinery (ACM)* emphasize the integration of mobile programming-related topics into computer science education [3]. Therefore, the significance of educating novice students in mobile application development has grown substantially in universities and professional schools.

To enhance the education of mobile programming, particularly in Android programming, the *Android programming learning assistance system* called APLAS has been developed as a web-based application as a self-study tool [4]. APLAS assumes the use of Android Studio as the *Integrated Development Environment (IDE)* for the implementation of Android application programs. APLAS follows the *test-driven development (TDD)* method[5], where *JUnit*[6] and *Robolectric*[7] are employed for unit testing of student answer codes. *JUnit* assesses logic functions, while *Robolectric* evaluates user interfaces. Students can verify their answer codes by executing the provided test code within *Android Studio* [8].

APLAS provides *guidance documents, supplementary files, and test codes* to aid students in the learning process[9]. The *guide document* includes step-by-step instructions, covering the creation of a new project application, configuration, and writing code for Android components, a tutorial on accomplishing the task, and guidance on validating the developed source codes as answers to the assignment. The *test codes* are used to validate the student's answer codes. The *supplement files* cover the additional required files to complete the task, such as fonts, images, videos, animations, and style resources.

However, there are three drawbacks to be addressed in APLAS. First, in practical programming, students often need to write source code or build projects without the assistance of a guide document. Instead, they should rely on an application specification document. Second, 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 at the early stage of the course, a function that can monitor the progress on *Android Studio* is required to identify such students. Third, *Flutter* has become popular instead of *Android* as the uniform platform for different environments including Android, iOS, and browsers. The programming on *Flutter* should be mastered by students.

1.2 Contributions

This thesis presents the three implementations for the improvement of Android and Flutter programming learning assistance systems.

1.2.1 Investigation of Learning Outcomes in APLAS

The first contribution, I investigate learning outcomes in APLAS using assignments without *guidance documents* [10]. The results confirm the effectiveness of learning APLAS for this task, demonstrating that acquiring Android programming skills through APLAS is particularly beneficial for beginner students.

1.2.2 Implementation of Solution Progress Monitoring Function in APLAS

The second contribution of the thesis, I implement of *solution progress monitoring function* in APLAS[11][12]. This functionality operates by having the program on the student's PC send a test log file to the APLAS server each time the student executes test code to evaluate the answer code in *Android Studio*. Subsequently, the web interface displays crucial metrics, including the number of completed assignments, the time spent on them, and the number of validations submitted by each student. Through the utilization of this innovative function, teachers gain the ability to identify and support students facing challenges in completing assignments in APLAS. Real-time data facilitates timely teacher intervention and provides support for students in need of additional assistance.

1.2.3 Implementation Grammar-concept Understanding Problem for Flutter

The third contribution of this thesis, I implement the *Grammar Concept Understanding Problem (GUP)* for *Flutter* cross-platform mobile programming learning[13]. With the advent of cross-platform mobile programming, a paradigm shift has occurred in the mobile development landscape. Therefore, it is important for students to learn cross-platform application development. *Flutter* is gaining popularity as a software development kit for creating cross-platform applications compatible with *Android* and *iOS*. Based on our previous study of *Programming Learning Assistance Systems (PLAS)*[14], *Concept-Grammar Understanding Problems (GUP)*[15][16] can be adopted as initial learning tasks for novice students in cross-platform mobile programming, with a special focus on *Flutter*.

1.3 Contents of This Dissertation

The remaining part of this thesis is organized as follows. Chapter 2 reviews of Android programming learning assistance system Chapter 3 presents the investigations for the learning outcomes using assignments without documents in *APLAS*. Chapter 4 presents the solution progress monitoring function in *APLAS*. Chapter 5 presents the grammar-concept understating problem for flutter in mobile programming. Chapter 6 reviews relevant works in literature. Finally, Chapter 7 concludes this thesis with some future works.

Chapter 2

Review of Android Programming Learning Assistance System (APLAS)

This chapter reviews the *Android programming learning assistance system (APLAS).* APLAS is a programming learning platform designed to facilitate self-directed learning and provide instruction in Java-based Android programming[4].

2.1 APLAS Overview

APLAS is designed and implemented to provide a self-learning environment for Android programming. The system automatically validates students' answers for assignments using unit testing, following the *test-driven development (TDD)* method. By autonomously assessing students' answers, APLAS enables independent study of Android programming, reducing the need for continuous assistance from a teacher. 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 [17]. In *APLAS*, *JUnit* [6] and *Robolectric* [7] are actually used for *unit testing* of the student answer codes [18]. *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.

2.2 Software Architecture

APLAS consists of two main sides [19]: the client side and the server side, as depicted in Figure 2.1. The client side includes the Learning Platform and the Web Client, while the server side comprises the Validator and the Web Application. Their roles are described as follows[20]:

1. *Learning Platform* is used by the students to complete the 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.

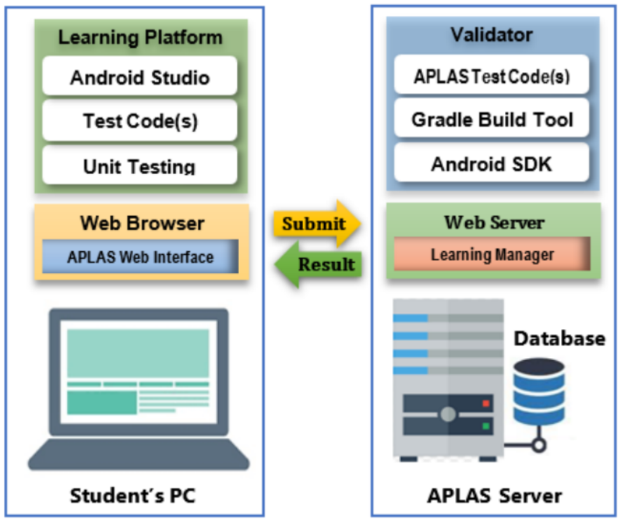


Figure 2.1: Software architecture of *APLAS*.

2.3 Learning Platform

The APLAS provides the interfaces on the web browser for the students where they can download the materials, submit the assignment answer files to the server, and receive the validation results of them from the server. To affirm the rightness of the answer files, the validator program will automatically validate them by running *JUnit* and *Robolectric* on the background. The validation results will be shown with a web browser that students and teachers can quickly view them.

2.3.1 Learning Environment

For learning Android programming through APLAS, each student is required to utilize a computer device that complies with the minimum hardware specifications outlined in the guide document. The distribution of learning materials to students is facilitated through the web application system.

2.3.2 Operation Procedures of Platform

The operational procedures of the platform encompass activities on both the client and server sides. The following steps are undertaken for each learning topic:

1. The teacher uploads the learning materials for each topic to distribute them among the students.
2. Students download the learning materials and complete the assignments by writing *Java*, *XML*, or *DSL* of *Gradle* [21] source codes using *Android Studio* on their PCs.
3. Students validate the source codes through the provided test codes within the materials on Android Studio.
4. Students submit their answer codes and learning reports to the server via a web browser.
5. The server automatically validates the answer codes, and the results are stored in the database.
6. The teacher and students can access to the results through a web browser.

2.3.3 Learning Material

APLAS encompasses diverse learning topics that span a broad spectrum of Android programming. Each topic zeroes in on a specific aspect of the entire process of constructing an Android application. As illustrated in Figure 2.2, each learning topic comprises multiple tasks essential for developing a complete Android application. Students can successfully finalize the corresponding Android application by systematically addressing the tasks one after another. Each task represents a fundamental assignment tailored to achieve a specific learning objective.

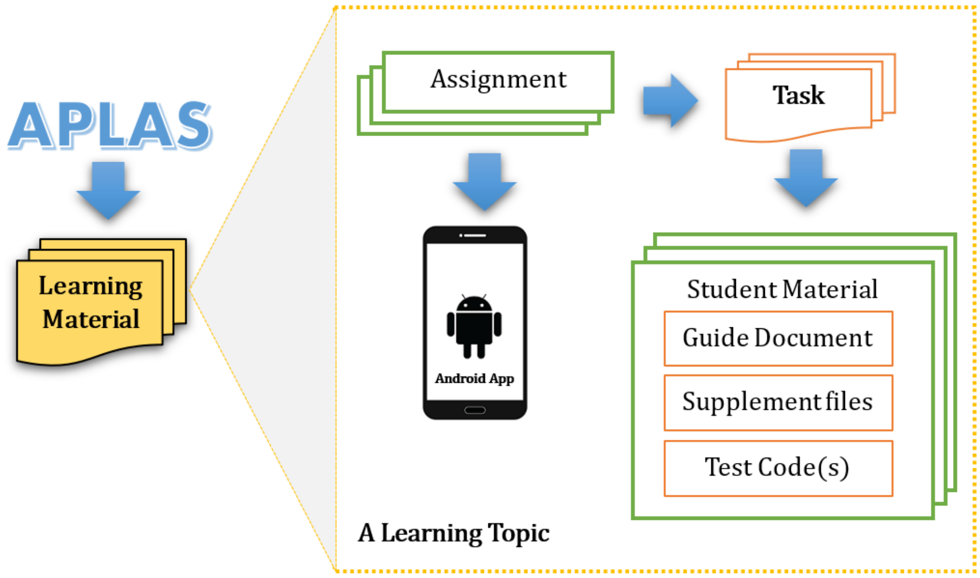


Figure 2.2: Structure of learning materials in *APLAS*.

The teacher will provide the file set comprising *guide documents*, *test codes*, and *supplement files* for each learning topic. The guide documents serve as manuals, directing students on tackling a specific task within the learning topic and solving the assignment. Test codes are employed to verify the accuracy of the student's answer codes. Supplement files encompass additional resources required to fulfil the task, such as fonts, images, videos, animations, and style resources.

2.3.4 Learning Process

The learning process at APLAS consists of several stages. Each stage covers several topics. Each topic consists of several tasks that students must complete in the order given. In each topic, a student has to solve the assignments by developing an Android application using Android Studio, as explained in the guide document. 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* by integrating the necessary Android components and libraries.
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.

2.4 Web Application Server

The web application offers the interfaces on web browsers to the students for downloading course materials, uploading assignment answer files, and obtaining validation results. The user authentication function is implemented to authorize three user roles, namely, the student, the teacher, and the administrator.

2.5 Validation

One learning topic in APLAS contains several tasks that must be solved by the students sequentially, one by one. The answer source code for each task must be validated using the test codes.

2.5.1 Answer Code Generation Using Android Studio

APLAS presumes that students will utilize Android Studio to compose their answer codes. Android Studio, recognized as the foremost *integrated development environment (IDE)*, comes bundled with *Android SDK*, allowing the development of Android applications through a blend of *Java* codes, *XML* codes, and *DSL of Gradle* codes. The APLAS architecture comprises various components operating on a *Java Virtual Machine*, as depicted in Figure 2.3.

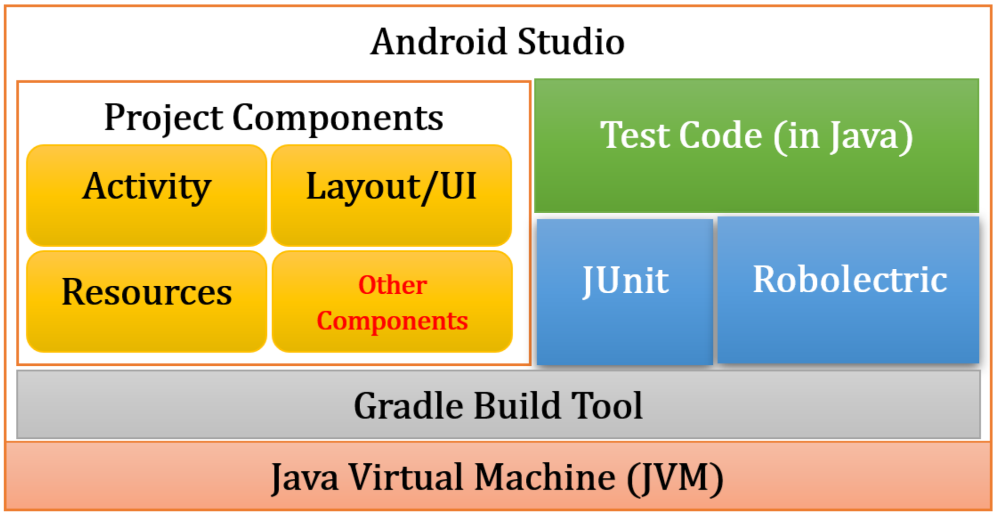


Figure 2.3: *APLAS* architecture in Android Studio.

2.5.2 Automatic Code Validation in Android Studio

In APLAS, the automatic validation function for answer codes is implemented using *JUnit* and *Robolectric*. These tools offer testing capabilities for source codes within *Android Studio*. The test codes are written in *Java,* combining *JUnit* for *unit testing* and *Robolectric* for *integrated testing*. The validation model and the validation components in APLAS are illustrated in Figure 2.4. Java source code testing can be done on *JUnit* using direct assertion methods. However, for specific features of the Android project, such as UI layout, Activity lifecycle, event listeners, and application resources, integration testing of the Android application necessitates the use of *Robolectric*.

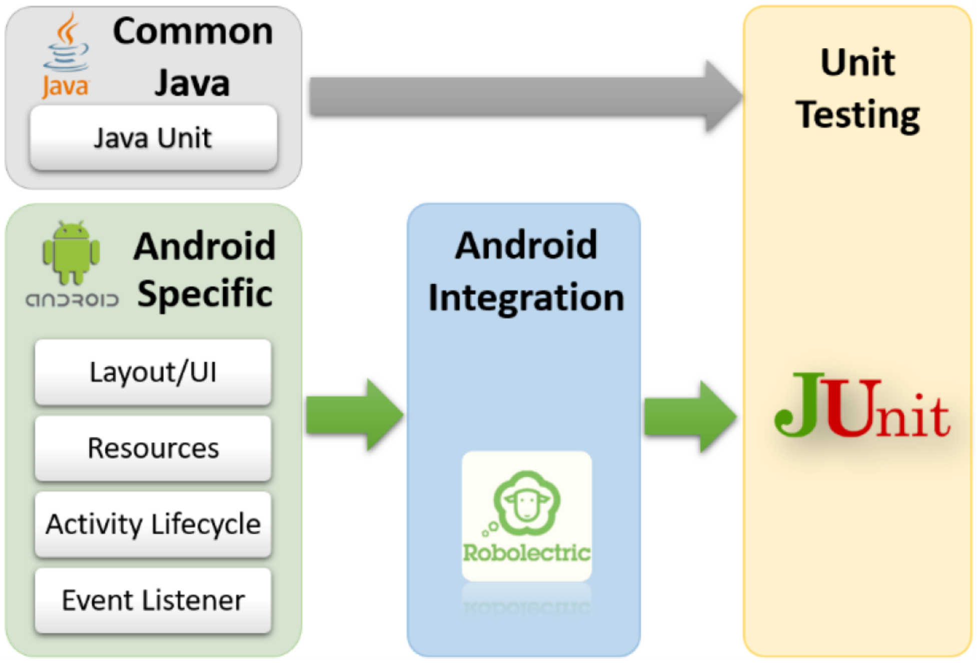


Figure 2.4: Validation process in *APLAS*.

2.5.3 Validation Procedure in Server

When a student submits answer codes, the validator program processes them by automatically executing the corresponding test codes for validations. The results are then displayed on the web interface for both the student and the teacher.

2.5.4 Validation Process in Server

The validation program is implemented on the server to automate the answer code validation process. The validation program is a multithreaded Java program that operates as a background job on the Linux operating system. As a crucial server-side component, its primary function is automatically validating the submitted answer codes. Constantly running in the background, it detects the new answer submissions and validates them by invoking *Gradle*. This involves compiling the folder containing the Android project and executing every test code, to ensure the accuracy of the answer codes within the folder. Subsequently, the program records the test results, execution time, any error messages (if present), and execution error messages (if any) in the database. Students and teachers can access these details through the web interface.

2.6 Summary

This chapter reviewed the *Android Programming Learning Assistance System (APLAS)*. It discussed the software architecture of the APLAS implementation, examining components such as the learning platform, web application server, and the validation process within APLAS.

Chapter 3

Implementation of APLAS Assignments without Guide Documents

This chapter presents the implementation of the assignments without guide documents and investigates the learning outcome of students in APLAS.

3.1 Introduction

The implementation of APLAS in the Android programming course has been proved to be highly effective. It garnered positive responses, motivated students to engage more deeply in learning, and notably enhanced their academic achievements [22]. APLAS addressed *programming challenges, complexities,* and *learning assistance issues* by providing students with comprehensive *guide documents* for learning and enabling the verification of source codes through the utilization of test codes. However, in practical programming, students often have to write source code or build projects without guide documents.

3.2 Assignment in APLAS

In APLAS, each learning topic is furnished with a set of files, including guide documents, test codes, and supplement files, which are provided by the teacher. The guide document functions as a guide, offering a guidance to students on how to approach a particular task within the learning topic and successfully complete the associated assignment.

The assignment solution process consists of the following five steps:

1. The student downloads the package of the files for this assignment from the APLAS server.
2. The student writes source codes to satisfy the specifications in the assignment on *Android Studio.*
3. The student runs the test codes on *Android Studio.*.
4. If the test results contain errors, the student will correct the source codes and go back to 3.
5. The student submits the source codes as the answer and the test results to the APLAS server.

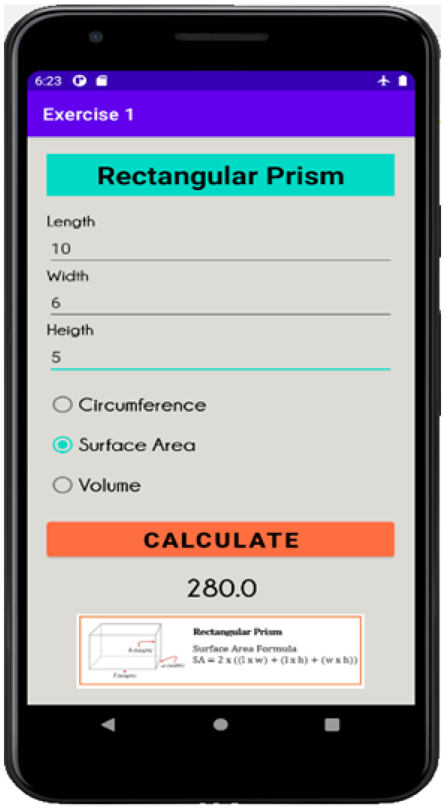
The *guide document* describes the details of the parameters and commands that should be included in the source codes. They are intended to direct students to complete the assignment. They contain *prerequisite learning topics*, *application descriptions*, *expected results*, *specifications*, *testing*, *and assignment submissions.*

3.3 Design Assignment without Guide Documents

In this chapter, the assignment is designed in APLAS without the *guidance document.* This assignment is intended to implement the required functionality in a basic application. This assignment makes it easier for students to start their works because several components developed in previous lessons in APLAS can be reused. To undertake this assignment, students must have studied fundamental topics in APLAS, including *basic UI* [9] and *basic activities.* [23]

3.3.1 Expected Interface for Assignment

A *Rectangular prism* or cuboid is a three-dimensional flat shape consisting of three pairs of rectangles. A cuboid has several terms for its sides, such as *length*, *width*, and *height*. In this assignment, students are asked to make applications to calculate the formulas for the *circumference*, *surface area*, and *volume* of *rectangular prism*. The application allows the user to input values for *length*, *width*, and *height*, then the user chooses one of the formulas among *circumference*, *surface area*, and *volume* when using the application. The user clicks the *calculate* to show the result and display the formula image in the *ImageView* component. The expected result is shown below.



Length
10
Width
6
Heigth
5

( ) Circumference
(✓) Surface Area
( ) Volume

280.0

Rectangular Prism
Surface Area Formula
SA = 2 x ((l x w) + (l x h) + (w x h))

Figure 3.1: Expected result for *rectangular prism* assignment.

3.3.2 Component and Layout of User Interface

Figure 3.2 illustrates the components and the layout of the user interface to be implemented. This application requires one *Activity* with widget components to make a user interface. Several *android widget* and features are adopted, such as *TextView*, *EditView*, *RadioButton*, *Button*, and *ImageView*. Table 3.2 shows the detailed specification of each used component. To facilitate application testing, students are suggested to use *attribute* for each component, as shown in Table 3.2.

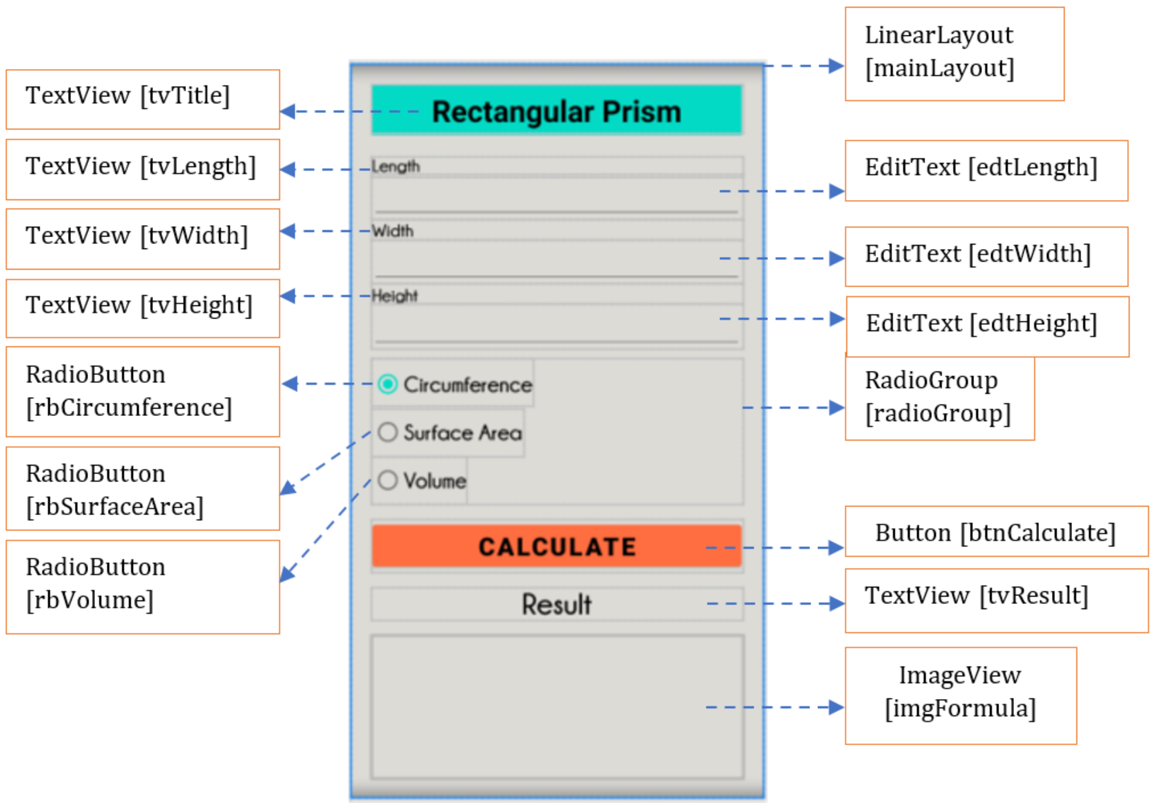


Figure 3.2: Components and layout of user interface.

3.4 Evaluation

In this section, we evaluate the implementation of the assignment without guide documents.

3.4.1 Evaluation Setup

For evaluations, we discuss the application results of this assignment to 54 undergraduate students in the IT department in Indonesia. These students have solved a sufficient number of assignments in APLAS using guide documents. After solving the assignments, they have to submit the source codes to the APLAS server.

3.4.2 Solving Activity Results

Table 4.1 shows the solution results of the students for an assignment in the rectangular prism topic. It was revealed that 53 students among 54 completed the assignment successfully. Only one student failed to write the source codes that passed the given test codes. The others completed the assignment without the guidance documents. Therefore, after solving a sufficient number of assignments in APLAS using guide documents, the students can write source codes for Android applications without guide documents for practical programming. Thus, the results prove that *Android programming* learning with APLAS is effective for learners. On the other hand, testing the code on the APLAS server takes 29-63 seconds for a single answer code.

Table 3.1: Student solution results in *rectangular prism* assignment.

# of attempted students	# of passed students	run time in APLAS server
54	53	29-63 sec

3.5 Summary

This chapter presented the implementation of assignments within APLAS, specifically those without accompanying guidance documents. The objective is to assess the learning outcomes of the APLAS system and recognize that in programming practice, students often need to write source code or build projects without guidance documents. The results confirm the effectiveness of APLAS learning on the assignments without guidance documents.

Table 3.2: Specification of user interface elements.

element	attribute	value
ViewGroup [LinearLayout]	id orientation padding background	@+id/mainLayout vertical 20dp @color/bgLayoutColor
EditText [edtLength]	id inputType	@+id/edtLength number
EditText [edtWidth]	id inputType	@+id/edtWidth number
EditText [edtHeight]	id inputType	@+id/edtHeight number
RadioGroup [radioGroup]	id orientation layoutMarginTop	@+id/radioGroup vertical 10sp
RadioButton [rbVolume]	id text textStyle textSize fontFamily	@+id/rbVolume @string/rbVolume bold 22sp @font/champagne
TextView [tvResult]	id text layoutMarginTop gravity textColor textSize textStyle fontFamily	@+id/tvResult @string/tvResult 15sp center @color/black 30sp bold @font/champagne
Button [btnCircumference]	id text fontFamily textSize textStyle checked	@+id/btnCircumference @string/btnCircumference @font/champagne 22sp bold true
Button [btnCalculate]	id text backgroundTint textColor textSize textStyle layoutMarginTop	@+id/btnCalcluate @string/btnCalculate @color/bgTvTitle @color/black 25sp bold @15sp
ImageView [imgFormula]	id layoutMarginTop	@+id/imgFormula 15sp

Chapter 4

Implementation of Solution Progress Monitoring Function in APLAS

This chapter presents the solution progress monitoring function in APLAS.

4.1 Introduction

In the current implementation of APLAS, the teacher can only check the students' final answers 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.

4.2 Android Library

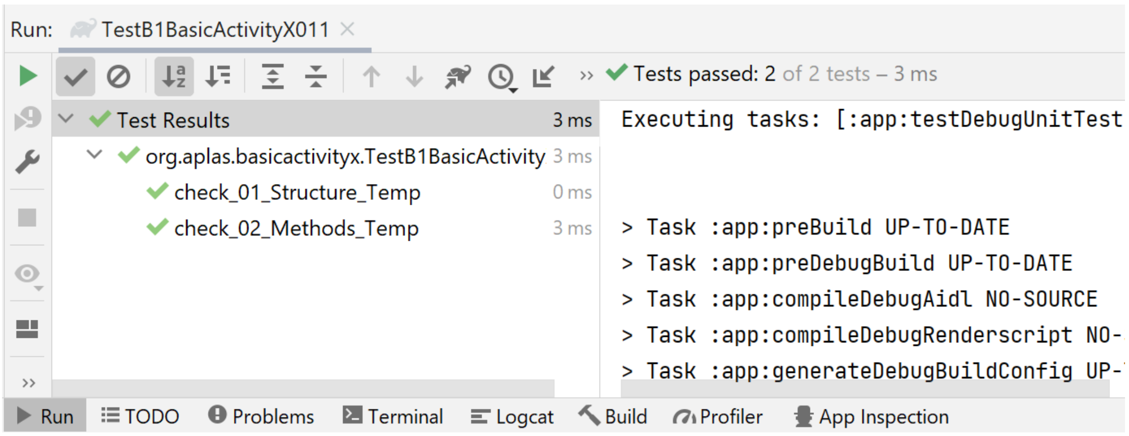
Android library has the same structure as an Android application module. It can contain everything required to create an application, such as the source code, the resource files, and the Android manifest [25]. Every day, new libraries and resources will be launched to expedite developments. The goal of a library developer is to simplify the complexity of the code and to package the code for future reuse.

4.3 Testing process in Android Application

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 [26].

4.3.1 Test Result View

The test results in Android Studio appear in the *Run* window. Figure 4.1 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.


Run: TestB1BasicActivityX011
Tests passed: 2 of 2 tests – 3 ms
Run
TODO
Problems
Terminal
Logcat
Build
Profiler
App Inspection

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

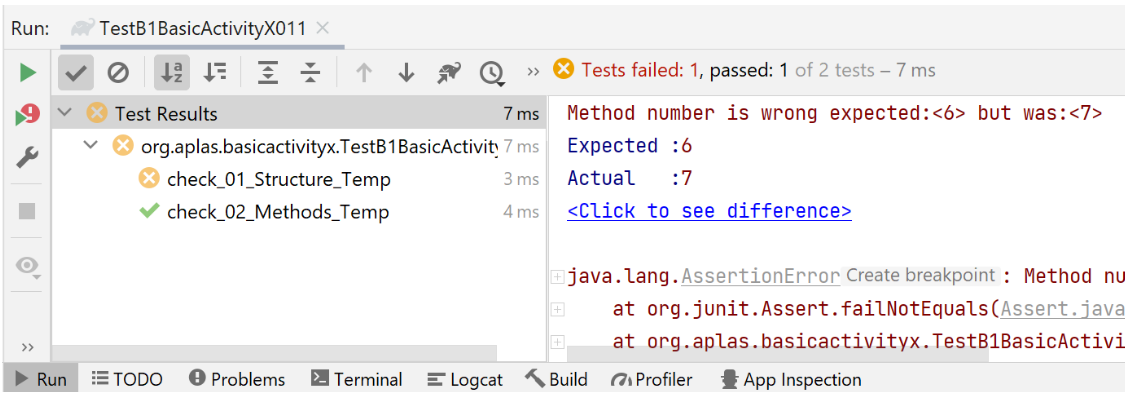
```
Executing tasks: [:app:testDebugUnitTest


> 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-
```

Figure 4.1: Successful test result.

4.3.2 Test Failure Analysis

The resulting window also displays the alert and the number of failed tests if they appear. Figure 4.2 shows an example of a failed test result. One failed 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 failed: 1, passed: 1 of 2 tests – 7 ms
Run
TODO
Problems
Terminal
Logcat
Build
Profiler
App Inspection

```
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
```

Figure 4.2: Failed test result.

4.4 Concepts of 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 codes 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 to reach the *PASSED* status for the current task by passing all the test cases in every test code.

Figure 4.3 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.

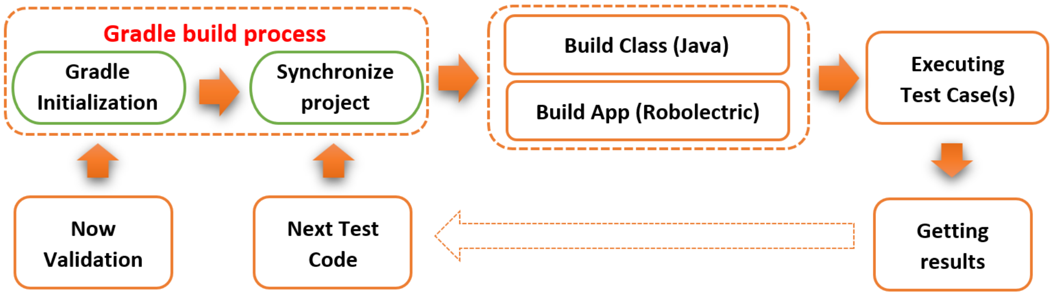


Figure 4.3: Unit testing process.

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.

4.5 Monitoring Function in APLAS

The *progress monitoring function* is built in to receive 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*. Using this program, the teacher can monitor the progress of solving the tasks on *Android Studio* by every student and can assist the students who may have difficulties in solving them.

4.5.1 Design and Implementation

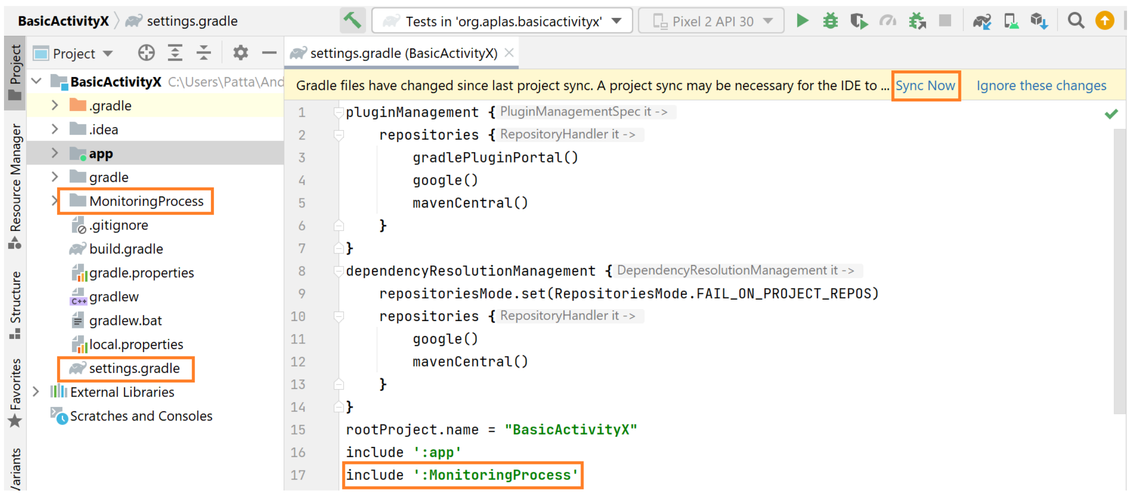
In *Android Studio*, a module refers to a discrete unit of functionality within an Android project. Each *Android Studio* project consists of one or more modules, each serving a specific purpose. The monitoring function is incorporated into a module within Android Studio. It is developed using the *Groovy* programming language.

Groovy is a dynamic, object-oriented programming language for the *Java Virtual Machine (JVM)*. It is designed to be concise, expressive, and compatible with Java [27]. Besides, *Groovy* is commonly used in the context of build scripts, particularly with the *Gradle* build system. *Gradle* is a powerful build automation tool that is often used for building and managing Android projects. Android Studio uses *Gradle* scripts, and these scripts are typically written in either *Groovy* or *Kotlin DSL* (domain-specific language).

An *Application Programming Interface (API)* [28] is established on the server side to acquire data from *monitoring functions*. This API is developed using the *PHP* programming language[29], while the database operates on *MySQL* [30].

Figure 4.4 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*.

Then, the client-side program of the *process monitoring function* will be executed automatically after running the test code. As shown in Figure 4.5, the corresponding script will be executed as a dependent of the *Gradle* task. Their roles are described as follows:



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to ...

```
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'
```

Figure 4.4: Adding module monitoring function in *Android Studio*.

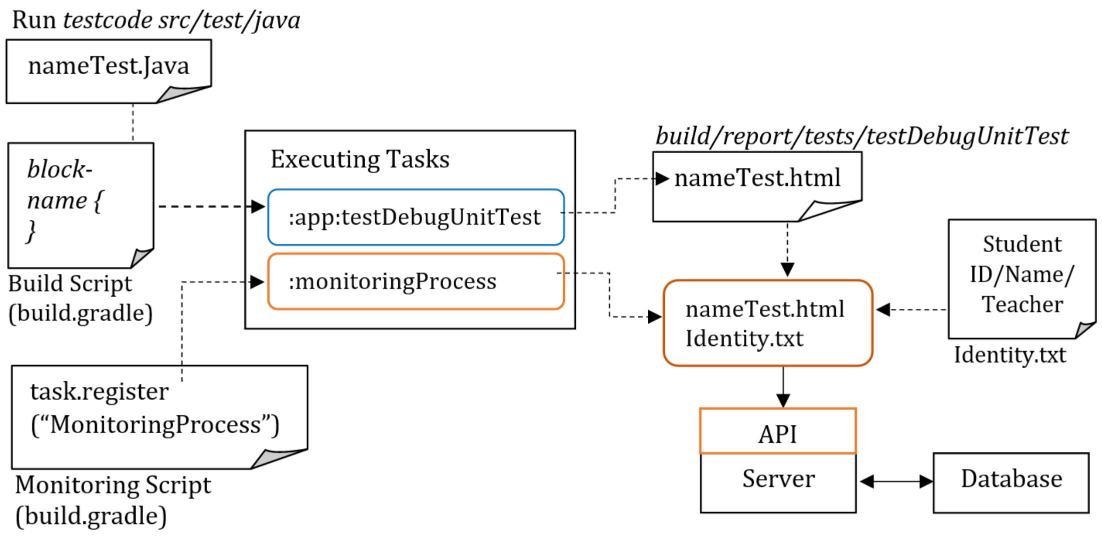


Figure 4.5: Flow of processes in Android Studio.

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 it to the server.

6. The server side of the API program will receive a set of reports from the client and then store them in the database.

4.5.2 Flow of Function

As in the APLAS architecture, *APLAS* consists of the *Student's PC* and the *APLAS server*. On the *Student's PC*, a student will complete the tasks on *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.

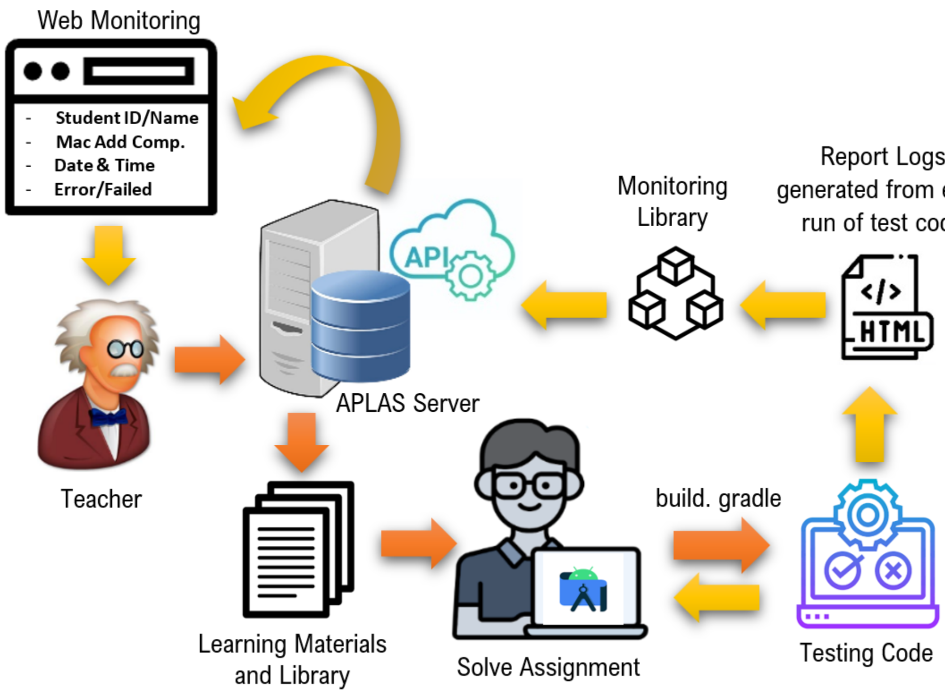


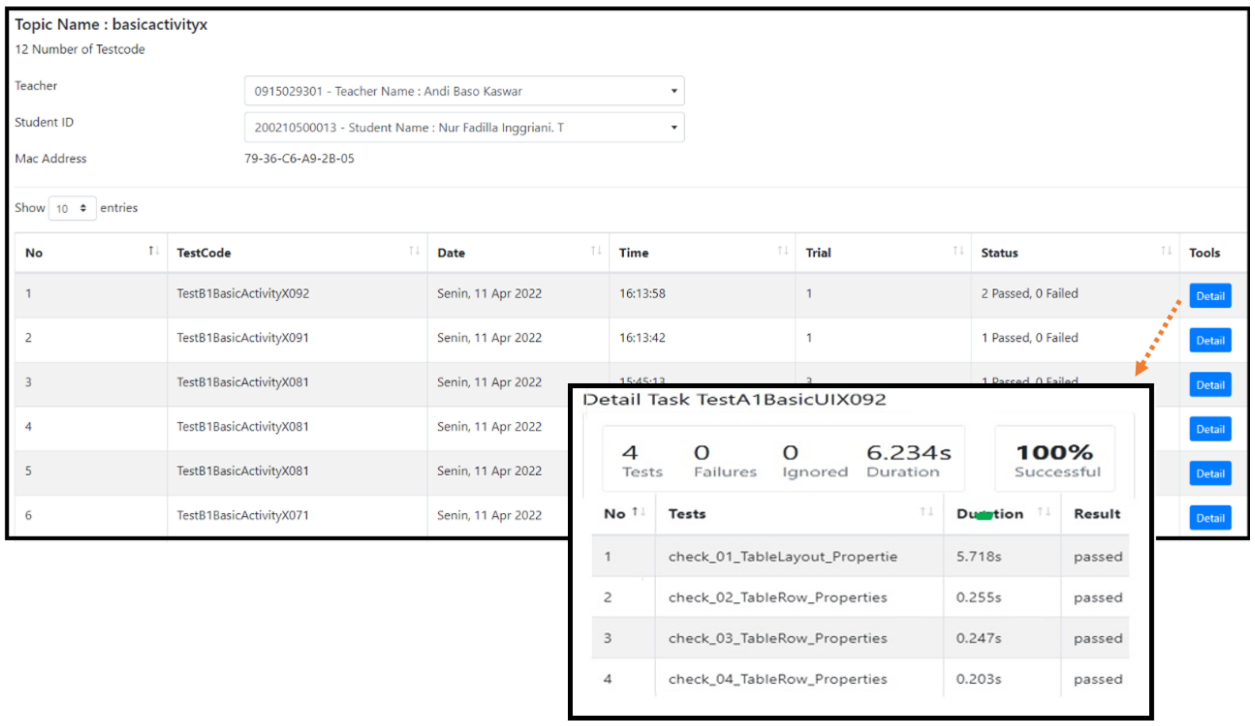
Figure 4.6: Flow of process monitoring function.

Figure 4.6 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.

4.5.3 Web Application

The program for the web-based monitoring interface receives the progress reports from the database on a regular basis. Figure 4.7 shows the user interface. The interface shows the current state of each task or test 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. The details menu will showcase the success or errors of any tests, along with the duration for which the test code was executed.


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				Detail
5	TestB1BasicActivityX081	Senin, 11 Apr 2022				Detail
6	TestB1BasicActivityX071	Senin, 11 Apr 2022				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

Figure 4.7: Interface for task status in one topic.

4.6 Evaluation

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

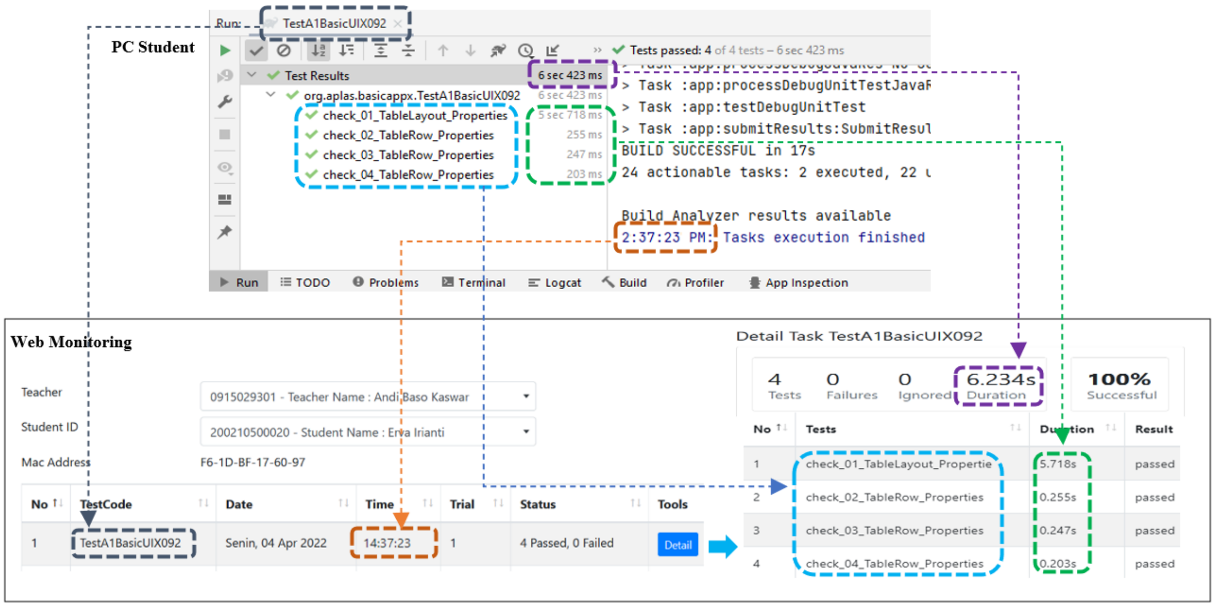
4.6.1 Evaluation Setup

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

4.6.2 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 Figure 4.7, the validation results of the tasks in *Android Studio* were successfully shown on the web interface in real time. Besides, as shown in Figure 4.8, 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.


PC Student
Web Monitoring
Run TODO Problems Terminal Logcat Build Profiler App Inspection

Run: TestA1BasicUIX092

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

Figure 4.8: Interface for status details in one task.

4.6.3 Solving Activity Results

Figure 4.9 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 262*min* minutes. The shortest time was 213*min* with running the test codes by 15 times, and the longest time was 311*min* with running the test codes by 25 times.

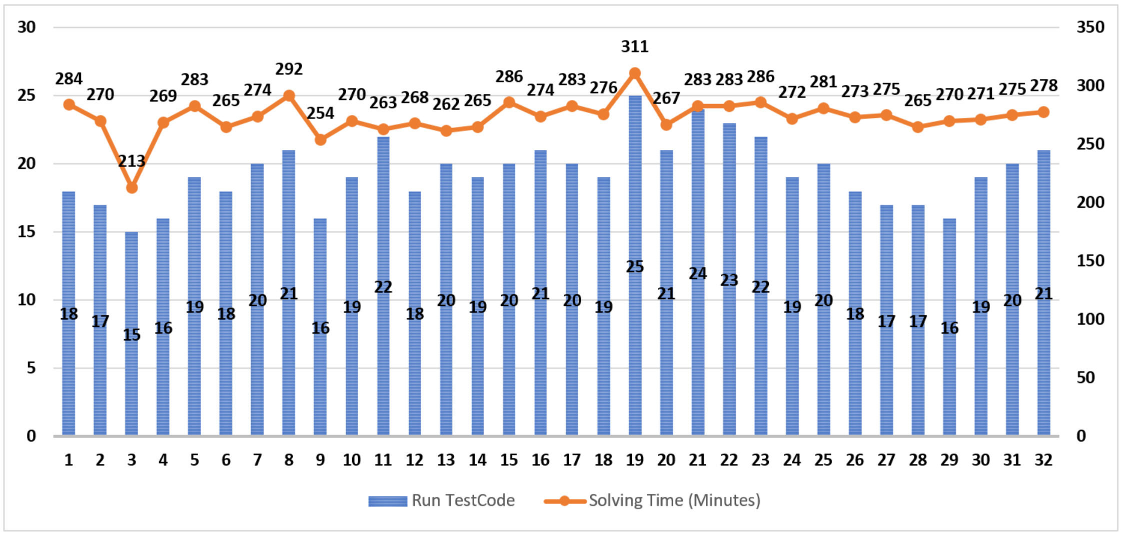


Figure 4.9: Solving performances of students.

Figure 4.10 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 with the proper guidance from the teacher, especially for the students who may have difficulty in solving the tasks in the *process monitoring function*.

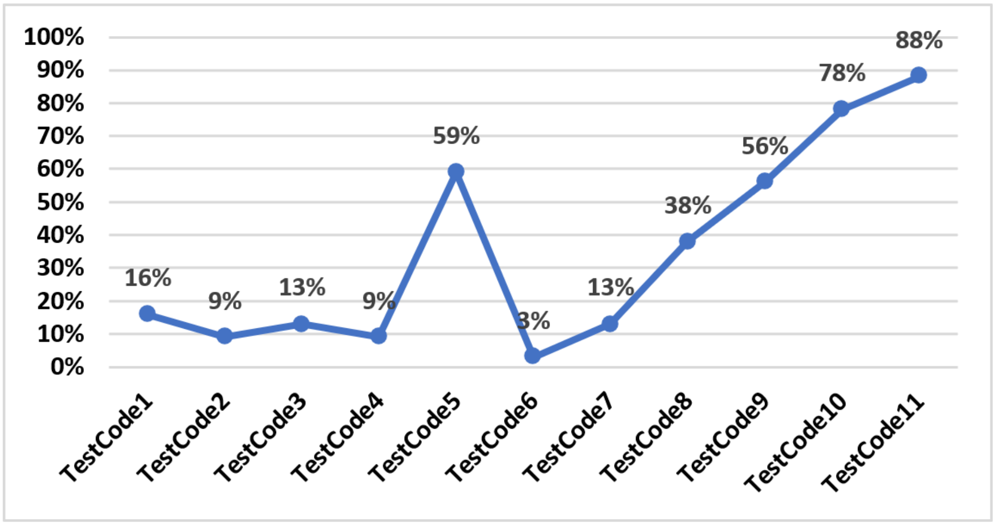


Figure 4.10: Students Correct when first running test code

4.6.4 Comparison with and Without Proposal

Table 4.1 compares the number of students who failed each task with and without the *process monitoring function* [23]. 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 4.1: Comparison of number 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

4.7 Summary

This chapter 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 the log files of *Android Studio* and shows the learning progress based on the log files at the web interfaces. The results showed that by using this function, the number of dropped-out students was reduced because the teacher could identify and assist the students experiencing difficulties at the early stage of the course.

Chapter 5

Implementation of Grammar-concept Understanding Problem for Flutter Programming Learning Assistance System

In this chapter, we present the grammar-concept understanding problem for the flutter programming learning assistance system.

5.1 Introduction

Traditionally, mobile app development has involved writing separate codebases for each platform, such as *iOS (Apple)* and *Android (Google)*. However, with the advent of cross-platform mobile programming, a paradigm shift has occurred in the mobile development landscape. Cross-platform mobile programming refers to the development of mobile applications that can run on multiple operating systems and platforms using a single codebase. It leverages frameworks and tools that allow developers to write a code once and deploy it across different platforms, to save time and efforts [31]. Therefore, it is important for students to study the cross-platform app development.

5.2 Overview JPLAS

Java Programming Learning Assistant System (JPLAS) has been developed as a web-based platform designed for exercises and learning programming languages [33]. JPLAS provides an offline answering function, enabling students to respond to problems even in the absence of an internet connection [32].

Currently, JPLAS features various types of exercise problems to accommodate a variety of students at different learning levels. The problem types in JPLAS are as follows:

1. *Grammar Concept Understanding Problem (GUP)*: This type of problem instance comprises a source code along with a set of questions focusing on grammar concepts or behaviours exhibited by the code [15] [16].

2. *Value Trace Problem (VTP)*: This type of problem necessitates students to trace the actual values of crucial variables in a code during its execution [34].

3. *Element Fill-in-blank Problem (EFP)*: This problem requires students to fill in the blank elements in a given Java code [32].

4. *Statement Fill-in-blank Problem (SFB)*: This type of problem mandates students to fill in the blank elements within a provided Java code [35].

5. *Code Writing Problem (CWP)*: This problem prompts students to create an entire code from scratch that adheres to the specifications outlined in the test code [33].

6. *Code Amendment Problem (CAP)*: In this problem type, students are presented with a Java source code containing various missing or erroneous elements, referred to as a problem code [36].

7. *Code Completion Problem (CCP)*: In this problem, students are presented with a source code containing several missing elements, with no explicit indication of their presence [37].

5.3 Cross-platform Flutter

Cross-platform development refers to the practice of creating software applications that can run on multiple operating systems or platforms. The primary goal is to write a code once and deploy it on various platforms, reducing the need for separate codebases for different environments. This approach is often chosen in order to save time, resources, and efforts compared to developing separate native applications for each platform [31].

Recently, *Flutter* has gained popularity as a software development toolkit for creating cross-platform applications compatible with Android and iOS. As a result, numerous software developers have embraced *Flutter* as the preferred choice [38]. *Flutte*r is an open-source UI software development toolkit created by *Google* [39]. It is used to build natively compiled applications for mobile, web, and desktop environments from a single codebase. *Flutter* uses the *Dart* programming language and provides a rich set of pre-designed widgets that make it easy to create custom and visually appealing user interfaces.

5.4 Design of GUP for Flutter

Based on our previous studies of JPLAS, the *grammar-concept understanding problem (GUP)*[15] [16] can be adopted as the initial learning task for novice students in cross-platform mobile programming, explicitly focusing on *Flutter*.

A GUP instance comprises a source code, a collection of questions, and their respective correct answers. Each question pertains to a fundamental grammar concept in *Flutter* programming that is presented in the source code. The question prompts the user to identify the corresponding keyword within the code. The correctness of an answer is determined by performing string matching with the correct answer. The GUP generation algorithm is implemented to assist teachers in generating GUP instances from the provided code.

5.4.1 GUP for Flutter

A GUP instance consists of a *Flutter* source code, a set of questions, and their correct answers. Each question pertains to a fundamental grammar concept in mobile programming with the *Flutter* framework, which is found within the source code. The question prompts the student to identify the corresponding element or keyword within the code. The student's answer is evaluated by comparing it to the correct answer using string matching.

5.4.2 Generation Procedure of GUP

A GUP instance file is generated through the following procedure:

- Read the text file that contains the Flutter source code.
- Extract the keywords from the source code that match the listed keywords.
- Choose the corresponding question from the *question list* for each extracted keyword.
- Identify the element within the source code that represents the correct answer.
- Discard any duplicated questions and correct answers that are selected as a pair.
- Generate the output GUP instance file, which includes the source code, corresponding questions, and correct answers.

5.4.3 Selection of Flutter Source Code

To utilize the algorithm, the teacher needs to initially prepare a source code file that encompasses the grammar concepts to be explored by students through solving the generated GUP instance. The algorithm will read this source code file and generate the corresponding GUP instance input file using the procedure outlined in Section 5.4.2. Before implementing the algorithm, creating the list that contains the keywords and their related questions is essential.

5.4.4 Generating Assignments

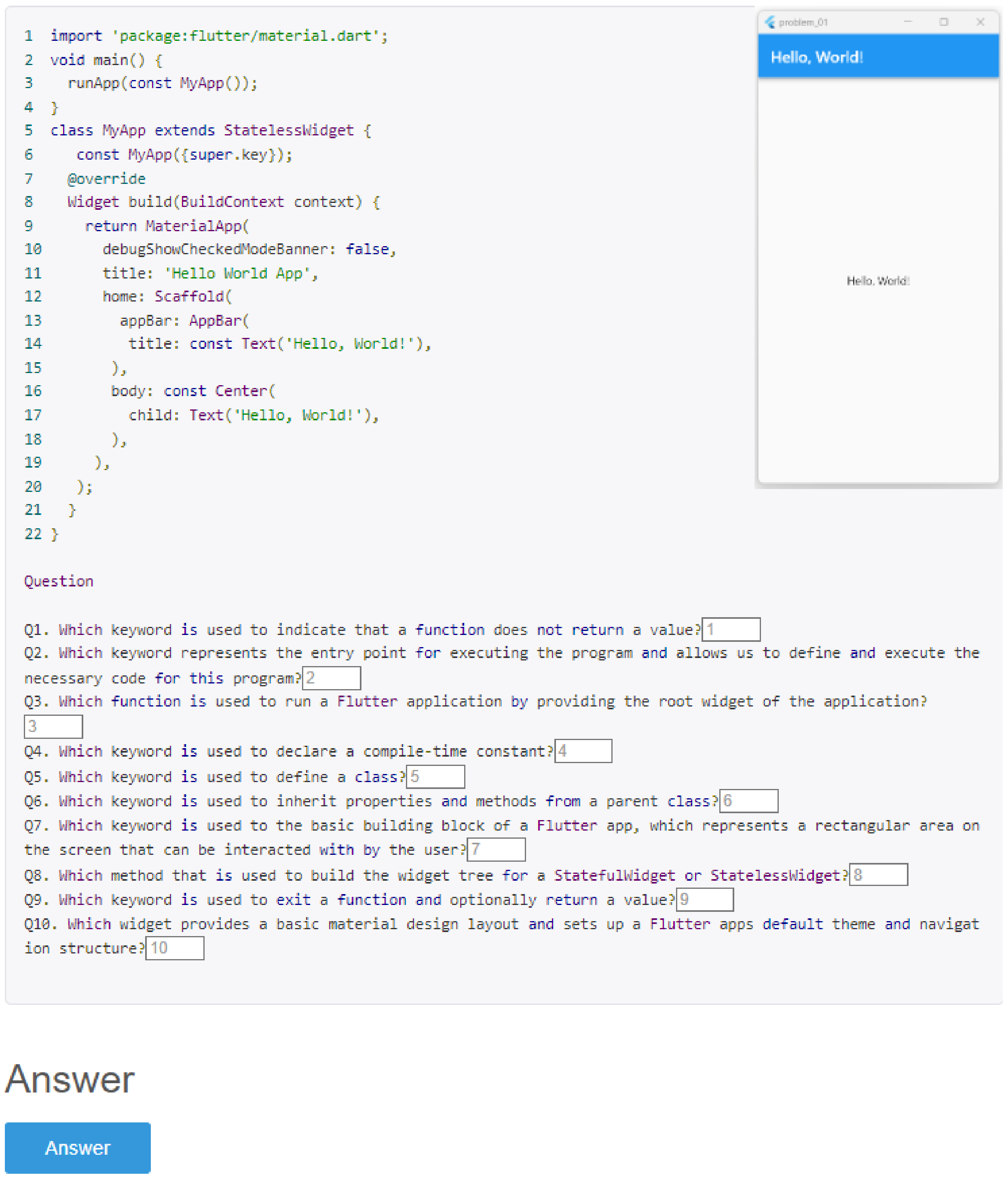
The algorithm requires the preparation of a predefined list containing keywords and their corresponding questions. Table 5.1 presents the list of 71 keywords related to the grammar topic of *Flutter* programming, accompanied by their respective questions. Each keyword is associated with a specific question that provides its meaning. The students are encouraged to develop comprehensive understanding of important grammar concepts in mobile programming using *Flutter* by attentively reading the questions and providing the corresponding keywords as answers.

5.4.5 Answer Interface for GUP

Figure 5.1 illustrates the answer interface using a web browser for solving a GUP Instance. In the interface, when a student provides an incorrect answer, the input form is highlighted in *red*, whereas a correct answer is displayed with a *white* background. Through this interface, students can review and submit their answers repeatedly until all of them are correct. The system automatically keeps the number of answer submission times and the student's answers for each submission.

Table 5.1: Keywords and q uestion in GUP for Mobile Programming Using Flutter Frameworks.

keywords	question	instance ID
AppBar	Which widget is used to create a top app bar, which typically contains the app's name, navigation icons, and other actions?	2,4,5,7,9
body	Which property is used to specify the main content of a 'Scaffold' widget?	2,3,4,5,6,9,11,12
border	Which property is used to define the border decoration for the table cells?	12
bottomNavigationBar	Which widget is used to create a navigation bar at the bottom of the screen, allowing the user to switch between multiple pages or sections of an app?	16
BottomNavigationBarItem	Which widget is used to represent a single item in a BottomNavigationBar?	16
build	Which method that is used to build the widget tree for a StatefulWidget or StatelessWidget?	1,2,6,11,
BuildContext	Which object provides contextual information about where a widget is located in the widget tree?	3,4,5,6,8,10,17
CheckboxListTile	Which widget in Flutter is used to create a checkbox with a label that can be toggled on and off?	13
child	Which property is used to specify a single child widget for a parent widget?	2,3,4,5,6,8,9,12
children	Which property is used to specify a list of child widgets for a parent widget?	3,4,8
class	Which keyword is used to define a class?	1,2,17,
Column	Which widget arranges its children vertically in a column, with optional spacing and alignment?	7,13,14,21
const	Which keyword is used to declare a compile-time constant?	1,7,9,
ConstrainedBox	Which widget is used to set constraints on the size of its child widget?	9
Container	Which widget can contain other widgets, and is often used as a layout element to arrange other widgets within it?	3,8,9
crossAxisCount	Which property is used to define the number of items in each row or column of the grid in Flutters GridView widget?	11
decoration	Which property is used to specify the visual decoration of a widget, such as BoxDecoration for Container or InputDecoration for TextField?	18
ElevatedButton	Which widget is used to create a flat button with a label?	6,17,18,19,21
Expanded	Which widget expands to fill all available space in its parent widget along a vertical or horizontal axis?	8
extends	Which keyword is used to inherit properties and methods from a parent class?	1
flex	Which property is used in Expanded widgets to determine the flex factor, which specifies how much space the widget should take up relative to other Expanded widgets in the same row or column?	8
GestureDetector	Which widget detects various gestures made by the user, such as taps, drags, and long-presses?	22
GridView	Which widget is used to create a grid of widgets, either with a fixed number of columns or with a dynamic number of columns based on the available space?	11
home	Which property in Flutter is used to set the widget that should be displayed as the main content of a screen?	17
Image	Which widget displays an image from a local file, network URL, or memory asset?	7
import	Which keyword is used to import definitions from another Dart module?	6,7
initState()	Which widget is used to initialize stateful data for the widget, and is called after the widget is mounted?	20
item	Which property is commonly used with widgets such as ListView, GridView, and DropdownButton to define the individual items that make up a list or a selection menu?	16
itemBuilder	Which property is used in ListView and GridView widgets in Flutter to provide a callback function that returns the widget for each item in the list or grid?	10
key	Which keyword is used to unique identifier can be assigned to a widget to help Flutter identify it and track changes to it?	5
late	Which keyword is used to declare a variable that will be initialized later, but before it is used?	20
leading	Which property is used to specify a widget that should be displayed to the left of the primary content?	10
ListView	Which widget displays a scrollable list of widgets, either vertically or horizontally?	10
main	Which keyword represents the entry point for executing the program and allows us to define and execute the necessary code for this program?	1,2,8
mainAxisAlignment	Which property is used in Row or Column widgets to determine how the child widgets are aligned along the main axis (horizontal for Row, vertical for Column)?	3,6,7,13,
MaterialApp	Which widget provides a basic material design layout and sets up a Flutter apps default theme and navigation structure?	1,2,3,8,9,10,21
Navigator.pop	Which method is used to remove the current screen from the navigation stack and return to the previous screen?	17
Navigator.push	Which methods are used for navigating between screens in Flutter?	17
onChanged	Which property is used to specify a callback function that is triggered when the user interacts with a widget, such as tapping on a button?	13,14,15
onTap	Which property is used to specify the callback function that is called when a widget is tapped, such as InkWell or GestureDetector?	10,16,22
Padding	Which property is used to specify the amount of space between the content of a widget and its border?	18
Positioned	Which widget positions its child at a specific location within a Stack?	4
return	Which keyword is used to exit a function and optionally return a value?	1,18
Scaffold	Which widget provides a basic layout structure for a screen, including an app bar, a body area, and optional drawers and snack bars?	3,5,9,15,22



```dart
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
   const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Hello World App',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Hello, World!'),
        ),
        body: const Center(
          child: Text('Hello, World!'),
        ),
      ),
    );
  }
}
```

Question

Q1. Which keyword is used to indicate that a function does not return a value? 1

Q2. Which keyword represents the entry point for executing the program and allows us to define and execute the necessary code for this program? 2

Q3. Which function is used to run a Flutter application by providing the root widget of the application? 3

Q4. Which keyword is used to declare a compile-time constant? 4

Q5. Which keyword is used to define a class? 5

Q6. Which keyword is used to inherit properties and methods from a parent class? 6

Q7. Which keyword is used to the basic building block of a Flutter app, which represents a rectangular area on the screen that can be interacted with by the user? 7

Q8. Which method that is used to build the widget tree for a StatefulWidget or StatelessWidget? 8

Q9. Which keyword is used to exit a function and optionally return a value? 9

Q10. Which widget provides a basic material design layout and sets up a Flutter apps default theme and navigation structure? 10

Answer

Answer

Figure 5.1: GUP answer interface.

5.5 Evaluation

In this section, we evaluate the proposal through application to undergraduate students who are taking the mobile programming course in the IT department at Makassar State University in Indonesia.

5.5.1 Evaluation setup

For evaluations, we generated 22 *GUP* instances with 177 questions from various source codes. They cover essential topics of using widgets in building mobile apps using the *Flutter* framework. It has been verified that the questions generated are appropriate for novice-level students. Then, we asked 109 undergraduate students to solve the given problems using the answer interface.

5.5.2 Individual GUP Instance Results

Figure 5.2 illustrates the correct answer rates for the problems, where it ranges from 98.16% to 100%. The number of submission times varies between 3.12 and 3.43 across the different problems. Most problems exhibit high correct rates, indicating good performances of the students in answering the questions. Notably, Problem *ID=8*, Problem *ID=21*, and Problem *ID=22* achieved perfect scores, attaining 100% correct rate. In contrast, Problem *ID=13* has the lowest correct rate, averaging 98.16%. This problem seems to pose more challenges for the students, resulting in a slightly lower correct rate compared to other problems. An interesting observation is that despite the lower correct rate, Problem *ID=13* has the highest number of submission times among all the problems, with an average of 3.36 submissions per student. This implies that students invested more effort to solve this problem, despite the lower accuracy.

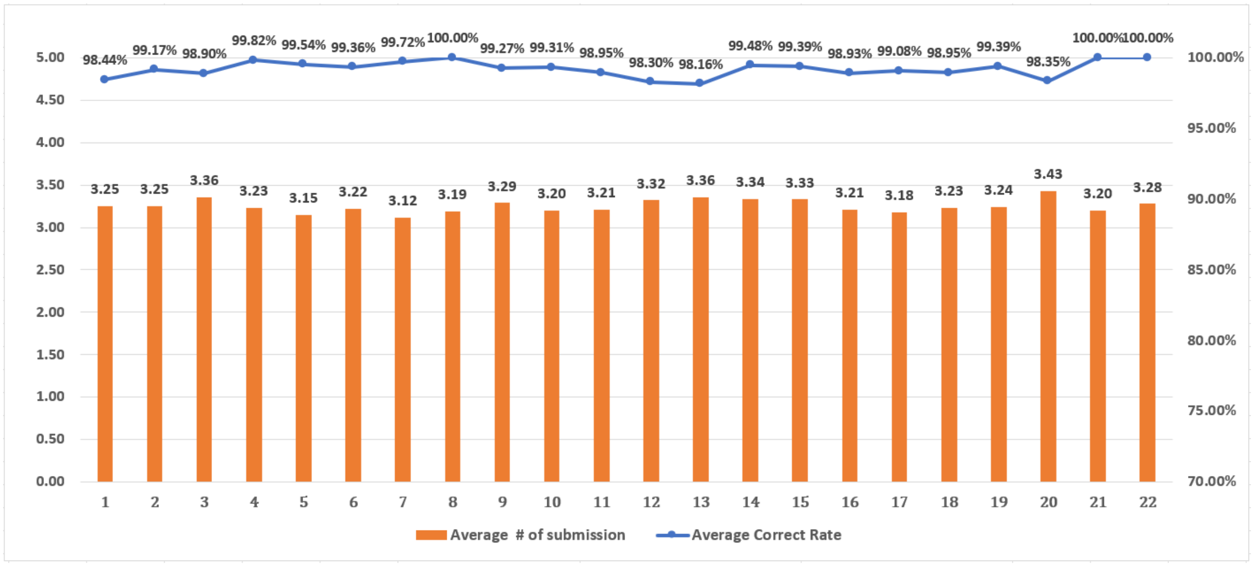


Figure 5.2: Performances for each GUP instances

Overall, the students consistently displayed engagements and efforts by submitting their answers approximately three or four times, on average, for each problem. This indicates their active involvements in reviewing and refining their responses, with the aim of achieving the highest level of accuracy and understanding possible.

5.5.3 Correct Answer Results

Table 5.2 illustrates the distribution of the number of correct answers by the students. Based on the data presented in the table, it can be observed that out of the total of 109 students, 92 students (84.40%) achieved the perfect score, attaining the 100% accuracy rate, although by submitting the answer multiple times to reach the correct answers. This indicates that these students possess good understanding of the fundamental grammar concepts required for the *Flutter* programming language. On the other hand, there are 17 students (15.60%) who answered below 100% accuracy. These students may benefit from refreshing their memory and revisiting the concepts and principles of *Flutter* programming to enhance their understanding and improve their accuracy in future instances.

Table 5.2: Correct answer rate distribution of students

range of correct answer rate (%)	# of students	rate of students (%)
90 - 95	9	8.26
96 - 99	8	7.34
100	92	84.40

In summary, the results provide insights into the distribution of correct answers among the students, highlighting that the majority of students achieved the high accuracy while emphasizing the need for further improvement and reinforcement for those who scored below 100%.

5.5.4 Submission Times Results

The distribution of the number of submission times provides insights into how frequently students are submitting their answers. It helps identify patterns and trends of their engagements in the given tasks, highlighting the range of efforts and dedication exhibited by the students in the learning process.

Table 5.3: Submission times distribution of students

range of submission times	# of students	# of rate of students (%)
22 - 35	6	5.50
36 - 55	8	7.34
56 - 75	62	56.88
76 - 100	27	24.77
101 - 130	4	3.67
131 - 150	2	1.83

Table 5.3 provides an overview of the distribution of the number of answer submission times by the students. Fourteen students among 109 (12.84%) submitted their answers by under 55 times or submitted their answers 1 to 2 times per instance. They were confident in their initial answers and required minimal adjustments or revisions to refine their answers. 89 students (81.65%) submitted their answers between 56 and 100 times, or on average, 3 to 4 times per instance. This implies that they undergo multiple cycles of reviews and modifications to enhance their understanding, accuracy, and thoroughness. Six students (5.50%) submitted their answers over 100 times, or on average more than 5 times per instance, despite having correct answer rates above 90%. This

indicates that these students approached the GUP instances with serious commitments, submitting their answers multiple times and demonstrating strong dedication to continuously improving and achieving the correct answers.

5.6 Summary

This chapter presented cross-platform and implementations of the *grammar-concept understanding problem (GUP)* in mobile programming using *Flutter* frameworks, serving as an introductory exploration of *Flutter* programming for novice students. A meticulous selection process resulted in 71 keywords, each accompanied by specific questions to generate GUP instances. To assess the effectiveness of this approach, 22 GUP instances were generated, encompassing 71 keywords that addressed fundamental grammar concepts. The study's outcomes affirmed the successful implementation of the proposed method in accurately gauging students' levels of comprehension.

Chapter 6

Related Works in Literature

In [40], Kose introduced a web-based system designed and developed to support project-based learning in the web design and programming course. The system provides efficient and advanced environments for students to learn how to design and implement websites effectively.

In [41], Yang et al. created an interactive testing system for use in computer programming classes. This technique supports students in examining their programming misconceptions and improving programming skills by evaluating various code statements and validating them. An experiment was conducted in the course for developing Android applications.

In [42], Hayashi et al. presented a paradigm of *collaborative learning* for teaching computer programming using the flipped classroom instructional strategy. To measure the effectiveness of this method, the examination results of students were compared with and without the proposal.

In [43], Hundt et al. presented the *System for Automated Code Evaluation (SAUCE)* as an interactive online tool to study parallel programming. *SAUCE* picks up common parallel algorithms based on *C++11*, *OpenMP*, *MPI*, and *CUDA* threads that can be interactively inserted in HPC or parallel computing courses.

In [44], Su et al. introduced a web-based application for learning programming fundamentals, including the scratch tool. The goal is to help students with online lessons and diagnostic reports based on the analysis of their learning portfolios.

In [45], Rekhawi et al. developed a web-based intelligent tutoring system for teaching Android programming. The system gives lessons on overviews of Android programming. It facilitates adaptable demonstrations and has favourable effects on evaluators.

In [46], Amalfitano et al. presented *juGULAR*, a hybrid GUI exploration approach to automatically discover gate GUIs during application explorations. *juGULAR* was designed in the modular software architecture for the Android mobile platform. Experiment results showed that the ability to investigate covered activities and covered lines of code was improved.

In [47], Domenach et al. proposed an online system called *Programming Assignment Submission System (PASS)*. It is a web-based interactive application system where students can send their answers to programming assignments and get feedback in real-time. *PASS* was tested in C++ programming assignments.

In [48], Chaudhari et al. proposed a system for online accommodations and evaluations of Java programming assignments. The objective is to research existing instruments, and design and develop intelligent web programs that PC students may submit programming assignments and get constant feedback.

In [49], Madeja et al. carried out studies to investigate students' blunders while attempting to solve Android programming projects. To successfully finish the testing, the students used Android testing tools for unit testing, integration testing, and user interface testing.

In [50], Khan et al. introduced a software called *AUTOGRADER* to evaluate the correctness of programming tasks automatically. It compares the answer codes to the reference implementations provided by the instructor to ensure that they follow the same logic. The instructor may use this tool for various purposes, including creating test cases and administering tests.

In [15] Aung et al. presented the *Grammar-Concept Understanding Problem (GUP)* as a new type of exercise problem in JPLAS. The results show that the proposal is effective in identifying the students who do not understand Java programming well and need more instruction from the teacher.

In [16] Lu et al. presented the GUP in the C Programming Learning Assistant system (CPLAS). Their solution results revealed the hard grammar concepts and the progress of C programming studies by novice students.

In [38], Yan et al. presented an implementation of an automated *Dart* code verification system for assisting mobile application programming learning using *Flutter*. Their work focused on developing a learning support system that integrates an automated *Dart* code verification mechanism, which draws upon a software testing methodology commonly used in Android application development.

In [51], Neeman et al. presented developing a cross-platform mobile course using a multi-paradigm library. This research focuses on creating a mobile course utilizing the *React JavaScript Library* to develop cross-platform applications. *React JavaScript* apps encompass various features, including imperative, declarative, functional, object-oriented, markup, and scripting language elements.

In [52], Rogers et al. presented teaching cross-platform technology and democracy. They discuss designing and implementing an innovative mobile computing course to improve students' technical abilities and decision-making skills. Additionally, it incorporated an aspect of democracy by involving students in the design process of the curriculum.

In [53], Xiang et al. presented the design and implementation of a primary school app inventor programming mobile learning platform based on *WeChat* applet. The goal is to create a *WeChat* widget mobile learning platform that provides online learning resources for students, enabling them to utilize their fragmented time effectively and enhance their learning efficiency.

In [54], Qin et al. introduced a series of programming teaching methods for iOS APP mobile development, incorporating creative design to enhance students' programming thinking. This approach offers novel insights for more effective and practical programming education.

In [55], Krusche et al. introduced *ArTEMiS*, an automated assessment management system designed for interactive learning. This system automatically evaluates programming exercise solutions and offers immediate feedback. Additionally, *ArTEMiS* includes an online code editor with interactive exercise instructions.

In [56], Tung et al. introduced *Programming Learning Web (PLWeb)*, a comprehensive system designed to manage programming exercises, aid teachers in exercise design, and facilitate students' programming studies. *PLWeb* provides an *integrated development environment (IDE)* as part of its offerings.

In [57], Costa et al. proposed a model that assesses the effectiveness of a computer programming learning system. This model supports students in overcoming learning obstacles and provides guidance for teachers in shaping programming and teaching strategies.

Chapter 7

Conclusion

This thesis presented the three implementations for improvements of Android and Flutter programming learning assistance systems.

Firstly, I implemented assignments within APLAS, specifically those without accompanying *guidance documents*. The objective is to assess the learning outcomes within the APLAS system and recognize that in programming practice, students often need to write source code or build projects without the help of guidance documents. The results confirm the effectiveness of APLAS learning on this assignment.

Secondly, I implemented *solution progress monitoring function* in APLAS. This functionality operates by having the program on the student's PC send a test log file to the APLAS server each time the student executes test code to evaluate the answer code in *Android Studio*. Subsequently, the web interface displays crucial metrics, including the number of completed assignments, the time spent on them, and the number of validations submitted by each student. Through the utilization of this innovative function, teachers gain the ability to identify and support students facing challenges in completing assignments in APLAS. Real-time data facilitates timely teacher intervention and provides support for students in need of additional assistance. The results showed that by using this function, the number of dropped-out students was reduced because the teacher could identify and assist the students experiencing difficulties at the early stage of the course.

Thirdly, I implemented the *Grammar Concept Understanding Problem (GUP)* for *Flutter* cross-platform mobile programming learning. With the advent of cross-platform mobile programming, a paradigm shift has occurred in the mobile development landscape. Therefore, it is important for students to learn cross-platform application development. *Flutter* is gaining popularity as a software development kit for creating cross-platform applications compatible with *Android* and *iOS*. Based on our previous study of *Programming Learning Assistance Systems (PLAS)*, *Concept-Grammar Understanding Problems (GUP)* can be adopted as initial learning tasks for novice students in cross-platform mobile programming, with a special focus on *Flutter*. The study's outcomes affirmed the successful implementation of the proposed method in accurately gauging students' levels of comprehension.

In future works, we aim to integrate the problem model from the *programming learning assistant system* into cross-platform mobile programming learning, with a particular focus on *Flutter*. This involves exploring the creation of *Code Modification Questions (CMP)*, *Code Writing Problems (CWP)*, and the utilization of test code for validating program code. Additionally, we plan to develop a completion activity monitoring function in *Flutter*.

Bibliography

[1] P. Nawrocki, K. Wrona, M. Marczak, B. Sniezynski "AGH University of Science and Technology," IEEE Computer Society, 2021.

[2] M. Ohshiro, Y. Yamakawa, K. J. Mackin, K. Matsushita, and E. Nunohiro, "Programming learning support system with learning progress monitoring feature," in Proc. SCIS & ISIS, pp. 1465-1468, Dec. 2010.

[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. Sys., vol. 16, no. 1, pp. 115-135, 2019.

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

[6] JUnit: a simple framework to write repeatable tests, `https://junit.org/junit4/` [accessed on 14 May 2023.]

[7] Robolectric: a framework that brings fast and reliable unit tests to Android, `http://robolectric.org/` [Accessed on 14 May 2023.]

[8] Google Developer, "Android Studio provides the fastest tools for building apps on every type of Android device," `https://developer.android.com/studio`. [Accessed: 14 May 2023.]

[9] Y. W. Syaifudin, N. Funabiki, and M. Kuribayashi, "Learning model for Android programming learning assistant system," Proc. IEICE General Conf., Tokyo, Japan, 2019.

[10] A. R. Patta, N. Funabiki, Y. W. Syaifuddin and W. -C. Kao, "An Investigation of Learning Outcomes Using Assignment without Guide Documents in Android Programming Learning Assistance System," IEEE ICCE-TW, pp. 195-196, 2022.

[11] A. R. Patta, N. Funabiki, M. Kuribayashi and Y. W. Syaifuddin, "An Implementation of Solution Progress Monitoring Function in Android Programming Learning Assistance System," Int. J. Inf. Educ. Technol., vol 13, no. 10, pp. 1597-1603, October 2023.

[12] A. R. Patta, N. Funabiki, Y. W. Syaifuddin and W. -C. Kao, "An Implementation of Solving Activity Monitoring Function in Android Programming Learning Assistance System," 5th ICVEE 2022, pp. 84-88, 2022.

[13] A. R. Patta, N. Funabiki, X. Lu and Y. W. Syaifuddin, “A Study of Grammar-concept Understanding Problem for Flutter Cross-platform Mobile Programming Learning,” 6th ICVEE 2023, pp. 195-196, 2023.

[14] N. Ishihara, N. Funabiki, M. Kuribayashi, and W. C. Kao, “A software architecture for Java programming learning assistant system,” Int. J. Comput. Soft. Eng., vol. 2, no. 1, 2017.

[15] S. T. Aung, N. Funabiki, Y. W. Syaifuddin, H. H. S. Kyaw, “A Proposal of Grammar-concept Understanding Problem in Java Programming Learning Assistant System,” J. Adv. Inf. Tech., vol. 12, no. 4, pp. 342-350, November 2021.

[16] X. Lu, N. Funabiki, S. T. Aung, H. H. S. Kyaw, K. Ueda, W. C. Kao, “A Study of Grammar-concept Understanding Problem in C Programming Learning Assistant System,” ITE Trans. on MTA, vol. 10, no. 4, pp. 198-207, 2022.

[17] P. G. F. Garcia. and F.D. Rosa, “RoBlock – a web app for programming learning,” Int. Journal of Emerging Technologies in Learning (IJET), Vol. 11, No. 12, pp. 45-53, 2016.

[18] L. Koskela, Test Driven: “Practical TDD and Acceptance TDD for Java Developer, Manning Publications,” Shelter Island, New York, NY 2008.

[19] Y. W. Syaifudin, N. Funabiki, M. Mentari, H. E. Dien, I. Mu’aasyiqiin, M. Kuribayashi, W.-C. Kao, “A Web-based Online Platform of Distribution, Collection, and Validation for Assignments in Android Programming Learning Assistance System,” IAENG Int. J. Comput. Sci., vol. 29, no.3, September 2021.

[20] Y. W. Syaifudin, N. Funabiki, M. Kuribayashi, “A Proposal of Advanced Widgets Learning Topic for Interactive Application in Android Programming Learning Assistance System,” SN COMPUT. SCI. 2, 172, 2021.

[21] Gradle Inc.: `https://docs.gradle.org/6.7/release-notes.html`. [Accessed on 14 May 2023.]

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

[23] 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,” Proc. Forum Inf. Tech., Okayama, Japan, 2019.

[24] N. Smyth, Android Studio 3.0 Development Essentials 8th ed. CreateSpace Independent Publishing Platform; 2017.

[25] Android Library, Google Developers, `https://developer.android.com/studio/projects/android-library/` [Accessed on 29 November 2022].

[26] Test in Android: Google Developers, test in Android Studio, `https://developer.android.com/studio/build` [accessed on 29 November 2022].

[27] Apache Groovy, `https://groovy-lang.org/` [Accessed on 5 December 2022).

[28] Application Programming Interface (API), https://en.wikipedia.org/wiki/API [Accessed on 5 December 2022.

[29] Hypertext Preprocessor (PHP), https://www.php.net/ [Accessed on 5 December 2022.

[30] MySQL, https://www.mysql.com/ [Accessed on 5 December 2022.

[31] M. Mahendra. and B. Anggorojati, "Evaluating the performance of Android-based Cross-Platform App Development Frameworks," Int. Conf. on Comm. and Inf. Processing. (ICCIP), pp. 32-37, 2020.

[32] N. Funabiki, Y. Matsushima, T. Nakanishi, and N. Amano,"A Java programming learning assistant system using test-driven development method," IAENG Int. J. Computer Science, vol. 40, no. 1, pp. 38-46, February 2013

[33] N. Funabiki, H. Masaoka, N. Ishihara, I-W. Lai, and W-C. Kao,"Offline answering function for fill-in-blank problems in Java programming learning assistant system," in Proc. IEEE ICCE-Taiwan, pp. 324-325, 2016.

[34] K. K. Zaw, N. Funabiki, and W.-C. Kao, "A proposal of value trace problem for algorithm code reading in Java programming learning assistant system," Inform. Eng. Exp., vol. 1, no.3, pp. 9-18, Sep. 2015

[35] N. Ishihara, N. Funabiki, and W.-C. Kao, "A proposal of statement fill-in-blank problem using program dependence graph in Java programming learning assistant system," Info. Engr. Exp., vol. 1, no. 3, pp. 19-28, Sept. 2015.

[36] H.H.S. Kyaw, N. Funabiki, and W.-C. Kao, "A proposal of code amendment problem in Java programming learning assistant system," International Journal of Information and Education Technology (IJIET), vol. 10, No. 10, pp. 751-756, Oct. 2020.

[37] H.H.S. Kyaw, S.S. Wint, N. Funabiki, and W.-C. Kao, "A code completion problem in Java programming learning assistant system," IAENG International Journal of Computer Science (IJCS), vol. 47, No. 3, pp. 350-359, Sept. 2020.

[38] Y. W. Syaifudin, A. S. Hatjrianto, N. Funabiki, D. Y. Liliana, A. B. Kaswar, U. Nurhasan, "An Implementation of Automatic Dart Code Verification for Mobile Application Programming Learning Assistance System Using Flutter," Int. Conf. Elec. Inf. Tech., pp 322-326, 2022.

[39] Flutter, https://flutter.dev/ [Accessed on 5 April 2023].

[40] 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.

[41] T. C. Yang, S. J. Yang, G. J. Hwang, "Development of an interactive test system for students' improving learning outcomes in a computer programming course". In: Proceedings of IEEE 14th International Conference on advance learning Technology, Athens, Greece, pp.637-639, 2014.

[42] Y. Hayashi, K. I. Fukamachi, H. Komatsugawa, ”Collaborative Learning in Computer Programming Courses That Adopted the Flipped Classroom”, In Proceedings of the International Conference on Learning Technology and Computer Engineering, Taipei, Taiwan, pp. 209-212, 2015.

[43] C. Hundt, M. Schlarb, B. Schmidt, ”SAUCE: a web application for interactive teaching and learning of parallel programming”, J Parallel Distributed Comput, Vol. 105, pp. 163-173, July, 2017.

[44] J. M. Su, S. J. Wang, ”A Web-based learning activity integrated with scratch tool to support programming learning”. In: Proceedings of the 10th international conference on ubi-media computer and workshops (Ubi-Media), Pattaya, Thailand, pp. 1-4, 2017.

[45] H. A. A. Rekhawi, 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.

[46] D. Amalfitano, V. Riccio, N. Amatucci, V. technologyA. R. Fasolino. ”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.

[47] F. Domenach and G. Portides, “PASS - a programming assignment submission system,” In Proceeding of the International Conference on Interactive Mobile Communication Technologies and Learning (IMCL), pp. 195-199, November 2015.

[48] S. Chaudhari and N. J. Uke, ”A method for submitting programming language assignment for effective evaluations,” in Proc. ICCUBEA, pp. 1-4, 2018.

S. Chaudhari and N. J. Uke, ”A Method for Submitting Programming Language Assignment for Effective Evaluations,” In Proceeding of the Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), pp. 1-4, 2018.

[49] M. Madeja and J. Poruban, “Automatic assessment of assignments for Android application programming courses,” In Proceeding of the IEEE 14th International Scientific Conference on Informatics, pp. 232-237, 2017.

[50] 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.

[51] A. Neeman, ”Developing a cross-platform mobile course using a multi-paradigm library.”, J. Comput. Sci. Coll., Vol. 37, no. 4, pp 71, October 2021.

[52] M. P. Rogers and B. Siever, ”Teaching Cross-Platform Technology and Democracy.”, J. Comput. Sci. Coll., Vol 38, no. 5, pp. 75–86, November 2022.

[53] K. Xiang, F. Xu, D. Hu, H. Zhou and M. Li, ”Design and Implementation of Primary School App Inventor Programming Mobile Learning Platform based on Wechat Applet,” Int. Conf. on Comp. Sci. and Edu. (ICCSE),” pp. 547-552, 2020.

[54] L. Qin, B. Li, L. -P. Yang, ”Programming Thinking Training and Course Design for iOS Mobile Development,” Int. Conf. on Comp. Sci. and Edu. (ICCSE), pp. 577-582, 2020.

[55] S. Krusche and A. Seitz, ”ArTEMiS - an automatic assessment management system for interactive learning,” Proc. ITiCSE, pp. 155-159, May 2016.

[56] S. H. Tung, T. Te Lin, Y. H. Lin, ”An exercise management system for teaching programming,” J. Softw. Vol. 8, no. 7, pp. 1718-1725, July 2013.

[57] C. J. Costa, M. Aparicio, ”Evaluating success of a programming learning tool,” Proc. Int. Conf. Syst. Design Comm., pp. 73-78, May 2014.