

Paper ESTIC Alexander

by Alexander Setiawan

Submission date: 06-Jan-2023 10:56PM (UTC+0700)

Submission ID: 1989239459

File name: Paper_ESTIC_Alexander_2022.pdf (1.47M)

Word count: 3087

Character count: 15524

Virtual Tour Museum Application Using Flutter Framework

Alexander Setiawan¹⁾ Andreas Handoyo²⁾ Silvia Rostianingsih³⁾ Clarissa Angelia⁴⁾
Wendy Santoso⁵⁾, and Jeremy Dion⁶⁾

Author Affiliations

^{1,2,3,4,5} Informatics Department, Faculty of Industrial Technology, Petra Christian University Indonesia

Author Emails

alexander@petra.ac.id¹ handoyo@petra.ac.id² silvia@petra.ac.id³

Abstract. According to the Kamus Besar Bahasa Indonesia (KBBI), a museum's function is to protect, develop, utilize collections, and communicate them to the public. The public often knows the word museum itself as a historical place or a place that has high cultural value. Usually, museums are also used as study tour places for students in Indonesia. In 2020, Indonesia has 439 museums spread across the country. However, not all of these museums can be visited by the public due to distance and time limitations. Therefore, media such as a mobile application is needed so that people can visit the museum virtually without being limited by distance and time. In its creation, the Virtual Tour Museum application will implement a framework using the Dart programming language. This application will also be useful as a means of promotion and introduction of Indonesian culture and history to the public, especially the younger generation through the Virtual Museum media.

Keywords: Application, Flutter, Dart, Museum, Figma, Laravel, Virtual Tour.

INTRODUCTION

Museum is a place to maintain, research, communicate, and exhibit cultural heritage. Museums are often used as places for tourism and educational facilities by the public. In Indonesia, there are around 439 museums spread from Sabang to Merauke. These museums store a wealth of knowledge about Indonesian history and culture that should be preserved and introduced to the public, especially the younger generation. Even though the number of museums in Indonesia is relatively large, in fact not all museums can be visited by the public due to distance and time limitations. Therefore, we need media that can help connect the public with these museums. The media is in the form of a Virtual Tour Museum application where people can visit the museum virtually. It is hoped that this application can be useful as a means of promotion and introduction of Indonesian culture and history to the public, especially the younger generation. The Virtual Tour Museum application was created by implementing the Dart programming language with the framework Flutter.

Flutter as Mobile Application Framework

Flutter is a framework made by Google that is used to create mobile applications, either Android or iOS, websites and desktops [2]. Flutter itself has been used by various large companies such as Google, Alibaba.com, and Tencent because it can save time and effort by simply requiring one codebase to develop applications on various platforms. The advantage of Flutter compared to others is hot reload where when changing application code, application changes can be seen immediately without the need to re-run the program so that it helps debug the program faster [1]. Applications created with Flutter are written in the Dart programming language. The Dart language has many features in common with other modern languages such as Kotlin and Swift and can be compiled into JavaScript code. The Dart language is designed to be familiar with other programming languages so that it is easy for those who already understand other programming languages and for those who are just starting their journey as a developer.

THE PROCESS OF MAKING THE MOBILE APPLICATION

UI/UX Application Design

The Virtual Tour Museum application was designed using Figma. Figma was chosen because of its ease in making prototype design views and application flows. In addition, Figma users can collaborate in terms of designing the appearance of the application so that it makes the design process easier. The design of this application takes a minimalist and elegant theme, so it is hoped that it will further highlight the museum which is the main feature of this application. The appearance of this application is divided into several frames such as Home, Museum Details, Explore, Bookmark, and Profile. Each frame has animations and interactions so that it can clarify the flow of the application. Later, application users will be asked to log in first to use the full features of the application. Following are some of the Virtual Tour Museum application designs:

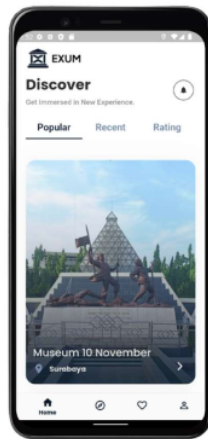


FIGURE 1. Home page display.

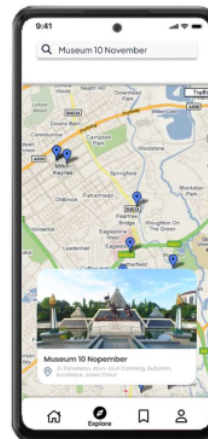


FIGURE 2. Explore page view.

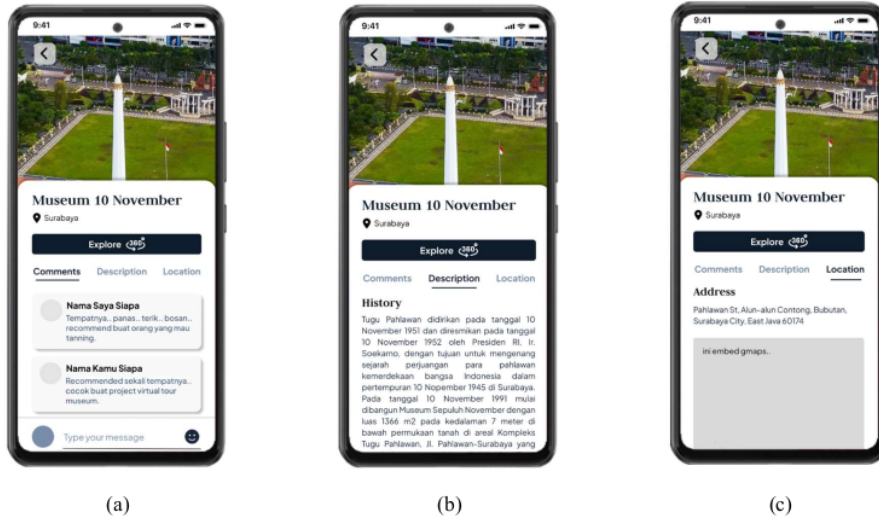


FIGURE 3. Museum Details page display. (a) Comments uploaded by users. (b) Description of the museum. (c) Map the location and address of the museum.

Fig. 1 consists of several sections, at the very top there is the logo of the application. Then underneath there is a slogan to invite users to know more about this Virtual Tour Museum application. Next to the slogan, there is a notification button where the user will be directed to the notification page to see incoming notifications. After that, at the bottom, there is a tab bar that displays three options, namely popular, recent, and rating. The popular tab will display a list of currently popular museums, the recent tab displays a list of recently uploaded museums and the rating displays museums based on ratings.

Fig. 2 uses the Google Maps API to retrieve museum locations around the user and display them on a map. Then using the data obtained, the application will also display a card containing information about the museums. At the top of the map, there is a search bar that users can use to find the museum they want.

Fig. 3 displays the detailed pages of each museum. On this page, there is information such as the name of the museum and its location. After that, there is a tab bar that contains three options, namely comments, description, and location. In the comments tab users can see comments related to the museum, users can also write comments in the input field that is available at the bottom. Then the description tab is used to display a brief description of the museum and the location tab to view the location of the museum.

Flutter Project Setup

Design that has been created and then realized using Flutter. Work on this project is done using Visual Studio Code (VS Code) as the IDE and Android Studio as the emulator. VS Code was chosen as the IDE for writing Flutter code due to its low minimum specifications compared to Android Studio. Flutter projects can be run using google chrome or the emulator from Android Studio if they have been previously downloaded. When the project is run, by default VS Code will use google chrome to display the Flutter project that has been created. Before starting to create a Flutter project in VS Code, several requirements must be met first.

The first step is to download the VS Code on each laptop or computer. VS Code is a source code editor made by Microsoft. One of the advantages of this application is that it can use various programming languages such as Flutter and Dart by using the extensions provided in the application. VS Code can be downloaded on the main website then choose the installer according to the device's operating system. After downloading the installer VS Code, Fig. 4 shows the setup that needs to be done to use the application. After completing the setup, open the VS Code application. In the menu section on the left, select the Extensions menu with the checkered logo. After opening the Extensions menu in VS Code, Fig. 5 and Fig. 6 show extensions of Flutter and Dart that need to be installed. Press the install button then wait for the installation to finish.

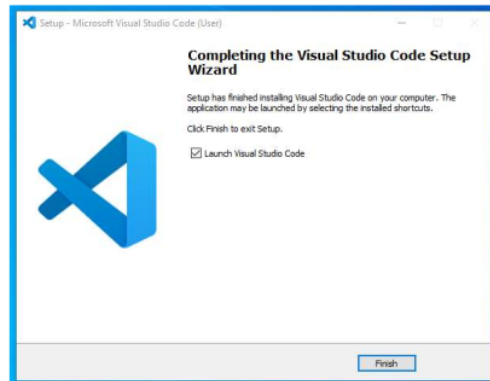


FIGURE 4. Visual Studio Code (VS Code) setup wizard display.

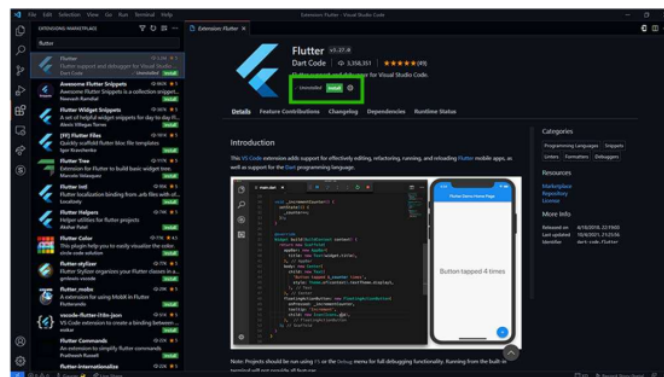


FIGURE 5. Installing the Flutter extension on VS Code.

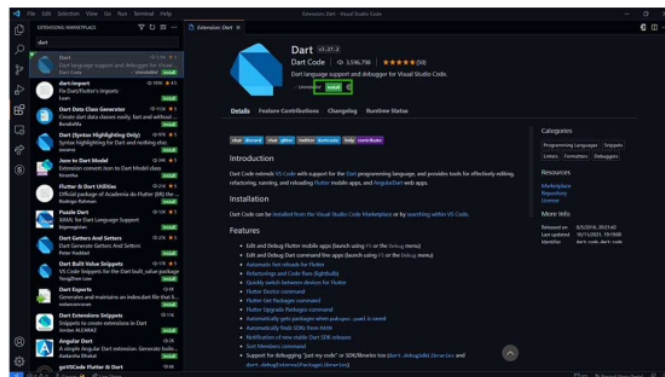


FIGURE 6. Installing the Dart extension on VS Code.

After installing Flutter and Dart, it is necessary to check whether the device used meets the conditions needed to use the Flutter framework. The trick is to run the Flutter Doctor in the VS Code terminal or in the command prompt of the device you are using. In Fig. 7 it can be seen that Flutter and VS Code have been successfully installed on the device indicated by a green check icon on the leading text.

```

Doctor summary (to see all details, run Flutter doctor -v):
[✓] Flutter (Channel stable, 2.2.3, on macOS 11.6 20G165 darwin-arm, locale ja-JP)
[✓] Android toolchain - develop for Android devices (Android SDK version 29.0.2)
[✓] Xcode - develop for iOS and macOS
[✓] Chrome - develop for the web
[!] Android Studio (not installed)
[✓] IntelliJ IDEA Community Edition (version 2019.1.3)
[✓] VS Code (version 1.61.1)
[✓] Connected device (1 available)

```

FIGURE 7. Display of Flutter Doctor in VS Code terminal or Command Prompt.

The second step is to install git. Git is one of the Version Control Systems (VCS). VCS is a system that can store and manage snapshots of changes to source code. Besides Git, there are Subversion, Mercurial, and CVS. The advantages of VCS are being able to track changes to the source code, making it easier to collaborate if there is a project that is being worked on by more than one person, and being able to find out who made the changes and when the changes were made. In this application project, Git needs to be installed because the Virtual Tour Museum application project will be uploaded on Github to make the process easier. Installation can be started by downloading the installer on the main Git website and then following the setup process.

After installing VS Code and git, the next step is to install Android Studio. Android Studio is required to be a virtual emulator when the Flutter program is running. Follow the steps required in Android Studio to maximize the facilities that will be used when working on Flutter projects. Just like the previous step, the installer can be downloaded on the main website. After that, Fig. 8 shows the setup view from Android Studio. Follow the application setup process and then open the Android Studio application to install the API and emulator as needed. If Android Studio and the emulator have been installed successfully, run the flutter doctor in the VS Code terminal or Command Prompt. Make sure the program runs properly without any problems.



FIGURE 8. View of the Android Studio setup wizard.

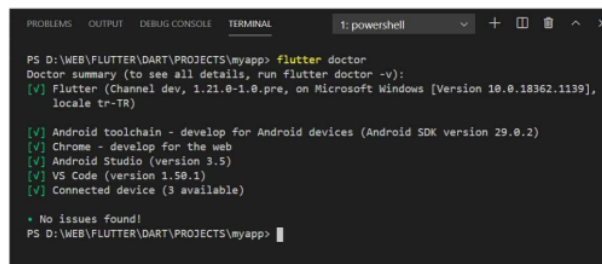


FIGURE 9. What Flutter Doctor looks like after installing Android Studio.

After all the steps above have been successfully carried out, the final step is to create a new Flutter project in VS Code. Open VS Code then presses the shortcut `ctrl + shift + p` to create a new project. When the keyword "flutter" is typed into the input box, a display similar to Figure 10 will appear. After that, select the "Flutter: New Project" option then press enter then select the "Application" option. When pressed later the user will be asked to select the folder path to put the project and then enter the project name. At this stage, a new Flutter project is successfully created and ready to start developing Flutter applications on VS Code.

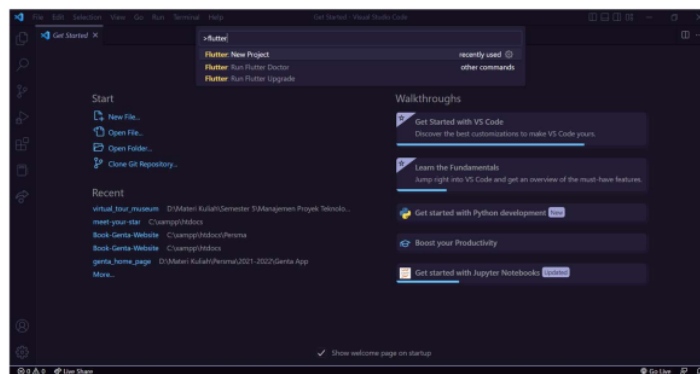


FIGURE 10. Initialize Flutter project in VS Code.

Developing Application using Flutter and Dart

In a Flutter project, by default when the project is first initialized, VS Code will generate the files and folders required for Flutter application development. There are several folders generated by VS Code as shown in Figure 11. The following describes the Flutter project directory structure:

1. android folder > contains source code for Android applications
2. folder ios > contains source code for iOS applications
3. lib folder > contains source code Dart, this folder is used to store application code
4. test folder > contains Dart source code for application testing
5. .gitignore > Git file
6. .metadata > file containing project metadata that is automatically generated
7. .packages > file containing the package path address created by pub
8. flutter_app.iml > XML file containing project description
9. pubspec.lock > file containing library or package versions. This file is generated by pub. Its function is to lock the version of the
10. pubspec.yaml package > a file that contains information about the project and required libraries
11. README.md > a markdown file that contains an explanation of the source code

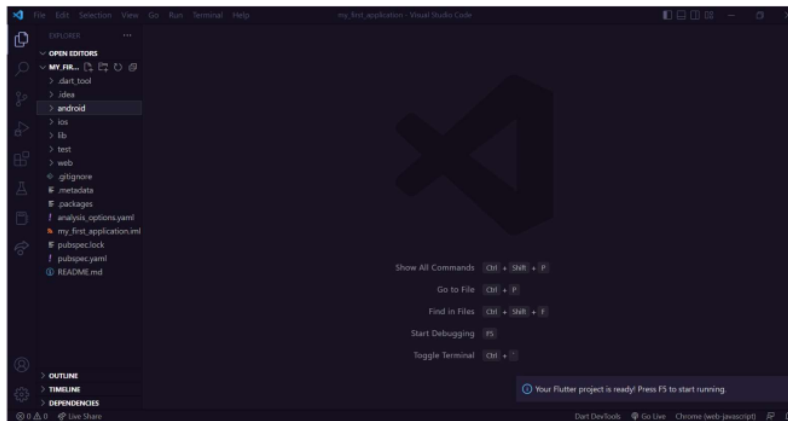


FIGURE 11. Flutter project directory structure.

In the lib folder, there is a main.dart file which is generated automatically by VS Code. The main.dart file will be read first when the application project is run. In Fig. 12, the Flutter code structure is divided into 3 sections:

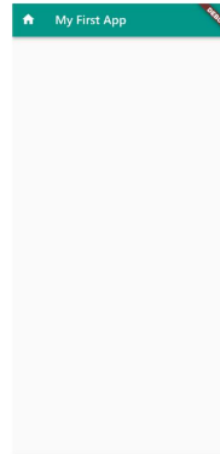
1. Import Section > Place to declare or import the required libraries in the application.
2. Main Section > The main function of the application will be the entry point. This function will be executed the first time when the application is started.
3. Widgets section > Used to create widgets. A Flutter app consists of an array of widgets. Widgets are elements such as Buttons, Text, Layout, Images, and so on. Each widget has various properties and can be modified according to needs. In Fig. 12, the widget used is the AppBar widget which is the main widget of the application. Inside the AppBar widget, there are properties such as backgroundColor, leading and title. The title property functions to display text on the app bar of the application. The title property accepts only the Text widget as its parameter. The Widget Text is used to display text. In addition, the backgroundColor property is used to change the background color of the app bar, the leading property to display the icon before the title, and many more.


```

1 import 'package:flutter/material.dart';
2
3 Run [Debug] Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({Key? key}) : super(key: key);
10
11 @override
12 Widget build(BuildContext context) {
13   return MaterialApp(
14     title: 'Flutter Application',
15     theme: ThemeData(
16       primarySwatch: Colors.blue,
17     ), // ThemeData
18     home: Scaffold(
19       appBar: AppBar(
20         backgroundColor: Colors.teal,
21         leading: Icon(Icons.home),
22         title: Text("My First App"), // AppBar
23       ), // Scaffold
24     ); // MaterialApp
25 }

```

(a)



(b)

FIGURE 12. Flutter application code structure. (a) Application code. (b) App view.

Implementation of the Google Maps API in Flutter

The Virtual Tour Museum application will use Google Maps to retrieve museum locations around the user and display a map Figure 2. To display Google Maps in an application, an API key that is generated directly by Google is required. Each application has a different API key, here are the steps for implementing the Google Maps API in Flutter.

1. Google Maps SDK Registration

On console.cloud.google.com > API & Services > Enabled API & Services, make sure to enable the three APIs listed in Figure 13. The Places API will be used to fetch the required place data from Google Maps. The Maps SDK for Android is required to display Google Maps in Android-based applications, and the Maps SDK for iOS is required to display Google Maps in iOS-based applications.

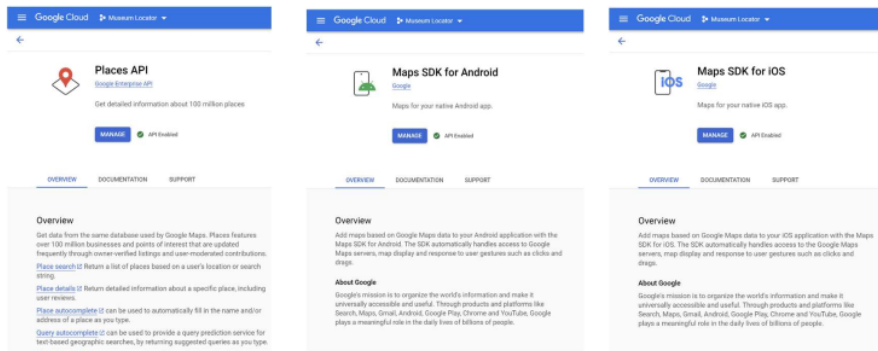


FIGURE 13. API required for initialization of Google Maps. (a) Places API. (b) Maps SDK for Android. (c) Maps SDK for iOS.

Go to `console.cloud.google.com > credentials > create credentials > API key` to create a new API key. Install the restriction API key for the three APIs above. This API key will later be used to display Google Maps when the application is running.

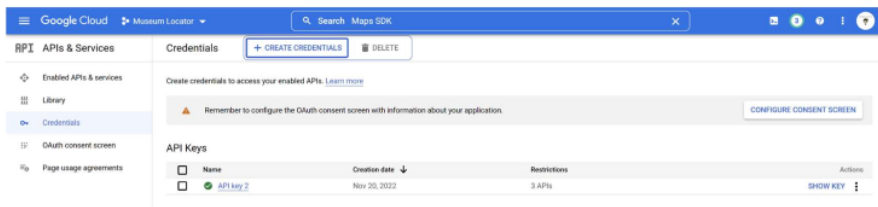


FIGURE 14. Credential view on Google cloud console.

2. Integrating Google Maps API with Flutter

a. Android Application

Add meta-data and permissions as in Fig. 15 in the flutter application folder > android > app > src > AndroidManifest.xml. This metadata is useful for displaying Google Maps when the application is running. Meanwhile, permissions are added to display permission requests when you want to retrieve user location data when the application is running.

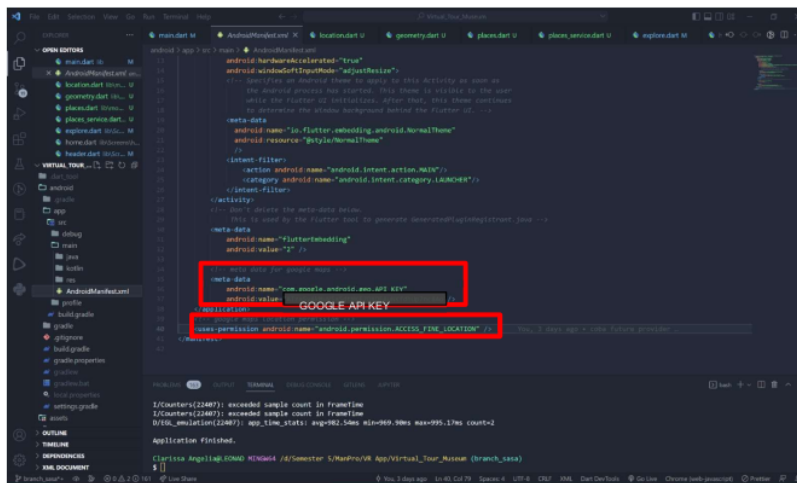


FIGURE 15. Display AndroidManifest.xml

b. iOS

Add the program as shown in Fig. 16 in the flutter application folder > ios > Runner > AppDelegate.swift and the flutter application folder > ios > Runner > Info.plist. The program in AppDelegate.swift is useful for displaying Google Maps when the application is running. Meanwhile, the program in Info.plist is useful for displaying permission requests when you want to retrieve user location data when the application is running.

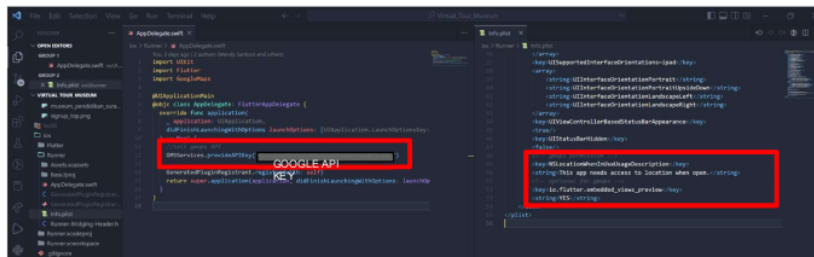


FIGURE 16. Display AppDelegate.swift and Info.plist

3. Adding Dependencies In pubspec.yaml

Add a list of dependencies as shown in Fig. 17 in the pubspec.yaml file in the dependencies section.

```
#google maps dependency
google_maps_flutter: ^2.2.1
geolocator: ^9.0.2
provider: ^6.0.4
url_launcher: ^6.1.7
http: ^0.13.5
flutter_rating_bar: ^4.0.1
location: ^4.4.0
```

FIGURE 17. Google Maps dependency list

Using the Laravel Framework as an Application Database

After the application display is realized using Flutter, the next step is to connect it to the database. Flutter itself is capable of receiving and sending data to the server in JSON form. Therefore an API is needed that connects the frontend that has been made with the backend which will later process the input provided by the user. Laravel itself is a framework that maximizes the use of PHP which functions to connect the database and the frontend that has been created. By using tools "Composer" and "Artisan", the quality of the backend of this application is further accelerated because Composer is useful for downloading libraries as additional features in applications such as Google reCaptcha. Meanwhile, Artisan is useful for databases easily forming the necessary

The database that we created consists of several tables, such as the user table, the data_museum table, and the Comments table. The user table is useful for storing some important user information such as name, username, password, as well as role held. The role itself is divided into 2, namely Admin and User. table data_museum will store the data needed on the Museum Details page. Such as the name of the museum, the history of the museum, the address and location of the museum, and also the link to access virtual reality from the museum. For the virtual reality itself, it was created using the 3DVista application which combines several 360 photos from the museum into a virtual reality with the output of a link to access it. The Comments table will later be connected to the data_museum by taking the id from data_museum to differentiate comments in each museum.

CONCLUSION

The Virtual Tour Museum application has the goal of making it easier for Indonesian people to visit museums without being hindered by distance and time. In addition, this application is an educational tool and introduces museums in Indonesia. With the 360 virtual tour feature, users can easily look around the museum virtually. Making this application uses 3DVista to create 3D virtual tours and frameworks for applications. Later the 3D virtual tour generated from 3DVista will be included in the Flutter application so that many people can easily use it because there are Android and iOS versions available thanks to the cross-platform framework Flutter

ACKNOWLEDGMENTS

² We thank the Institute for Research and Community Service involved in making this idea for giving us the opportunity to implement it. Moreover, we are also grateful to the Institute for Research and Community Service that has reimbursed all the costs we spent on this project.

REFERENCES

1. A. Salim and RRA Gamawanto, *J. Inf. Commun. Technol.* 8 261-281 (2021)
2. A. Tashildar, N. Shah, R. Gala, T. Giri and P. Chavhan, *J. Mod. Eng. Technol.* 02 1262 (2020)
3. K. Wasilewski and W. Zabierowski, *J. Sci. Technol.* 21 3324 (2021)
4. K. Jonathan, R. Intan, and Liliana, *Comput. Phys.* 10 293-298 (2022)

Paper ESTIC Alexander

ORIGINALITY REPORT

2%

SIMILARITY INDEX

0%

INTERNET SOURCES

2%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

1

Marco L. Napoli. "Beginning Flutter®", Wiley, 2019

Publication

1%

2

Umar, Hasdinar, Sabaruddin Rahman, A. Yasir Baeda, and Syerly Klara. "Identification of Coastal Problem and Prediction of Coastal Erosion Sedimentation in South Sulawesi", Procedia Engineering, 2015.

Publication

1%

Exclude quotes On

Exclude matches < 1%

Exclude bibliography On