An Implementation of Multimedia Resources Topic for in Flutter Programming Learning Assistance System

1st Yan Watequlis Syaifudin
Dept. of Information Technology
State Polytechnic of Malang
Malang, Indonesia
qulis@polinema.ac.id

2nd Akhmadheta Hafid Prasetyawan
Dept. of Information Technology
State Polytechnic of Malang
Malang, Indonesia
2041720221@student.polinema.ac.id

3rd Endah Septa Sintiya
Dept. of Information Technology
State Polytechnic of Malang
Malang, Indonesia
e.septa@polinema.ac.id

4th Nobuo Funabiki
Dept. of Elec. and Comm. Engineering
Okayama University
Japan, Indonesia
funabiki@okayama-u.ac.jp

5th Indrazno Siradjuddin
Dept. of Electrical Engineering
State Polytechnic of Malang
Malang, Indonesia
indrazno@polinema.ac.id

6th Hidayati Nur Chasanah
Dept. of Information Technology
State Polytechnic of Malang
Malang, Indonesia
1931710028@student.polinema.ac.id

***Abstract*—The increasing demand for proficient mobile app developers necessitates an effective learning framework. This research explores the integration of automated testing in a self-paced learning framework for developing multimedia applications using Flutter, an open-source UI software development tool. Motivated by the rising popularity of Flutter in mobile application development, particularly for its cross-platform capabilities and customizable widgets, the study addresses the gap in mobile programming education by designing and implementing learning modules that incorporate test-driven development (TDD) practices. The learning topic, part of the *Flutter Programming Learning Assistance System (FPLAS)*, facilitates self-learning in Android programming through a project-based approach. Evaluation through student feedback from 40 participants showed significant improvements in understanding, engagement, and project completion rates, resulting in a 100% success rate and positive feedback, although some constructive suggestions for improvement were noted. These findings suggest that integrating TDD with structured instructional modules can greatly benefit mobile programming education by encouraging self-directed learning and enhancing practical coding skills.**

***Index Terms*—Flutter; multimedia applications; TDD; self-learning; learning assistance.**

I. Introduction

The proliferation of smartphones and mobile applications has fundamentally altered how we communicate, work, and entertain ourselves [1]. As the mobile application market continues to grow, understanding how software works and the ability to write programs are now common requirements when hiring employees [2]. There is also an increasing demand for skilled developers proficient in cross-platform frameworks such as Flutter. Flutter, an open-source UI software development kit created by Google, has garnered significant attention due to its capability to create natively compiled applications for mobile, web, and desktop from a single codebase [3]. This versatility and efficiency make it an attractive choice for developers and educators alike.

Educational frameworks in mobile programming must evolve to keep pace with these technological advancements. Traditional classroom-based instruction often falls short in providing the hands-on, practical experience necessary for mastering mobile development. Consequently, there has been a shift towards self-directed learning (SDL), which empowers learners to take control of their educational journey. SDL has been shown to foster critical thinking, problem-solving skills, and a deeper understanding of complex subjects [4].

However, self-directed learning in mobile programming presents unique challenges. Learners often struggle with the absence of structured guidance and immediate feedback, which are crucial for effective learning. Timely feedback is necessary to motivate students to learn and improve their activities [5]. To address these issues, this study integrates automated testing into the instructional design. Automated testing, particularly through the use of pre-prepared test files, provides learners with resources to validate their code and receive instant feedback. This approach has been proven to improve code quality, facilitate early error detection, and enhance learner engagement [6].

Moreover, the integration of multimedia elements in mobile applications adds another layer of complexity. Multimedia applications, which include audio, video, and interactive content, require a comprehensive understanding of both user interface (UI) design and user interaction functionality. Previous research highlights the importance of project-based learning and real-world application development in acquiring these skills [7]. Therefore, this study aims to develop a comprehensive instructional module that leverages SDL principles, automated testing resources, and multimedia application development using Flutter.

Several studies have explored the benefits of SDL in programming education. For instance, Knowles' theory of andragogy emphasizes the importance of self-directed learning for adult learners, suggesting that they learn best when they take responsibility for their education [8]. Similarly, Garrison's model of SDL highlights the interplay between self-management, self-monitoring, and motivation in the learning process [9]. In the context of programming, SDL has been linked to improved problem-solving abilities and greater adaptability to new technologies [4].

Furthermore, the implementation of automated testing in educational settings has shown promising results. Mekterović et al. demonstrated that automated testing can provide immediate feedback, helping learners identify and rectify errors early in the development process [10]. Barra et al. reviewed various systems for automatic assessment of programming assignments, concluding that such systems can significantly improve learning outcomes by ensuring consistent and objective evaluation [11].

This paper introduces the development of a topic of self-learning to develop a mobile application that integrates multimedia resources, namely *Multimedia App*, in *Flutter Programming Learning Assistance System (FPLAS)* [12]. Specifically, it focuses on providing a self-paced learning platform that allows students to learn mobile programming using the Flutter framework. Applying a test-driven development method [13] replicating the project-based learning experience in a computer-based learning system requires the implementation of automated testing, including unit and widget testing to generate *automated assistance* [14]. Flutter incorporates a native testing framework, 'flutter_test', which facilitates both types of automated testing for Flutter-based applications.

To provide diverse learning topics, FPLAS provides *learning topics* that represent each subject. The *Multimedia App* topic provides a self-directed learning experience to develop multimedia applications with multimedia interaction from various sources. It combines sources and types of mixed media resources and controls the interaction with them which requires the ability to utilize widgets effectively, especially interactive media widgets. To provide project-based learning, an application has been planned, namely '*MediaPlayer*'. This application contains a splash, home page and media player for audio and video files. Through six learning stages to be completed sequentially, students can build this application and achieve the learning objectives.

The remainder of this paper is structured into five sections, which include: Section II presents several studies related to this study, Section III reviews the concept of FPLAS, Section IV explains the implementation of *Multimedia App* learning topic, Section V evaluates the application of this learning topic to students, Lastly, the implementation results are concluded in Section VI with the presentation of the future works.

II. Related Works

The integration of automated testing in programming education has shown significant improvements in code quality and learning outcomes. Tampubolon et al. conducted a comprehensive literature review on the challenges of applying test-driven development (TDD) in software engineering, highlighting its benefits in educational contexts [15]. Combéfis, S. demonstrated that automated test generation can enhance learning in programming courses by providing students with immediate feedback on their code [16]. Their findings are supported by Paiva et al., who emphasized the importance of automated testing in providing consistent and objective evaluation of programming assignments [17].

Falode et al. developed an interactive mobile application for learning educational technology concepts, which incorporated elements of SDL and multimedia. Their study found that interactive, self-paced learning modules significantly improved student engagement and understanding [18]. In the context of mobile programming, SDL has been linked to improved problem-solving abilities and greater adaptability to new technologies. Hmelo-Silver found that PBL fosters deep learning and critical thinking skills by encouraging students to engage in self-directed inquiry and problem-solving [19].

In previous research, the authors proposed FPLAS [14], which features an automated Dart code verification tool inspired by the *Android Programming Learning Assistance System (APLAS)* [20]. This system, positively reviewed by 40 IT students, offers learning materials and assignments focused on simple app development, assisting students in learning and correctly completing Flutter-based project assignments.

III. Flutter Programming Learning Assistance System

The concepts of FPLAS and TDD strategy appropriation are clarified in this section.

A. Overview

As an open-source system from Google, Flutter permits engineers to make feature-rich, cross-platform applications with a single code base, sparing time and assets.Flutter ensures high-quality app development that is robust, user-friendly, and runs flawlessly on both Android and iOS platforms [21]. Flutter's ubiquity in versatile app improvement, particularly for interactive media application, has expanded massively due to its adaptability and effectiveness.

FPLAS is a self-paced learning system that allows students to learn Flutter programming independently, supported by computer-based teaching. Figure 1 shows the flow of the FPLAS learning model. The FPLAS learning model provides learning materials in the form of guide files, supplementary files, and test codes for students to start learning independently. Following the provided guidelines, the student completes the given tasks and reviews his/her own written codes using the provided test codes before collecting the answer codes. Essentially, it provides a code review tool that helps students improve their written code to meet the requirements of the set task.

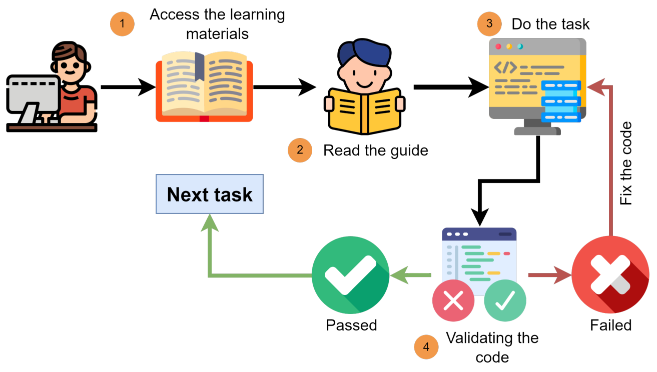


Fig. 1. Flutter programming self-learning model

B. *Application of TDD Method for Self-Directed Learning Model*

Various learning methods or models that utilize computers have been used, such as Computer Assisted Learning (CAL) [22], to offer an interactive and individualized learning experience. In addition, the application of Test-Driven Development (TDD) methods has proven to be an effective approach in self-paced programming learning systems. This method involves a cycle of developing tests, generating code to meet those tests, and refining the code to improve its design, with the main goal of ensuring the accuracy of the software through continuous verification that the code meets the specified requirements through automated testing. This approach allows students to learn at their own pace and provides immediate feedback [23].

C. *Automatic Code Verification in Flutter*

Automated Dart code verification in Flutter can be achieved by integrating automated testing into the development process. This practice is crucial for ensuring software quality and code precision, streamlining the development workflow. By running automated tests regularly and consistently, developers can detect code errors early. Flutter provides extensive testing frameworks and tools to support the creation and execution of these tests, namely flutter_test package. The primary categories of automated testing in Flutter are unit testing and widget testing.

D. *Providing Automated Assistance through Testing in Flutter*

The implementation of automated testing in Flutter necessitates the definition of test files, the creation of test cases, and their execution using builtin testing frameworks like `flutter_test`. Central to these testing procedures is the ’`expect`’ function, which assumes a critical role in specifying test assertions. Listing 1 illustrates a code snippet aimed at verifying the existence of the ’`SvgPicture`’ widget within the application’s widget hierarchy. This function guarantees that the application properly displays the SvgPicture widget; should the widget not appear, it generates an error message ’*Expected to find a SvgPicture widget.*’ to provide immediate feedback.

Listing 1
WIDGET TESTING IN FLUTTER

```
import
    'package:flutter_test/flutter_test.dart';
import 'package:flutter_svg/svg.dart';

void main() {
  testWidgets('Check for SvgPicture widget',
      (WidgetTester tester) async {
    final svgPictureFinder =
        find.byType(SvgPicture);
    expect(
      svgPictureFinder,
      findsOneWidget,
      reason: 'Expected to find an SvgPicture
          widget',
    );
  });
}
```

IV. Implementation of Mobile Application Multimedia Resources Topic

This section covers the concept of learning objectives, multimedia applications, and UI widgets in Flutter, as well as the application for student assignments and the different stages of learning.

A. *Learning Objective*

These studies underscore the benefits of SDL and automated testing in programming education but reveal a lack of comprehensive instructional modules specifically for multimedia applications in Flutter. This research aims to bridge this gap by developing a structured learning module that integrates SDL principles, automated testing resources, and multimedia application development using Flutter. Focusing on a learning topic at FPLAS, namely *Multimedia App*, the goal is to equip students with the knowledge and skills to create mobile applications that integrate diverse multimedia sources. By providing a self-paced learning system for mobile multimedia application development, students will master the use of development frameworks and tools to build cross-platform applications that offer a seamless and attractive user experience.

B. *Multimedia Application in Flutter*

Multimedia is a combination of various elements such as text, sound, images, and animations, enabling users to navigate, interact, or communicate within an application [24]. In the context of mobile development using Flutter, multimedia refers to the integration of diverse media elements like `text`, `audio`, `images`, and `video` into mobile apps created with the Flutter framework. This allows developers to create engaging and interactive user experiences. Integrating multimedia in Flutter can be achieved by leveraging various packages and libraries available on pub.dev. Some popular packages that address multimedia utilities in Flutter include:

1) google_fonts: *google_fonts* allows the integration of fonts from the Google Fonts collection into a Flutter application, providing flexibility in typography. For instance, to apply the `Roboto` font to text, simply add the `style` property with `GoogleFonts.roboto()` to the relevant text widget.

Listing 2
APPLYING GOOGLE FONTS IN FLUTTER

```
Text('Hello, Flutter!', style:
    GoogleFonts.roboto()),
```

2) flutter_svg: *flutter_svg* is used to display images in the SVG format, which scales without losing quality. This is ideal for displaying icons or logos that need to remain sharp at various screen sizes. For example, to display the Flutter logo in SVG format:

Listing 3
DISPLAYING SVG IMAGES IN FLUTTER

```
SvgPicture.asset('assets/flutter_logo.svg'),
```

3) audioplayers: *audioplayers* facilitates audio playback and control, allowing developers to add sound elements to their applications. For instance, playing a local audio file can be done using:

Listing 4
PLAYING AUDIO FILES IN FLUTTER

```
await audioPlayer.play('assets/sample.mp3');
```

4) video_player: *video_player* is used to play videos within an application, providing full control over the playback. Developers can easily control video actions such as play, pause, and seek. For example, to start video playback:

Listing 5
PLAYING VIDEOS IN FLUTTER

```
_controller.play();
```

5) cached_network_image: *cached_network_image* loads and caches images from the internet to accelerate image rendering and save data. This is particularly useful for applications that frequently download images from the internet, such as news or social media apps. For instance, to display an image from a URL:

Listing 6
CACHING NETWORK IMAGES IN FLUTTER

```
CachedNetworkImage(imageUrl:
    'https://example.com/image.jpg'),
```

C. UI Widgets

This subsection provides an overview of Flutter's widgets and the specific widgets adopted for the *Multimedia App* learning topic.

1) Types of Widgets: Widgets serve as the fundamental components of the application's user interface, encompassing both structural elements and stylistic nuances. They are inherently immutable and each represents a specific part of the UI. Flutter employs a comprehensive widget-based architecture, where nearly every aspect of the interface—such as layout components (like *Rows* and *Columns*), multimedia elements (including *Text*, *Image*, and *SvgPicture*), styling attributes (such as *Colors* and *Themes*), and even alignment and padding—is treated as a widget. This methodology fosters a highly modular and reusable design paradigm, enabling widgets to be nested within each other to create intricate and responsive layouts.

2) Adopted Multimedia Widgets and Third-party Package as Learning Components: In order to broaden students' understanding of multimedia widgets and third-party packages, this learning module incorporates 10 components, detailed in Table I, that students are encouraged to explore and utilize. These tools are essential for assisting students in integrating and developing multimedia applications effectively.

TABLE I
TEN LEARNING COMPONENTS FOR MULTIMEDIA APPLICATIONS

No	**Component**	**Description**
C1	Text Widget	A widget that displays text on the screen.
C2	Google Fonts Style	A tool to apply custom fonts using the `google_fonts` package.
C3	AnimatedOpacity	A widget that animates the opacity of its child widget.
C4	Icon Widget	A widget that displays graphical icons representing actions or objects.
C5	Asset	A resource, like an image or file, used within the app, sourced locally or online.
C6	SvgPicture Widget	A widget that renders SVG (Scalable Vector Graphics) images.
C7	Image Widget	A widget that displays images from local or network sources.
C8	CachedNetworkImage Widget	A widget that loads and caches images from the internet, reducing load times.
C9	Package audioplayers	A package that plays audio files, supporting various formats.
C10	Package video_player	A package that plays video files with support for multiple formats and controls.

D. Application for Student's Assignment

To provide project-based learning for this learning topic, an app for student work has been prepared, namely '*MediaPlayer*'. This application has a display composed of various media such as images (svg and jpeg), audio (mp3), and video (mp4). In addition, this application also has an attractive display such as animation and audio and video interaction. To achieve this, there are several multimedia widgets that are combined to produce 4 application pages, namely the splash screen, home, music player, and video player as shown in Figure 2.

E. Learning Objective

This study highlights the importance of learning mobile Flutter development, particularly multimedia applications, as a crucial skill for students to enhance their ability to create modern applications that integrate various types of media. By providing a self-learning system for this topic, students are expected to become proficient Flutter developers, capable of creating responsive and visually engaging applications. By the end of this learning module, students are expected to achieve the learning outcomes outlined in Table II.

TABLE II
LEARNING OBJECTIVES FOR MULTIMEDIA APPLICATIONS

No	**Description**
LO1	Students can set up the environment and create a new Flutter project.
LO2	Students can prepare the initial code for a simple media player Flutter project.
LO3	Students can implement basic UI components for multimedia, such as a colored Container widget.
LO4	Students can display local SVG assets in a Flutter project using the flutter_svg package.
LO5	Students can implement simple animations using the Animated Container widget.
LO6	Students can style Text widgets using the google_fonts package.
LO7	Students can use the Icon widget.
LO8	Students can display images with the Image widget and use cached_network_image to load images from the internet.
LO9	Students can manipulate formatted string data using extension functions.
LO10	Students can implement audio interactions like play, pause, and stop using the audioplayers package.
LO11	Students can work with audio from both local assets and the internet.
LO12	Students can implement video interactions like play, pause, and stop using the video_player package.
LO13	Students can work with video from both local assets and the internet.

F. Learning Stages

To achieve the learning objective, a systematic step-by-step approach is essential, particularly to build the *MediaPlayer* application. The instructional module consists of six main components, each focusing on a specific aspect of multimedia application development using Flutter, as explained in Table III.

TABLE III
LEARNING STAGES FOR *Multimedia App* TOPIC

No	**Stage**	**Learning Objective**	**Components**
S1	Installing Flutter	LO1	-
S2	Creating a Flutter project	LO2	C2, C5, C6, C8, C9, C10
S3	Implementing splash screen with animation	LO3, LO4, LO5	C3, C5, C6
S4	Display image and text widget	LO4, LO6, LO7, LO8, LO9	C1, C2, C3, C4, C5, C6, C7, C8
S5	Building a music player page with multimedia packages	LO6, LO7, LO8, LO11, LO11	C1, C2, C4, C5, C7, C8, C9
S6	Developing a video player with custom controls	LO5, LO6, LO8, LO12, LO13	C1, C2, C3, C4, C5, C7, C8, C10

G. Test Codes

To assist students in validating their Flutter project code, particularly in the multimedia programming context, we provide automated test files. These test files allow students to verify their implementation against expected outcomes. The following are some key aspects being tested:

1) Testing UI Property Values: In this test, the properties of UI elements are verified to ensure they match the expected values. For example, the following code checks the shape and color of a `Container` decoration. First, the decoration is retrieved from the `Container`, and the test asserts that its shape is `BoxShape.circle`. Next, it checks that the color of the decoration is `null`. If any of these properties are incorrect, a detailed error message will be provided. The `expect` keyword is used to compare the actual value with the expected value and provide feedback if the assertion fails.

Listing 7
PROPERTY TEST

```
final decoration = container.decoration as
    BoxDecoration;
expect(
   decoration.shape,
   equals(BoxShape.circle),
   reason: 'Container decoration shape is not
       BoxShape.circle',
);
expect(
   decoration.color,
   isNull,
   reason: 'Container decoration color is not
       null',
);
```

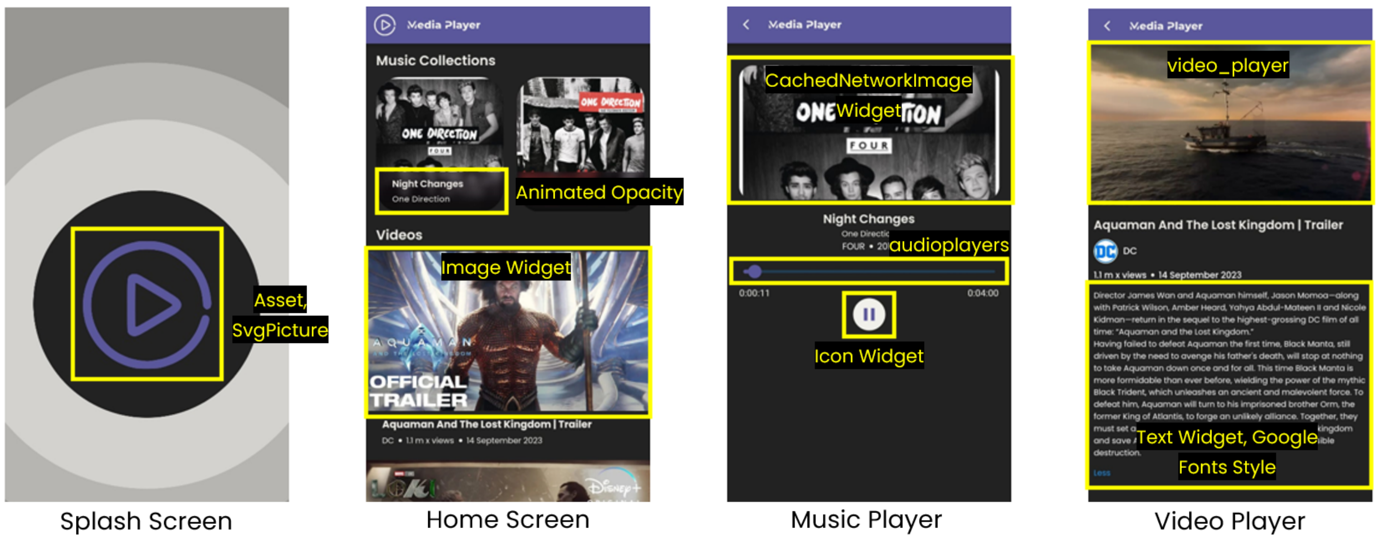


Fig. 2. User interface of *MediaPlayer* application

Keywords:

- **expect**: Compares the actual value to the expected value and throws an error if they don't match.
- **equals**: A matcher used to check if two values are equal.
- **isNull**: A matcher used to check if a value is 'null'.

2) Testing Callback Functions: This test ensures that callback functions, such as those triggered when a button is pressed, perform the expected actions. The following code checks if the `onPressed` function of a custom button widget `ControllButton` is correctly linked to the `playPause` function in the state. First, the `ControllButton` widget is found in the widget tree. Then, the `onPressed` callback is checked to ensure it is connected to the correct function. If the connection is incorrect, appropriate feedback will be provided to guide students in fixing their code.

Listing 8
CALLBACK FUNCTION TEST

```
final icnButton =
    tester.widget<ControllButton>
(icnButtonFinder);
expect(
  icnButton.onPressed.toString(),
  musicPlayerState.playPause.toString(),
  reason: 'ControllButton onPressed should
      call playPause',
);
```

Keywords:

- **widget**: Finds a widget in the widget tree that matches the given finder.

3) Testing State Changes Through Interaction: This test simulates user interaction to ensure that it triggers the correct state changes. For example, the following code tests whether the `playPause` method correctly toggles the 'play' state in a music player. First, the `playPause` method is simulated by calling it. Then, `pumpAndSettle` is used to wait for all animations and state changes to complete, ensuring that the UI has fully updated. After that, the test checks if the `play` state has changed to `true`, as expected. If the state change does not occur as intended, feedback will be provided to help students correct their implementation.

Listing 9
STATE CHANGE TEST

```
await musicPlayerState.playPause();
await tester.pumpAndSettle();
expect(
  musicPlayerState.play,
  isTrue,
  reason:
     'Calling the playPause method for the
        first time will change the play
        state to true (because initial
        state of play is false)',
);
```

Keywords:

- **pumpAndSettle**: Allows the tester to wait for all animations and microtasks to finish, ensuring the UI has stabilized after a state change.
- **isTrue**: A matcher used to verify that a value is 'true'.

H. Automatic Feedback

Feedback is automatically generated from the test results, allowing students to identify and correct issues in their multimedia Flutter projects (*MediaPlayer*). The process of creating feedback through source code validation uses flutter_test as a testing tool that automatically identifies code issues such as bugs, errors, and unexpected behavior by running test code.

As illustrated in the Fig 3 and Fig 4, the Flutter test framework provides detailed feedback to help students identify the issue and make corrections. The case in Fig 3 shows a failure message in the form of *"Expected a Transform widget as child of ClipRRect, found Container."*. This means that the widget that students write as a child of the `ClipRRect` is not a `Transform` (expected), but a `Container` (actual). This mismatch triggers a test failure.

Another example in the case of Fig 4 case, the test was designed to check the `CircleComponent` widget's rendering, particularly focusing on its color. The failure occurred because the expected color of the `Container` decoration was a MaterialColor with the value `Color(0xff2196f3)`, but the actual color was `null`. This mismatch triggers a test failure, and the framework provides feedback indicating that the *"Container decoration color does not match provided color."*. Conversely, the passed result will appear as Fig 5 if the code written is correct.

V. Evaluation

This area strengthens the adequacy of the research by carrying out an evaluation on the Multimedia Application topk to enhance students' independent learning.

A. Evaluation Setup

The evaluation involved 40 students, all of whom had learned the Flutter framework and were familiar with the basics of Flutter projects. They were tasked with progressing through the six learning phases sequentially, focusing on creating a multimedia app, exemplified by the *MediaPlayer* app case study. The students will do this work independently with their own devices, with a deadline of four days for completion.

B. Result of The Collected Application

During the evaluation process, students must upload their MediaPlayer project after completing all learning stages. This is done to review and assess the applications they create. Figure 6 displays a student-built application that is very similar to the example application provided. This similarity is due to the presence of test code (unit and widget tests), which checks each property and widget to meet certain requirements. The only difference is in the order of the property list, which does not affect the appearance of the application as long as the values match the specifications.

C. Topic Completion Time Results

All experimental results are collected after the student evaluation process. Table IV shows that 40 students passed each stage (S1 to S6). This indicates that the tasks were appropriate for the students' abilities, or that the instructional materials were effective. The shortest time recorded to complete the stages varied, with Stage 1 being the fastest (5 minutes) and Stage 6 being the longest (185 minutes), indicating varying complexity or student familiarity with the tasks. However, the average time gives a different picture: Stage 6 appeared to be the most time-consuming (136,3 minutes on average), hinting at its complexity or difficulty, followed by Stages 5 and 4, largely due to the substantial use of widgets and the large number of widgets involved in these stages. Stages 1, 2 and 3 showed much lower average times (14.9, 20,9 and 30,8 minutes respectively), indicating that they were easier.

Based on the analysis of these results, it can be concluded that the length of each stage by students depends on the number of states, widgets or components, and steps in the stages. Where in stage 1 focuses on the software installation process required in the development of the Flutter application where most testers have installed it and there are only 9 practicum steps to initiate the Flutter project. While in stage 6 there are 45 steps to create 3 components and 1 page with a total of 47 widgets and 8 states. In addition, this stage also provides project collection steps with a total of 15 steps.

TABLE IV
Evaluation result in solving time

Result	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
Passed (students)	40	40	40	40	40	40
fastest (minutes)	5	10	15	80	75	100
average (minutes)	15	21,1	30,8	106,3	94,1	136,7
longest (minutes)	30	40	50	160	130	185

D. Task Difficulties

On the topic of completion time results, there are quite diverse results for each stage. Where stages 1 and 2 spend a short processing time of 15 and 21.1 minutes. Stage 3 requires an average processing time of 30.8 minutes. Meanwhile, in stages 4, 5, and 6 there is a significant spike in data where each stage requires an average time of more than 90 minutes. With this in mind, each stage can be grouped as in Table V.

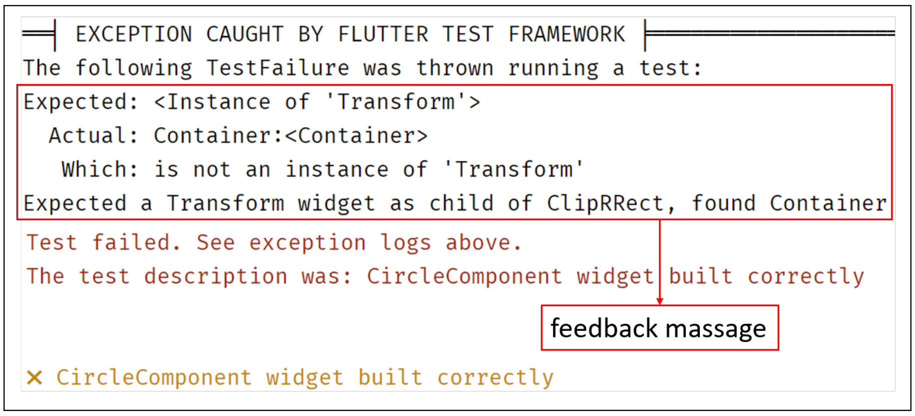
TABLE V
Learning stages complexity

Level	**Required Time (minutes)**	**Learning Stages**
Easy	$0 < \text{time} \leq 30$	S1, S2
Medium	$30 < \text{time} \leq 60$	S3
Hard	$\text{time} > 60$	S4, S5, S6

E. Feedback on Multimedia Application Topics

Feedback was collected from students, revealing overwhelmingly positive responses and some constructive suggestions. Positive feedback accounted for 85% of all responses. Examples of positive feedback include:

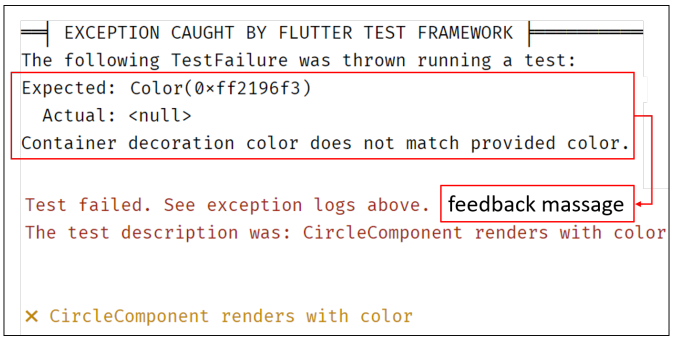
- *"This learning module is well-organized and easy to follow for beginners like me."*
- *"The tutorial is very helpful and builds an understanding of making a media player. The explanation of the component diagram first makes the tutorial clearer and more directed, allowing students to work seriously without copying and pasting."*
- *"The tutorial is clear and easy to understand, I can create my own application."*



```
═╡ EXCEPTION CAUGHT BY FLUTTER TEST FRAMEWORK ╞═══════════════
The following TestFailure was thrown running a test:
Expected: <Instance of 'Transform'>
  Actual: Container:<Container>
   Which: is not an instance of 'Transform'
Expected a Transform widget as child of ClipRRect, found Container
 Test failed. See exception logs above.
 The test description was: CircleComponent widget built correctly

 ✕ CircleComponent widget built correctly
```

Fig. 3. Automatic feedback for fail test

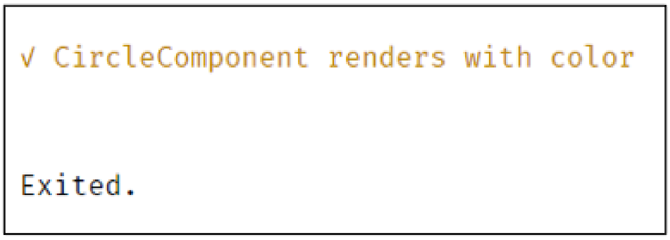


```
═╡ EXCEPTION CAUGHT BY FLUTTER TEST FRAMEWORK ╞═══════════
The following TestFailure was thrown running a test:
Expected: Color(0xff2196f3)
  Actual: <null>
Container decoration color does not match provided color.

 Test failed. See exception logs above.
 The test description was: CircleComponent renders with color

 ✕ CircleComponent renders with color
```

Fig. 4. Automatic feedback for fail test



```
√ CircleComponent renders with color


Exited.
```

Fig. 5. Passed test output

- *"I learned a lot, especially about using state and creating widgets for separation purposes."*
- *"The media player project was very helpful in understanding multimedia in Flutter, especially the part about building the user interface and the description when verifying the code, which was very helpful in the problem-solving process."*

Additionally, constructive suggestions made up 15% of the feedback. Examples include:

- *"Sufficient, but the steps in the guide could be further clarified."*
- *"Perhaps an explanation about the code could be added."*

VI. Conclusion

This paper presents the development of a self-learning module called *Multimedia App* within the *Flutter Programming Learning Assistance System (FPLAS)*, incorporating test-driven development methods to provide automated assistance in a project-based learning environment. The focus on integrating multimedia resources in mobile application development requires a deep understanding of multimedia widgets and third-party packages. To this end, the module includes 10 key components, including multimedia widgets and third-party packages, for students to explore and utilize, essential for effectively integrating and developing multimedia applications. An evaluation involving 40 students confirmed the system's effectiveness, with feedback highlighting the clarity and utility of tasks, though some suggested improvements like clearer guides and more code explanations.

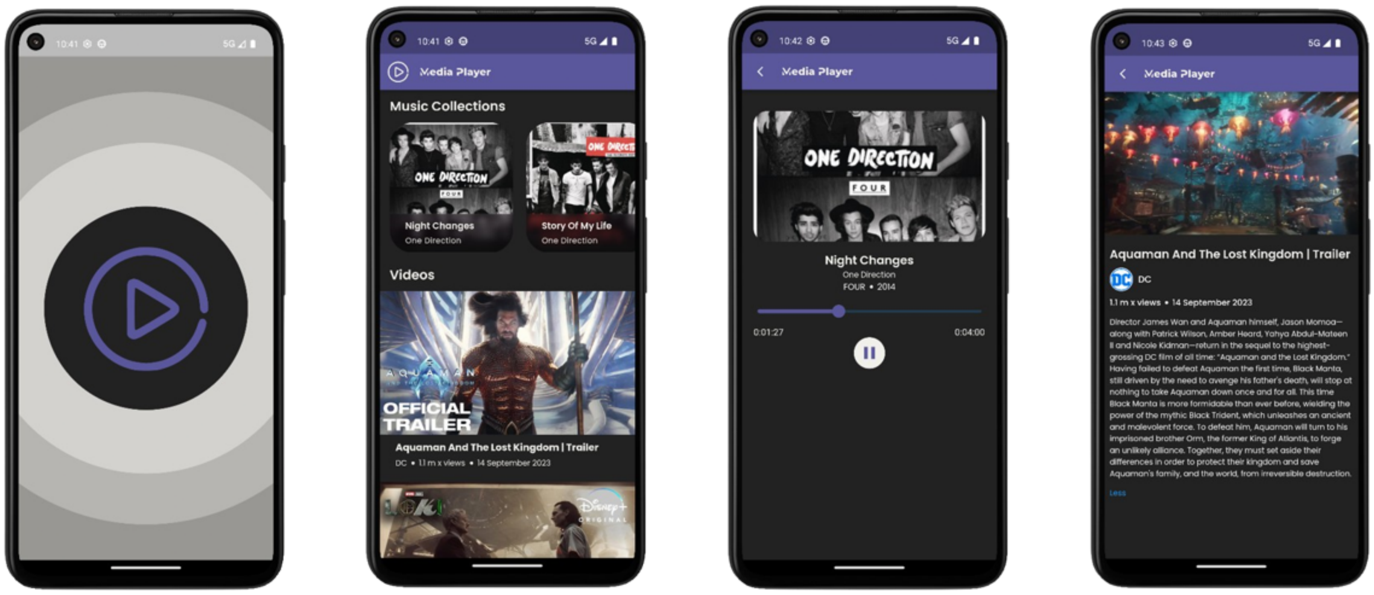
By integrating SDL principles with automated testing and comprehensive instructional design, this study enhanced self-directed learning in mobile programming using Flutter. The instructional modules, covering various stages from installation to video player implementation, provided structured guidance and immediate feedback. The evaluation showed successful completion of each module stage, with largely positive feedback from participants. This approach facilitated immediate error detection and deeper comprehension, illustrating its potential for broader application in programming education. Future efforts will focus on refining these modules and incorporating additional multimedia elements to further enrich the learning experience.

Acknowledgment

This research is a component of the iCLOP (Intelligent Computer Assisted Programming Learning Platform) research initiative. We would like to express our gratitude to Applied Science University in Bahrain (https://www.asu.edu.bh/) for offering free registration to the students who authored this academic paper.

References

[1] L. Pancani, E. Preti, and P. Riva, "The psychology of smartphone: The development of the smartphone impact scale (sis)," *Assessment*, vol. 27, no. 6, pp. 1176–1197, 2020.



Director James Wan and Aquaman himself, Jason Momoa—along with Patrick Wilson, Amber Heard, Yahya Abdul-Mateen II and Nicole Kidman—return in the sequel to the highest-grossing DC film of all time: "Aquaman and the Lost Kingdom." Having failed to defeat Aquaman the first time, Black Manta, still driven by the need to avenge his father's death, will stop at nothing to take Aquaman down once and for all. This time Black Manta is more formidable than ever before, wielding the power of the mythic Black Trident, which unleashes an ancient and malevolent force. To defeat him, Aquaman will turn to his imprisoned brother Orm, the former King of Atlantis, to forge an unlikely alliance. Together, they must set aside their differences in order to protect their kingdom and save Aquaman's family, and the world, from irreversible destruction.

Fig. 6. Sample student application interface

[2] J. Skalka, M. Drlik, L. Benko, J. Kapusta, J. C. Rodríguez del Pino, E. Smyrnova-Trybulska, A. Stolinska, P. Svec, and P. Turcinek, "Conceptual framework for programming skills development based on microlearning and automated source code evaluation in virtual learning environment," *Sustainability*, vol. 13, no. 6, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/6/3293

[3] S. Y. Ameen and D. Y. Mohammed, "Developing cross-platform library using flutter," *European Journal of Engineering and Technology Research*, vol. 7, no. 2, p. 18–21, Mar. 2022. [Online]. Available: https://www.ej-eng.org/index.php/ejeng/article/view/2740

[4] S. Avsec and M. Jagiełło-Kowalczyk, "Investigating possibilities of developing self-directed learning in architecture students using design thinking," *Sustainability*, vol. 13, no. 8, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/8/4369

[5] H.-C. Jeong and W.-Y. So, "Difficulties of online physical education classes in middle and high school and an efficient operation plan to address them," *International Journal of Environmental Research and Public Health*, vol. 17, no. 19, 2020. [Online]. Available: https://www.mdpi.com/1660-4601/17/19/7279

[6] J. Skalka and M. Drlík, "Development of automatic source code evaluation tests using grey-box methods: A programming education case study," *IEEE Access*, vol. 11, pp. 106 772–106 792, 2023.

[7] A. J. Rohm, M. Stefl, and N. Ward, "Future proof and real-world ready: The role of live project-based learning in students' skill development," *Journal of Marketing Education*, vol. 43, no. 2, pp. 204–215, 2021. [Online]. Available: https://doi.org/10.1177/02734753211001409

[8] F. Ahammad, "Self-directed learning: A core concept in adult education," *The Online Journal of Distance Education and e-Learning*, vol. 11, no. 3, 2023.

[9] M. Zhu, C. J. Bonk, and M. Y. Doo, "Self-directed learning in moocs: Exploring the relationships among motivation, self-monitoring, and self-management," *Educational Technology Research and Development*, vol. 68, pp. 2073–2093, 2020.

[10] I. Mekterović, L. Brkić, B. Milašinović, and M. Baranović, "Building a comprehensive automated programming assessment system," *IEEE Access*, vol. 8, pp. 81 154–81 172, 2020.

[11] E. Barra, S. Lopez-Pernas, A. Alonso, J. F. Sanchez-Rada, A. Gordillo, and J. Quemada, "Automated assessment in programming courses: A case study during the covid-19 era," *Sustainability*, vol. 12, no. 18, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/18/7451

[12] Y. W. Syaifudin, M. Mentari, A. B. Kaswar, Y. Ariyanto, U. D. Rosiani, and D. Puspitasari, "Implementation and evaluation of self-learning topic for sqlite integration in flutter programming learning assistance system," in *10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2023, pp. 589–594.

[13] D. Staegemann, M. Volk, M. Perera, C. Haertel, M. Pohl, C. Daase, and K. Turowski, "A literature review on the challenges of applying test-driven development in software engineering," *Complex Systems Informatics and Modeling Quarterly*, vol. 2022, 2022.

[14] Y. W. Syaifudin, A. S. Hatjrianto, N. Funabiki, D. Y. Liliana, A. B. Kaswar, and U. Nurhasan, "An implementation of automatic dart code verification for mobile application programming learning assistance system using flutter," in *2022 International Conference on Electrical and Information Technology (IEIT)*, 2022.

[15] S. M. Tampubolon and T. Raharjo, "Unveiling the benefits and challenges of test-driven development in agile: A systematic literature review," *Indonesian Journal of Computer Science*, vol. 13, no. 2, 2024.

[16] S. Combéfis, "Automated code assessment for education: Review, classification and perspectives on techniques and tools," *Software*, vol. 1, no. 1, pp. 3–30, 2022. [Online]. Available: https://www.mdpi.com/2674-113X/1/1/2

[17] J. C. Paiva, J. P. Leal, and A. Figueira, "Automated assessment in computer science education: A state-of-the-art review," *ACM Trans. Comput. Educ.*, vol. 22, no. 3, jun 2022. [Online]. Available: https://doi.org/10.1145/3513140

[18] O. C. Falode, K. Dome, E. J. Chukwuemeka, and M. E. Falode, "Development of an interactive mobile application for learning undergraduate educational technology concepts," *International Journal of Professional Development, Learners and Learning*, vol. 4, no. 1, p. ep2204, 2022.

[19] A. S. Gomoll, B. Hillenburg, and C. E. Hmelo-Silver, ""i have never had a pbl like this before": On viewing, reviewing, and co-design," *Interdisciplinary Journal of Problem-based Learning*, vol. 14, no. 1, 2020.

[20] 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, 2020.

[21] A. M. Sattar, P. Soni, M. K. Ranjan, A. Kumar, C. Sahu, S. Saxena, and P. Chaudhari, "Accelerating cross-platform development with flutter framework," *Journal of Open Source Developements*, vol. 10, no. 2, 2023.

[22] M. Wali and L. Ahmad, "Computer assisted learning (cal): A learning support system solution," *Webology*, vol. 18, no. 1, Apr 2021.

[23] Y. W. Syaifudin, D. Y. Liliana, A. B. Kaswar, T. Fatmawati, N. Funabiki, and I. Siradjuddin, "Generating automated assistance mechanism in android programming self-learning system using automatic testing tools," in *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2023, pp. 445–450.

[24] R. Rosnelly, "Rancang bangun aplikasi game edukasi untuk pembe-

lajaran bahasa inggris berbasis android," *Jurnal Mahasiswa Fakultas Teknik dan Ilmu Komputer*, vol. 12, no. 2, Jul. 2022.