Virtual Tour Museum Application Using Flutter Framework

Alexander Setiawan[1)] Andreas Handojo[2)] Silvia Rostianingsih[3)] Clarissa Angelia[4)] Wendy Santoso[5)], and Jeremy Dion[6)]

Author Affiliations
[1,2,3,4,5] *Informatics Department, Faculty of Industrial Technology, Petra Christian University Indonesia*

Author Emails
alexander@petra.ac.id[1] *handojo@petra.ac.id*[2] *silvia@petra.ac.id*[3]

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:

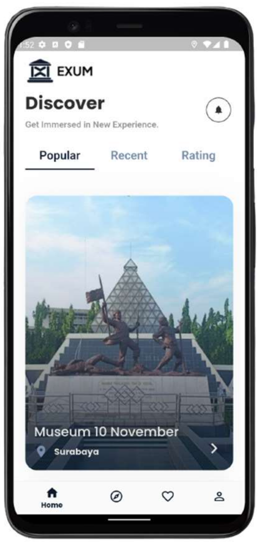


FIGURE 1. Home page display.

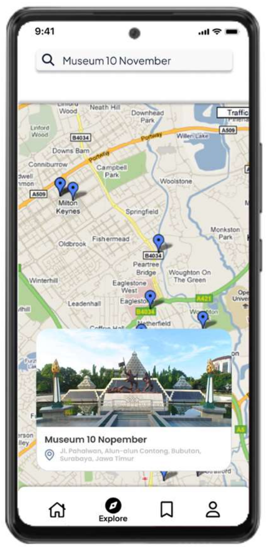
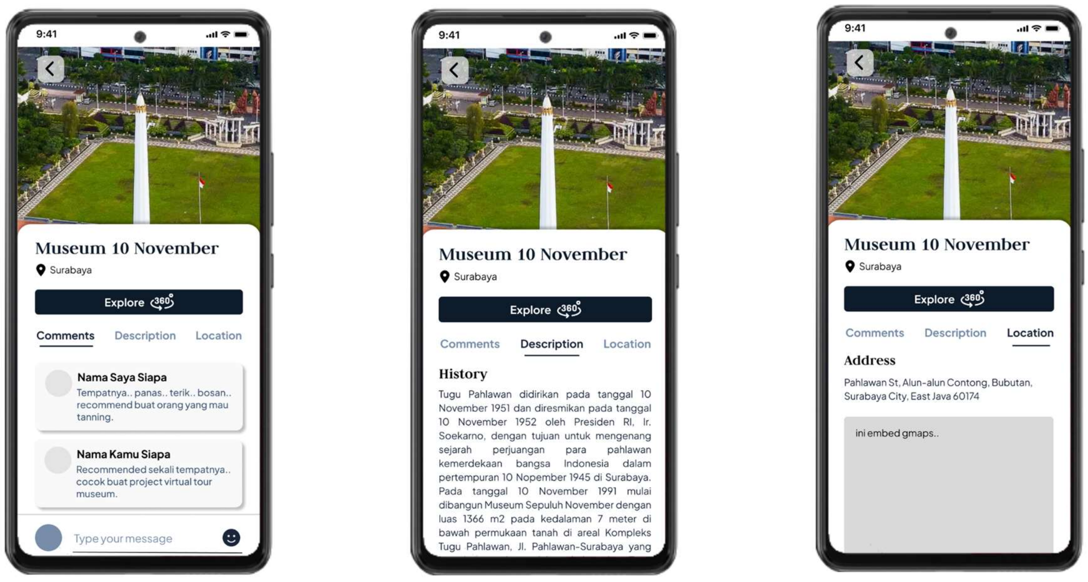

9:41
Museum 10 November
Milton Keynes
Museum 10 Nopember
Jl. Pahlawan, Alun-alun Contong, Bubutan, Surabaya, Jawa Timur
Explore

FIGURE 2. Explore page view.



Nama Saya Siapa
Tempatnya.. panas.. terik.. bosan.. recommend buat orang yang mau tanning.

Nama Kamu Siapa
Recommended sekali tempatnya.. cocok buat project virtual tour museum.

History

Tugu Pahlawan didirikan pada tanggal 10 November 1951 dan diresmikan pada tanggal 10 November 1952 oleh Presiden RI, Ir. Soekarno, dengan tujuan untuk mengenang sejarah perjuangan para pahlawan kemerdekaan bangsa Indonesia dalam pertempuran 10 Nopember 1945 di Surabaya. Pada tanggal 10 November 1991 mulai dibangun Museum Sepuluh November dengan luas 1366 m2 pada kedalaman 7 meter di bawah permukaan tanah di areal Kompleks Tugu Pahlawan, Jl. Pahlawan-Surabaya yang

Address

Pahlawan St, Alun-alun Contong, Bubutan, Surabaya City, East Java 60174

ini embed gmaps..

(a) (b) (c)

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.

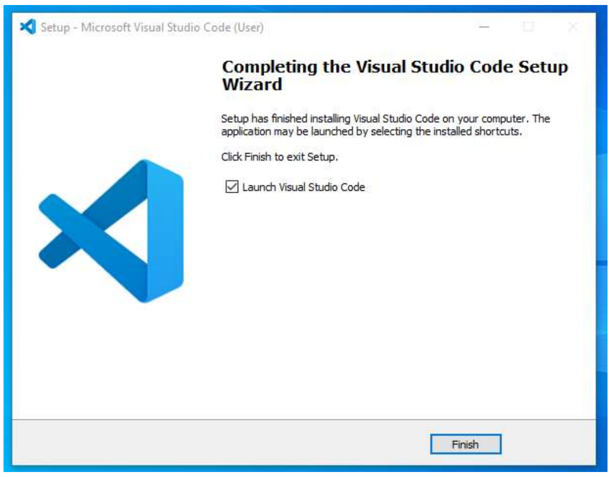
Fig. 2 uses the Google Maps API to retrieve museum locations around the user and display them on a map. Then using the data obtained, the application will also display a card containing information about the museums. At the top of the map, there is a search bar that users can use to find the museum they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



Setup - Microsoft Visual Studio Code (User)

Completing the Visual Studio Code Setup Wizard

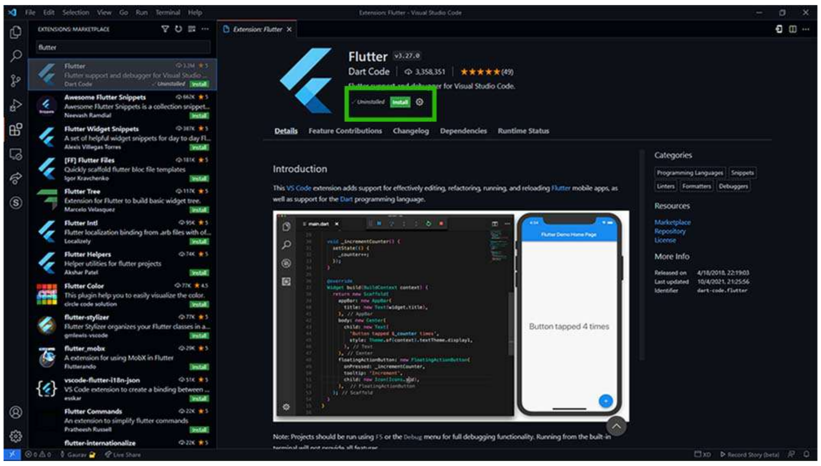
Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☑ Launch Visual Studio Code

Finish

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



File Edit Selection View Go Run Terminal Help
EXTENSIONS: MARKETPLACE
Extension: Flutter - Visual Studio Code
Extension: Flutter

flutter

Flutter — 3.3M ★5 — Flutter support and debugger for Visual Studio ... — Dart Code — Uninstalled — Install
Awesome Flutter Snippets — 662K ★5 — Awesome Flutter Snippets is a collection snippet... — Neevash Ramdial — Install
Flutter Widget Snippets — 387K ★5 — A set of helpful widget snippets for day to day Fl... — Alexis Villegas Torres — Install
[FF] Flutter Files — 181K ★5 — Quickly scaffold flutter bloc file templates — Igor Kravchenko — Install
Flutter Tree — 117K ★5 — Extension for Flutter to build basic widget tree. — Marcelo Velasquez — Install
Flutter Intl — 95K ★5 — Flutter localization binding from .arb files with of... — Localizely — Install
Flutter Helpers — 74K ★5 — Helper utilities for flutter projects — Akshar Patel — Install
Flutter Color — 77K ★4.5 — This plugin help you to easily visualize the color. — circle code solution — Install
flutter-stylizer — 77K ★5 — Flutter Stylizer organizes your Flutter classes in a... — gmlewis-vscode — Install
flutter_mobx — 29K ★5 — A extension for using MobX in Flutter — Flutterando — Install
vscode-flutter-i18n-json — 51K ★5 — VS Code extension to create a binding between ... — esskar — Install
Flutter Commands — 22K ★5 — An extension to simplify flutter commands — Pratheesh Russell — Install
flutter-internationalize — 22K ★5

Flutter v3.27.0
Dart Code | 3,358,351 | ★★★★★ (49)
Flutter support and debugger for Visual Studio Code.
Uninstalled Install

Details Feature Contributions Changelog Dependencies Runtime Status

Introduction

This VS Code extension adds support for effectively editing, refactoring, running, and reloading Flutter mobile apps, as well as support for the Dart programming language.

Flutter Demo Home Page
Button tapped 4 times

Note: Projects should be run using F5 or the Debug menu for full debugging functionality. Running from the built-in terminal will not provide all features.

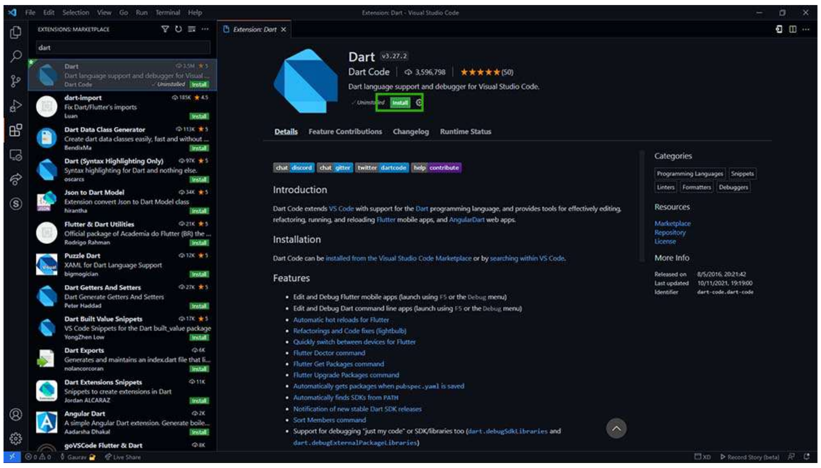
Categories
Programming Languages | Snippets | Linters | Formatters | Debuggers

Resources
Marketplace
Repository
License

More Info

Released on	4/18/2018, 22:19:03
Last updated	10/4/2021, 21:25:56
Identifier	dart-code.flutter

FIGURE 5. Installing the Flutter extension on VS Code.


Dart v3.27.2

Dart Code | 3,596,798 | ★★★★★ (50)

Dart language support and debugger for Visual Studio Code.

Details · Feature Contributions · Changelog · Runtime Status

Introduction

Dart Code extends VS Code with support for the Dart programming language, and provides tools for effectively editing, refactoring, running, and reloading Flutter mobile apps, and AngularDart web apps.

Installation

Dart Code can be installed from the Visual Studio Code Marketplace or by searching within VS Code.

Features

- Edit and Debug Flutter mobile apps (launch using F5 or the Debug menu)
- Edit and Debug Dart command line apps (launch using F5 or the Debug menu)
- Automatic hot reloads for Flutter
- Refactorings and Code fixes (lightbulb)
- Quickly switch between devices for Flutter
- Flutter Doctor command
- Flutter Get Packages command
- Flutter Upgrade Packages command
- Automatically gets packages when `pubspec.yaml` is saved
- Automatically finds SDKs from PATH
- Notification of new stable Dart SDK releases
- Sort Members command
- Support for debugging "just my code" or SDK/libraries too (`dart.debugSdkLibraries` and `dart.debugExternalPackageLibraries`)

Categories: Programming Languages, Snippets, Linters, Formatters, Debuggers

Resources: Marketplace, Repository, License

More Info

Released on	8/5/2016, 20:21:42
Last updated	10/11/2021, 19:19:00
Identifier	dart-code.dart-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.

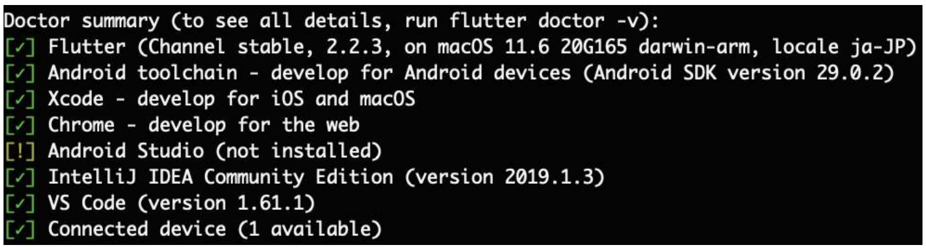
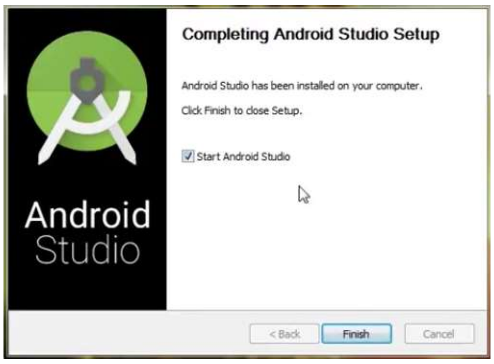

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.2.3, on macOS 11.6 20G165 darwin-arm, locale ja-JP)
[✓] Android toolchain - develop for Android devices (Android SDK version 29.0.2)
[✓] Xcode - develop for iOS and macOS
[✓] Chrome - develop for the web
[!] 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.



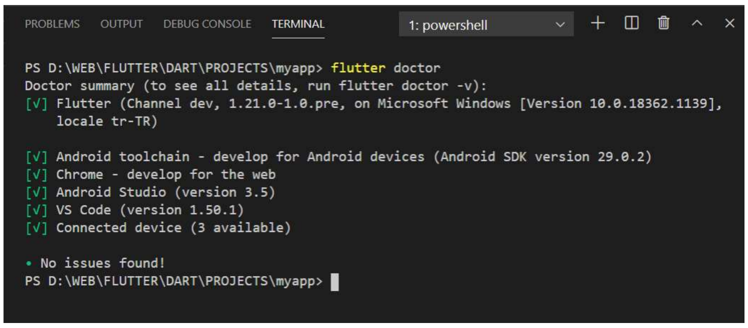
Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

☑ Start Android Studio

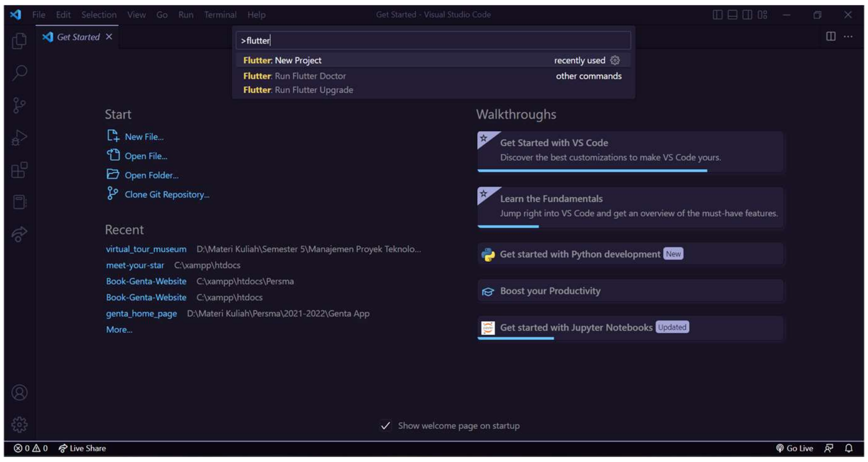
FIGURE 8. View of the Android Studio setup wizard.



```
PS D:\WEB\FLUTTER\DART\PROJECTS\myapp> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel dev, 1.21.0-1.0.pre, on Microsoft Windows [Version 10.0.18362.1139],
    locale tr-TR)

[√] Android toolchain - develop for Android devices (Android SDK version 29.0.2)
[√] Chrome - develop for the web
[√] Android Studio (version 3.5)
[√] VS Code (version 1.50.1)
[√] Connected device (3 available)

• No issues found!
PS D:\WEB\FLUTTER\DART\PROJECTS\myapp> █
```

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.



>flutter

Flutter: New Project   recently used

Flutter: Run Flutter Doctor   other commands

Flutter: Run Flutter Upgrade

Start

New File...

Open File...

Open Folder...

Clone Git Repository...

Recent

virtual_tour_museum   D:\Materi Kuliah\Semester 5\Manajemen Proyek Teknolo...

meet-your-star   C:\xampp\htdocs

Book-Genta-Website   C:\xampp\htdocs\Persma

Book-Genta-Website   C:\xampp\htdocs

genta_home_page   D:\Materi Kuliah\Persma\2021-2022\Genta App

More...

Walkthroughs

Get Started with VS Code
Discover the best customizations to make VS Code yours.

Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.

Get started with Python development New

Boost your Productivity

Get started with Jupyter Notebooks Updated

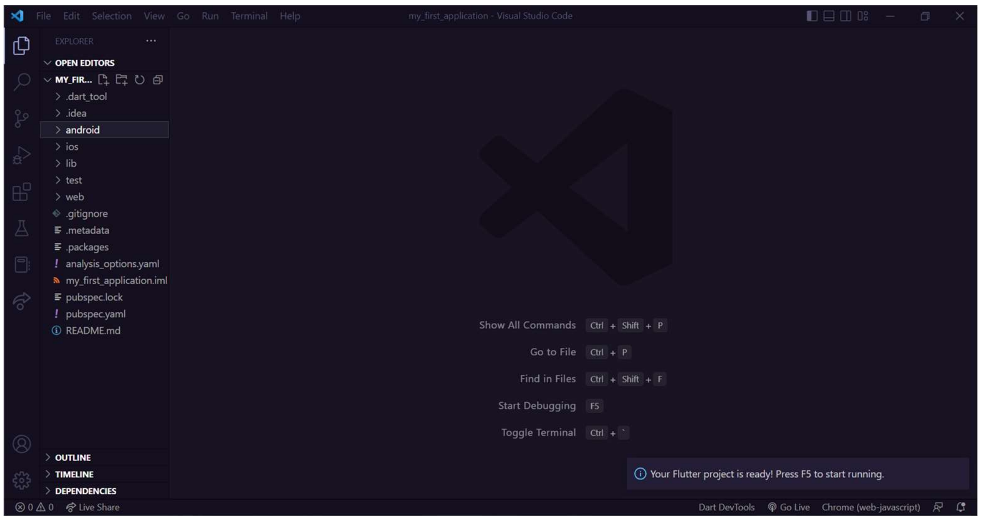
✓ Show welcome page on startup

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

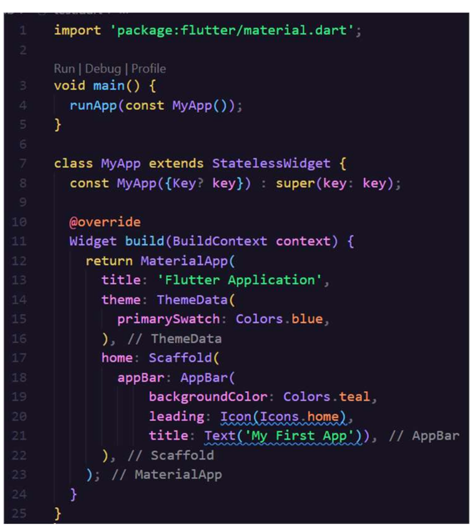


Your Flutter project is ready! Press F5 to start running.

FIGURE 11. Flutter project directory structure.

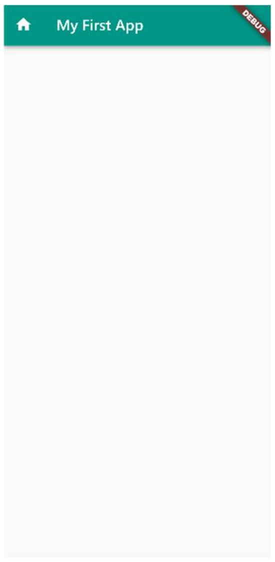
In the lib folder, there is a main.dart file which is generated automatically by VS Code. The main.dart file will be read first when the application project is run. In Fig. 12, the Flutter code structure is divided into 3 sections:

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.



```
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Application',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: Scaffold(
        appBar: AppBar(
            backgroundColor: Colors.teal,
            leading: Icon(Icons.home),
            title: Text('My First App')), // AppBar
      ), // Scaffold
    ); // MaterialApp
  }
}
```

(a)



(b)

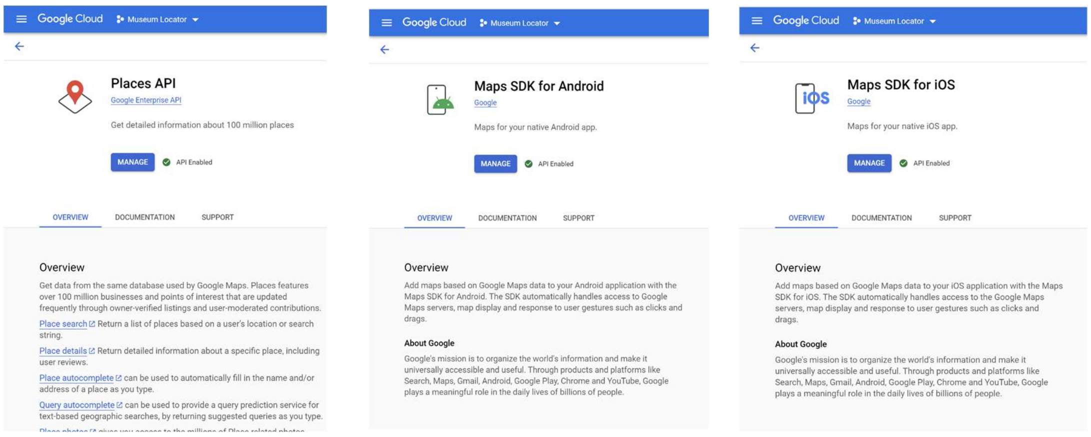
FIGURE 12. Flutter application code structure. (a) Application code. (b) App view.

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.



Places API
Google Enterprise API
Get detailed information about 100 million places

Overview
Get data from the same database used by Google Maps. Places features over 100 million businesses and points of interest that are updated frequently through owner-verified listings and user-moderated contributions.
Place search Return a list of places based on a user's location or search string.
Place details Return detailed information about a specific place, including user reviews.
Place autocomplete can be used to automatically fill in the name and/or address of a place as you type.
Query autocomplete can be used to provide a query prediction service for text-based geographic searches, by returning suggested queries as you type.

Maps SDK for Android
Google
Maps for your native Android app.

Overview
Add maps based on Google Maps data to your Android application with the Maps SDK for Android. The SDK automatically handles access to Google Maps servers, map display and response to user gestures such as clicks and drags.

About Google
Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

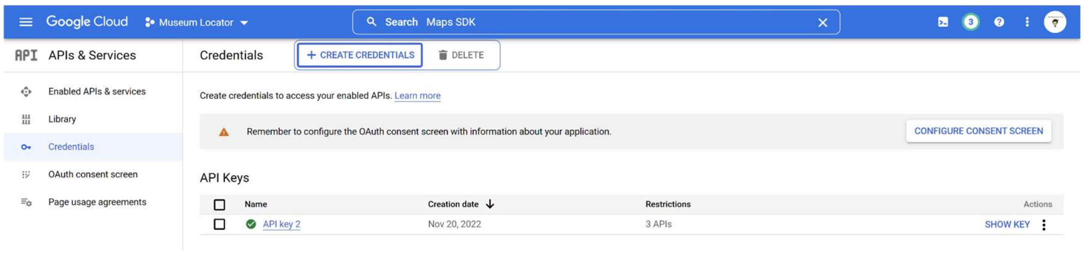
Maps SDK for iOS
Google
Maps for your native iOS app.

Overview
Add maps based on Google Maps data to your iOS application with the Maps SDK for iOS. The SDK automatically handles access to the Google Maps servers, map display and response to user gestures such as clicks and drags.

About Google
Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

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.



Credentials

Create credentials to access your enabled APIs. Learn more

Remember to configure the OAuth consent screen with information about your application.

API Keys

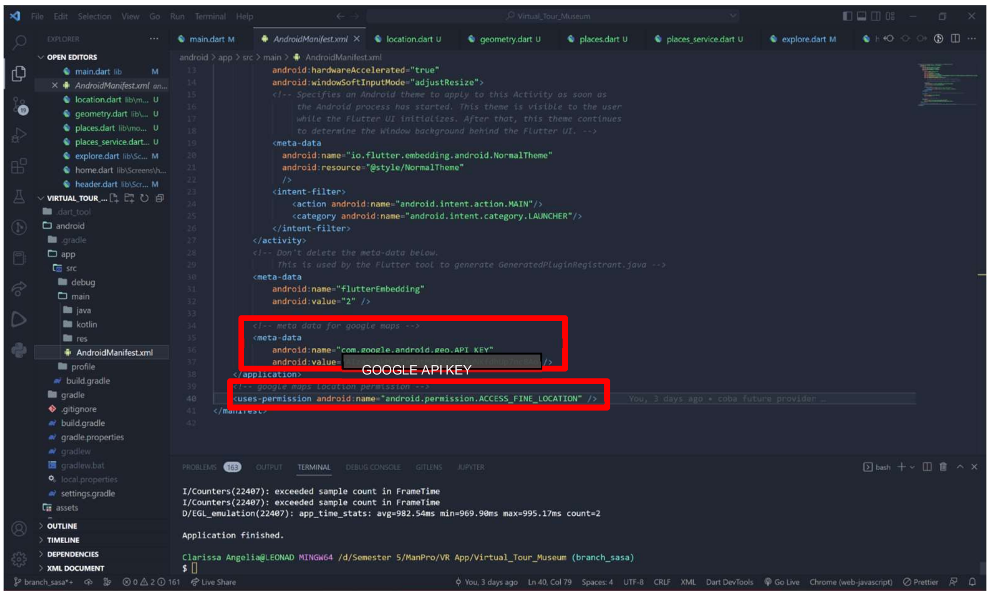
Name	Creation date	Restrictions	Actions
API key 2	Nov 20, 2022	3 APIs	SHOW KEY

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.



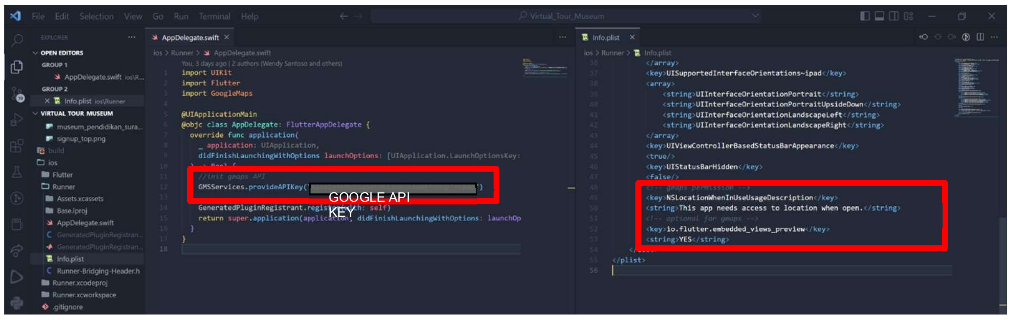
```
android:hardwareAccelerated="true"
android:windowSoftInputMode="adjustResize">
<!-- Specifies an Android theme to apply to this Activity as soon as
     the Android process has started. This theme is visible to the user
     while the Flutter UI initializes. After that, this theme continues
     to determine the Window background behind the Flutter UI. -->
<meta-data
  android:name="io.flutter.embedding.android.NormalTheme"
  android:resource="@style/NormalTheme"
  />
<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<!-- Don't delete the meta-data below.
     This is used by the Flutter tool to generate GeneratedPluginRegistrant.java -->
<meta-data
    android:name="flutterEmbedding"
    android:value="2" />

<!-- meta data for google maps -->
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value=[redacted]/>
</application>
<!-- google maps location permission -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
</manifest>
```

```
I/Counters(22407): exceeded sample count in FrameTime
I/Counters(22407): exceeded sample count in FrameTime
D/EGL_emulation(22407): app_time_stats: avg=982.54ms min=969.90ms max=995.17ms count=2

Application finished.

Clarissa Angelia@LEONAD MINGW64 /d/Semester 5/ManPro/VR App/Virtual_Tour_Museum (branch_sasa)
$ 
```

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.



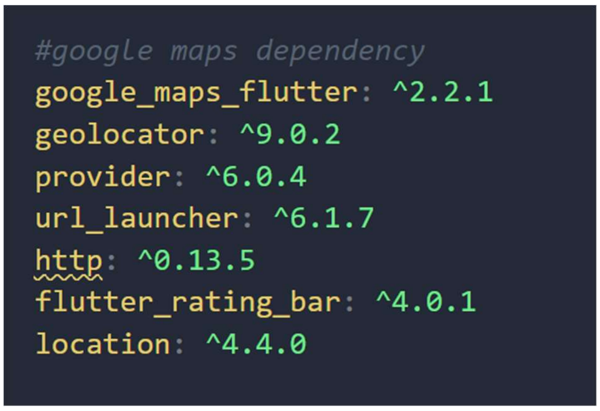
```swift
import UIKit
import Flutter
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey:
  ) -> Bool {
    //init gmaps API
    GMSServices.provideAPIKey([redacted])

    GeneratedPluginRegistrant.register(with: self)
    return super.application(application, didFinishLaunchingWithOptions: launchOp
  }
}
```

```xml
            </array>
            <key>UISupportedInterfaceOrientations~ipad</key>
            <array>
                <string>UIInterfaceOrientationPortrait</string>
                <string>UIInterfaceOrientationPortraitUpsideDown</string>
                <string>UIInterfaceOrientationLandscapeLeft</string>
                <string>UIInterfaceOrientationLandscapeRight</string>
            </array>
            <key>UIViewControllerBasedStatusBarAppearance</key>
            <true/>
            <key>UIStatusBarHidden</key>
            <false/>
            <!-- gmaps permission -->
            <key>NSLocationWhenInUseUsageDescription</key>
            <string>This app needs access to location when open.</string>
            <!-- optional for gmaps -->
            <key>io.flutter.embedded_views_preview</key>
            <string>YES</string>
        </dict>
</plist>
```

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

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)