

Installing Flutter

For Windows

Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

[flutter_windows_v1.7.8+hotfix.4-stable.zip](#)

For other release channels, and older builds, see the [SDK archive](#) page.

2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\src\flutter`; do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges).
3. Locate the file `flutter_console.bat` inside the `flutter` directory. Start it by double-clicking.

Update your path

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the `PATH` environment variable:

- From the Start search bar, type 'env' and select **Edit environment variables for your account**.
- Under **User variables** check if there is an entry called **Path**:
 - If the entry does exist, append the full path to `flutter\bin` using `;` as a separator from existing values.
 - If the entry does not exist, create a new user variable named `Path` with the full path to `flutter\bin` as its value.

Note that you have to close and reopen any existing console windows for these changes to take effect.

Run `flutter doctor`

From a console window that has the Flutter directory in the path (see above), run the following command to see if there are any platform dependencies you need to complete the setup:

```
C:\src\flutter>flutter doctor
```

This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

For Mac OS

Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

```
flutter_macos_v1.7.8+hotfix.4-stable.zip
```

For other release channels, and older builds, see the [SDK archive](#) page.

2. Extract the file in the desired location, for example:

```
$ cd ~/development  
$ unzip ~/Downloads/flutter_macos_v1.7.8+hotfix.4-stable.zip
```

3. Add the `flutter` tool to your path:

```
$ export PATH="$PATH:$(pwd)/flutter/bin"
```

This command sets your `PATH` variable for the *current* terminal window only. To permanently add Flutter to your path, see [Update your path](#).

Run flutter doctor

Run the following command to see if there are any dependencies you need to install to complete the setup (for verbose output, add the `-v` flag):

```
$ flutter doctor
```

This command checks your environment and displays a report to the terminal window. The Dart SDK is bundled with Flutter; it is not necessary to install Dart separately. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

Update your path

You can update your `PATH` variable for the current session only at the command line, as shown in [Get the Flutter SDK](#). You'll probably want to update this variable permanently, so you can run `flutter` commands in any terminal session.

The steps for modifying this variable permanently for all terminal sessions are machine-specific. Typically you add a line to a file that is executed whenever you open a new window. For example:

1. Determine the directory where you placed the Flutter SDK. You will need this in Step 3.
2. Open (or create) `$HOME/.bash_profile`. The file path and filename might be different on your machine.

3. Add the following line and change `[PATH_TO_FLUTTER_GIT_DIRECTORY]` to be the path where you cloned Flutter's git repo

```
$ export PATH="$PATH:[PATH TO FLUTTER GIT DIRECTORY]/flutter/bin"
```

4. Run `source $HOME/.bash_profile` to refresh the current window.
5. Verify that the `flutter/bin` directory is now in your PATH by running:

```
$ echo $PATH
```

Basic Understanding of Flutter

Flutter comes with a suite of powerful basic widgets, of which the following are very commonly used:

- [Text](#): The [Text](#) widget lets you create a run of styled text within your application.
- [Row](#), [Column](#): These flex widgets let you create flexible layouts in both the horizontal ([Row](#)) and vertical ([Column](#)) directions. Its design is based on the web's flexbox layout model.
- [Stack](#): Instead of being linearly oriented (either horizontally or vertically), a [Stack](#) widget lets you stack widgets on top of each other in paint order. You can then use the [Positioned](#) widget on children of a [Stack](#) to position them relative to the top, right, bottom, or left edge of the stack. Stacks are based on the web's absolute positioning layout model.
- [Container](#): The [Container](#) widget lets you create a rectangular visual element. A container can be decorated with a [BoxDecoration](#), such as a background, a border, or a shadow. A [Container](#) can also have margins, padding, and constraints applied to its size. In addition, a [Container](#) can be transformed in three dimensional space using a matrix.
- The [Scaffold](#) is good enough to create a general purpose mobile application and contains almost everything you need to create a functional and responsive app.
- The AppBar and Body No Scaffold application is complete without using these two as they are the bare minimum widgets that need to be used to create a minimal Material Design
- [Bottom Navigation Bar](#) as the name suggests, the bottom navigation bar, just like the **AppBar** is a horizontal strip at the bottom of the screen. It can have multiple items and can use text labels, icons or a combination of both
-

TabController class

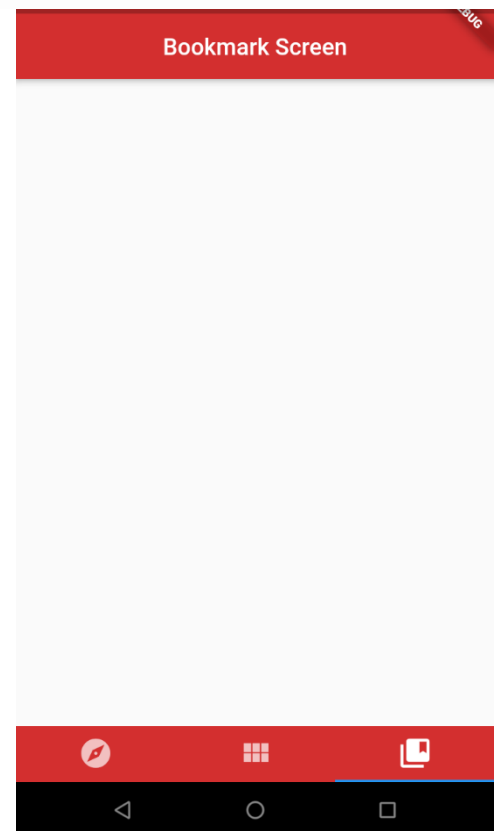
Coordinates tab selection between a [TabBar](#) and a [TabBarView](#).

The [index](#) property is the index of the selected tab and the [animation](#) represents the current scroll positions of the tab bar and the tab bar view. The selected tab's index can be changed with [animateTo](#).

A stateful widget that builds a [TabBar](#) or a [TabBarView](#) can create a [TabController](#) and share it directly.

```
@override
Widget build(BuildContext
context) {

  TabController controller;
  controller = new
  TabController(vsync:
  this,length: 3);
  return new Scaffold(
    bottomNavigationBar: new
    Material(
      color: Colors.red[700],
      child: new
      TabBar(controller:
      controller,tabs:<Tab>[
        new Tab(icon: new
        Icon(Icons.explore,size: 30.0,)),
        new Tab(icon: new
        Icon(Icons.view_module,size: 30.0,)),
        new Tab(icon: new
        Icon(Icons.collections_bookmark,size: 30.0,)),
      ]),
    ),
    body: new TabBarView(controller: controller,
```





تركيب
THE ASSEMBLY
MAKE | SMART | THINGS

```
children: <Widget>[  
    new HomeFeedScreen.HomeFeedScreen(),  
    new CatScreen.CategoriesScreen(),  
    new BookMarkScreen.BookmarkScreen()  
  
], ),  
  
);
```

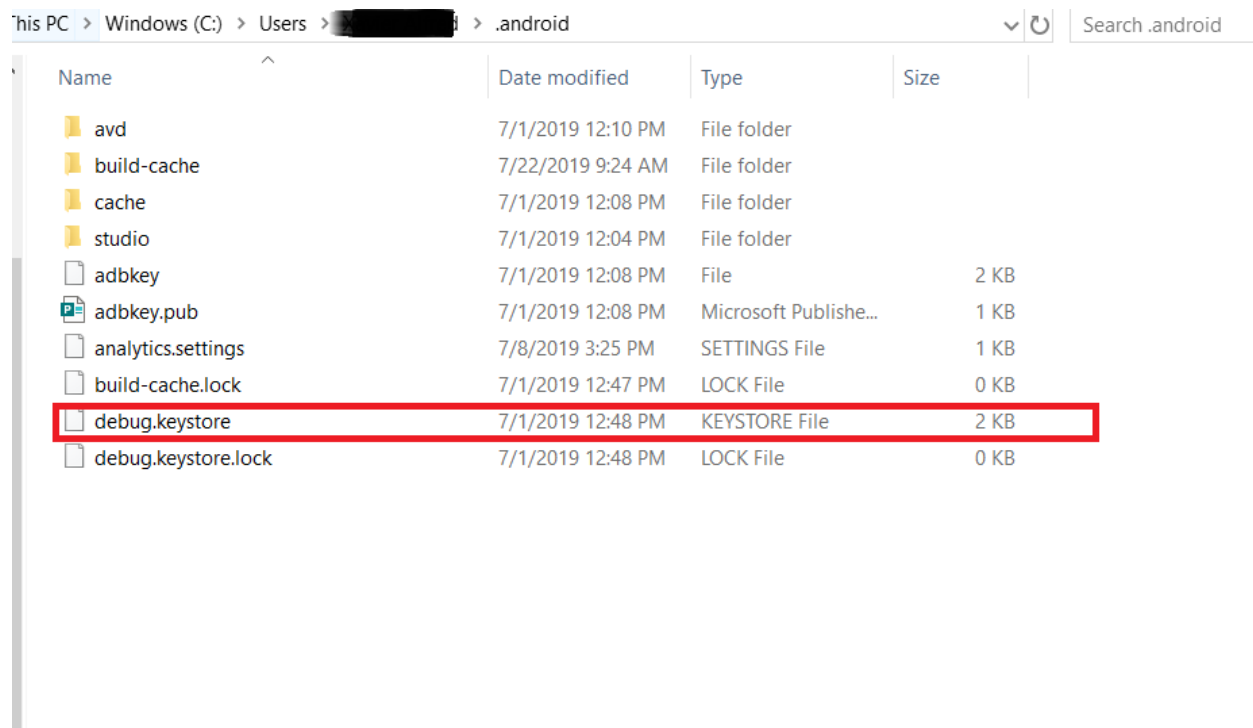
Getting Android Debug KeyStore SHA1 Fingerprint

In android there are two types of keystores. A debug keystore, and a release keystore. Debug keystore is generated automatically when the Android SDK is installed or run for the first time. Release keystore has to be generated manually by the user for each application before release. As it requires private information such as name, password etc. To obtain an Android SHA1 fingerprint from your desired keystore, please follow the steps below

For Windows

Locate the Android *Debug.keystore* which can be found under the directory

Directory: C:\Users\YOUR USERNAME\.android



This PC > Windows (C:) > Users > [REDACTED] > .android					Search .android
Name	Date modified	Type	Size		
avd	7/1/2019 12:10 PM	File folder			
build-cache	7/22/2019 9:24 AM	File folder			
cache	7/1/2019 12:08 PM	File folder			
studio	7/1/2019 12:04 PM	File folder			
adbkey	7/1/2019 12:08 PM	File	2 KB		
adbkey.pub	7/1/2019 12:08 PM	Microsoft Publishe...	1 KB		
analytics.settings	7/8/2019 3:25 PM	SETTINGS File	1 KB		
build-cache.lock	7/1/2019 12:47 PM	LOCK File	0 KB		
debug.keystore	7/1/2019 12:48 PM	KEYSTORE File	2 KB		
debug.keystore.lock	7/1/2019 12:48 PM	LOCK File	0 KB		

Once the file has been located run the following command line on the Command Prompt

```
keytool -list -v -alias androiddebugkey -keystore  
"C:\Users\YOUR USERNAME\.android\debug.keystore"
```

By default, you Android Debug keystore password is “**android**”

```
C:\Users\>keytool -list -v -alias androiddebugkey -keystore "C:\Users\...\android\debug.keystore"
Enter keystore password:
Alias name: androiddebugkey
Creation date: Jul 1, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: C=US, O=Android, CN=Android Debug
Issuer: C=US, O=Android, CN=Android Debug
Serial number: 1
Valid from: Mon Jul 01 12:48:44 GST 2019 until: Wed Jun 23 12:48:44 GST 2049
Certificate fingerprints:
  SHA1: DF:7B:01:95:8D:1B:0C:ED:28:60:F7:61:F2:...
  SHA256: 6F:14:FA:01:26:55:CF:A3:28:2C:29:...
Signature algorithm name: SHA1withRSA
Subject Public Key Algorithm: 1024-bit RSA key
Version: 1

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format
using "keytool -importkeystore -srckeystore C:\Users\...\android\debug.keystore -destkeystore C:\Users\...\android\debug.keystore -deststoretype pkcs12".
```

For Mac and Linux

keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android

For more information follow the full tutorial on <https://www.truicon.com/2015/04/obtaining-sha1-fingerprint-android-keystore/>

Adding Firebase to Flutter App

Step 1: Set up your environment

- [Install Flutter](#) for your specific operating system, including the installation of the following:
 - Flutter SDK
 - Supporting libraries
 - Platform-specific software and SDKs
- Install your preferred [editor or IDE](#), such as Android Studio, IntelliJ, Xcode, or VS Code.
- Open your Flutter app in your preferred editor or IDE.
- iOS development — The app must target iOS 8 or later.
- Android development — The app must target API level 16 (Jelly Bean) or later.
- Set up a device or emulator for running your app.
 - Android development — [Emulators](#) must use an emulator image with Google Play.
- [Sign into Firebase](#) using your Google account.

Step 2: Create a Firebase project

1. In the [Firebase console](#), click **Add project**, then select or enter a **Project name**.
If you have an existing Google Cloud Platform (GCP) project, you can select the project from the **Project name** dropdown menu. Otherwise, enter a new **Project name**.
2. *(Optional)* Edit the **Project ID**.

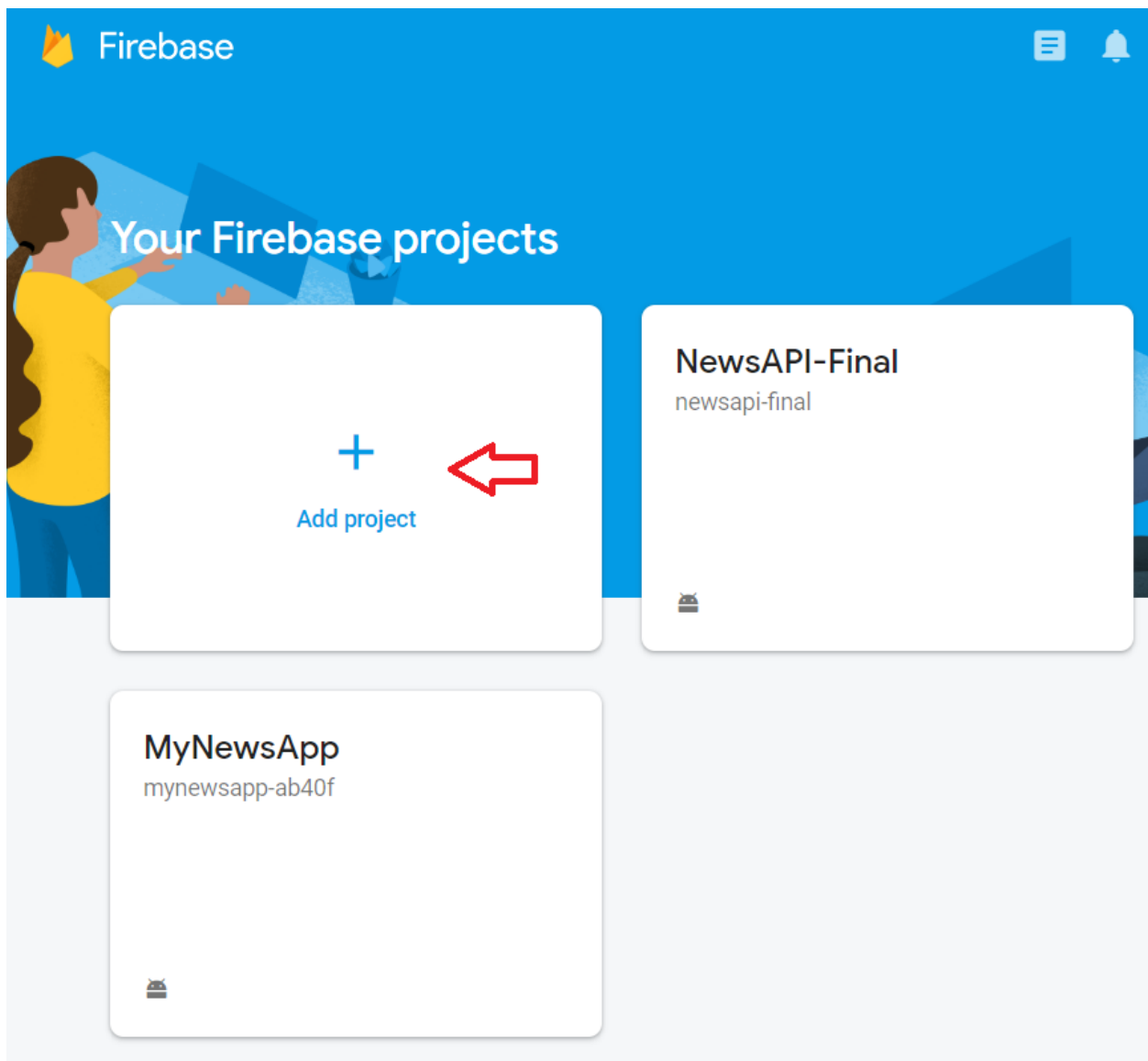
Firebase automatically assigns a unique ID to your Firebase project. Visit [Understand Firebase Projects](#) to learn about how Firebase uses the project ID.

After Firebase provisions resources for your Firebase project, you cannot change your project ID.

To use a specific identifier, you must edit your project ID during this setup step.

3. Follow the remaining setup steps in the Firebase console, then click **Create project** (or **Add Firebase**, if you're using an existing Google project).

Firebase automatically provisions resources for your Firebase project. When the process completes, you'll be taken to the overview page for your Firebase project in the Firebase console.



Enter your app's [application ID](#) in the **Android package name** field.

- An *application ID* is sometimes referred to as a *package name*.
 - Find this application ID in your module (app-level) Gradle file, usually `android/app/build.gradle` (example application ID: `com.yourcompany.yourproject`).
- Add the Firebase Android configuration file to your app:

Download the latest config file

[google-services.json](#)

This file contains configuration details such as keys and identifiers, for the services you just enabled.

App ID [?](#)

1:1008127664264:android:705eaadb6042e3da

App nickname

finalapp 

Package name

com.example.fire2

SHA certificate fingerprints [?](#)

Type [?](#)

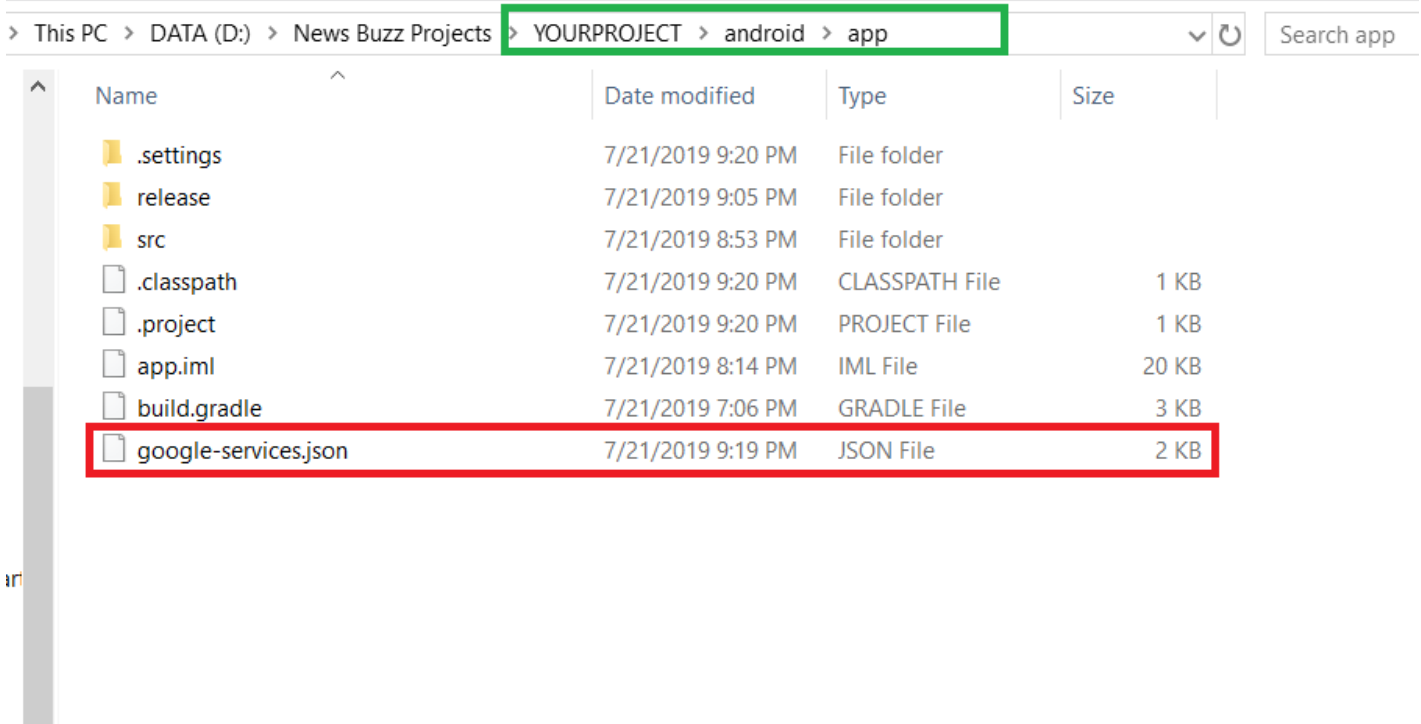
3:cd:4e:5e:fc:be:3f:2f:26:9c:57:e8:57:c7:14:53:ab:c1:d9:4c

SHA-1

[Add fingerprint](#)

[Remove this app](#)

- Click **Download google-services.json** to obtain your Firebase Android config file (`google-services.json`).
- You can download your [Firebase Android config file](#) again at any time.



To enable Firebase services in your Android app, add the [google-services plugin](#) to your Gradle files.

- a. In your root-level (project-level) Gradle file (android/build.gradle), add rules to include the Google Services plugin. Check that you have Google's Maven repository, as well.

```
b. buildscript {
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
    }

    // ...

    dependencies {
        // ...

        // Add the following line:
        classpath 'com.google.gms:google-services:3.2.1' // Google
        Services plugin
    }
}
```

```
}  
allprojects {  
    // ...  
  
    repositories {  
        // Check that you have following line (if not, add it):  
        google() // Google's Maven repository  
        // ...  
    }  
}
```

- b. In your module (app-level) Gradle file
(usually `android/app/build.gradle`), add the following line to the *bottom*
of the file

```
c. dependencies {  
    // ...  
}  
  
// ...  
  
// Add the following line to the bottom of the file:  
apply plugin: 'com.google.gms.google-services' // Google Play  
services Gradle plugin
```

Flutter Packages Used

Cupertino Icons

This is an asset repo containing the default set of icon assets used by Flutter's [Cupertino widgets](#).

```
cupertino_icons: ^0.1.2
```

firebase_auth plugin

A Flutter plugin to use the [Firebase Authentication API](#).

The Google Sign-in plugin is required to use the firebase_auth plugin for Google authentication.

```
firebase_auth: ^0.11.1+11
```

```
import 'package:firebase_auth/firebase_auth.dart';
```

http

This package contains a set of high-level functions and classes that make it easy to consume HTTP resources. It's platform-independent, and can be used on both the command-line and the browser.

```
http: ^0.12.0+1
```

```
import 'package:http/http.dart';
```

timeago

A library useful for creating fuzzy timestamps. (e.g. "5 minutes ago")

```
timeago: ^2.0.18
```

```
import 'package:timeago/timeago.dart';
```

google_sign_in

Flutter plugin for Google Sign-In, a secure authentication system for signing in with a Google account on Android and iOS.

```
google_sign_in: ^4.0.4
```

```
import 'package:google_sign_in/google_sign_in.dart';
```

Firebase Cloud Storage for Flutter

Flutter plugin for Firebase Cloud Storage, a powerful, simple, and cost-effective object storage service for Android and iOS.

```
firebase_storage: ^3.0.4
```

```
import 'package:firebase_storage/firebase_storage.dart';
```

Firebase Realtime Database for Flutter

Flutter plugin for Firebase Database, a cloud-hosted NoSQL database with Realtime data syncing across Android and iOS clients, and offline access.

```
firebase_database: ^3.0.5
```

```
import 'package:firebase_database/firebase_database.dart';
```

Fetching News from NewsAPI

Go to <https://newsapi.org/> and create a free account to get your unique API Key

Register for API key


First name


Email address

Choose a password



You are...

☒ I am an individual 

☐ I am a business, or am working on behalf of a business 



I'm not a robot



reCAPTCHA
[Privacy](#) - [Terms](#)

☐ I agree to the [terms](#).

☐ I promise to add an attribution link on my website or app to NewsAPI.org.

Submit

Create the following Asynchronous function in your dart file

```
Future getData() async{

    var response = await http.get(
        Uri.encodeFull(
            'https://newsapi.org/v2/top-
headlines?sources=' + newsSelection),
        headers: {
            "Accept": "application/json",
            "X-API-Key": "Your API KEY"
        });

    if(mounted) {
        this.setState(() {
            data = json.decode(response.body);
        });
    }
}
```