Name : Chapla Krishna Arpit

Roll No: 24

ID: 21ceuod006

Tutorial 10.1

Life cycle of the stateful widget:

It consist of the 3 state:

init State (): The init State gets triggered implicitly as

soon as the State initially get initialized. It is used when we want something to happen the moment our stateful widget is created.

build (): The build method gets triggered when the widgets are constructed and appear on the screen. It is used when we want something to happen every single time when our stateful widget gets rebuild.

deactivate(): Deactivate method gets called when the stateful widget gets destroyed ( just like destructor). It is used when we want something to happen just before our stateful widget gets destroyed.

Future: A Future is an object that represents the result of an asynchronous operation and can have two states:

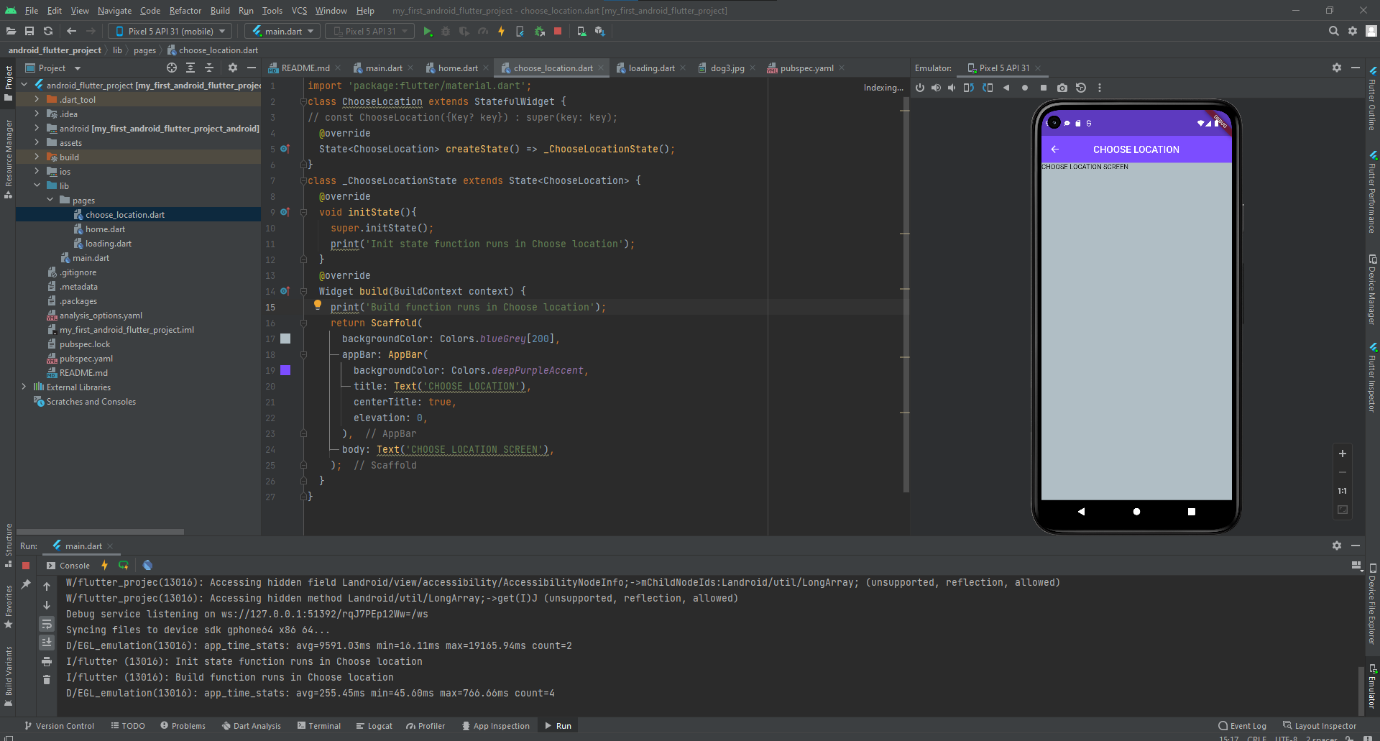
uncompleted or completed

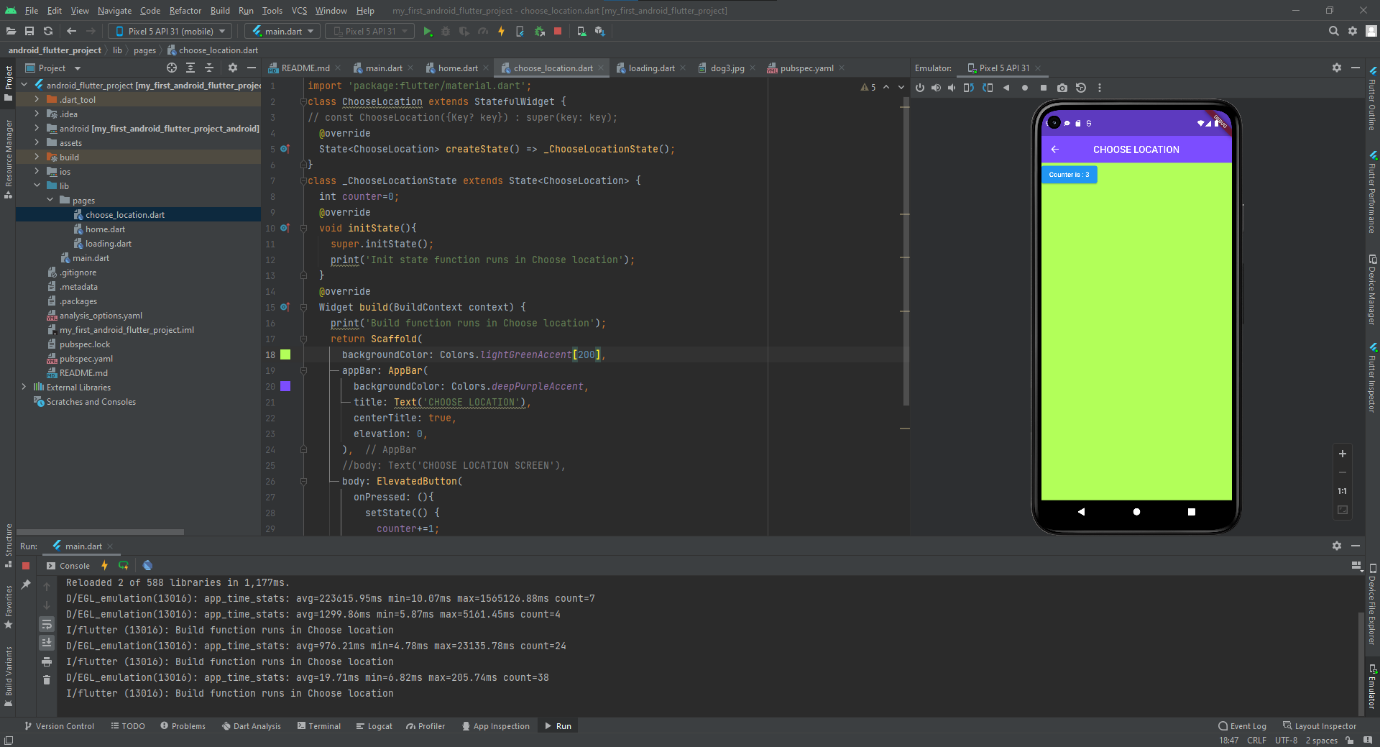
Async & Await: async and await are keywords that provide a way to make asynchronous operations appear synchronous.

Delay: Creates a future that runs its computation after a delay.The computation will be executed after the given duration has passed, and the future is completed with the result of the computation.

Duration: It represents a difference from one point in time to another.

1. Code test 1: Test for initState() and build(), setState().



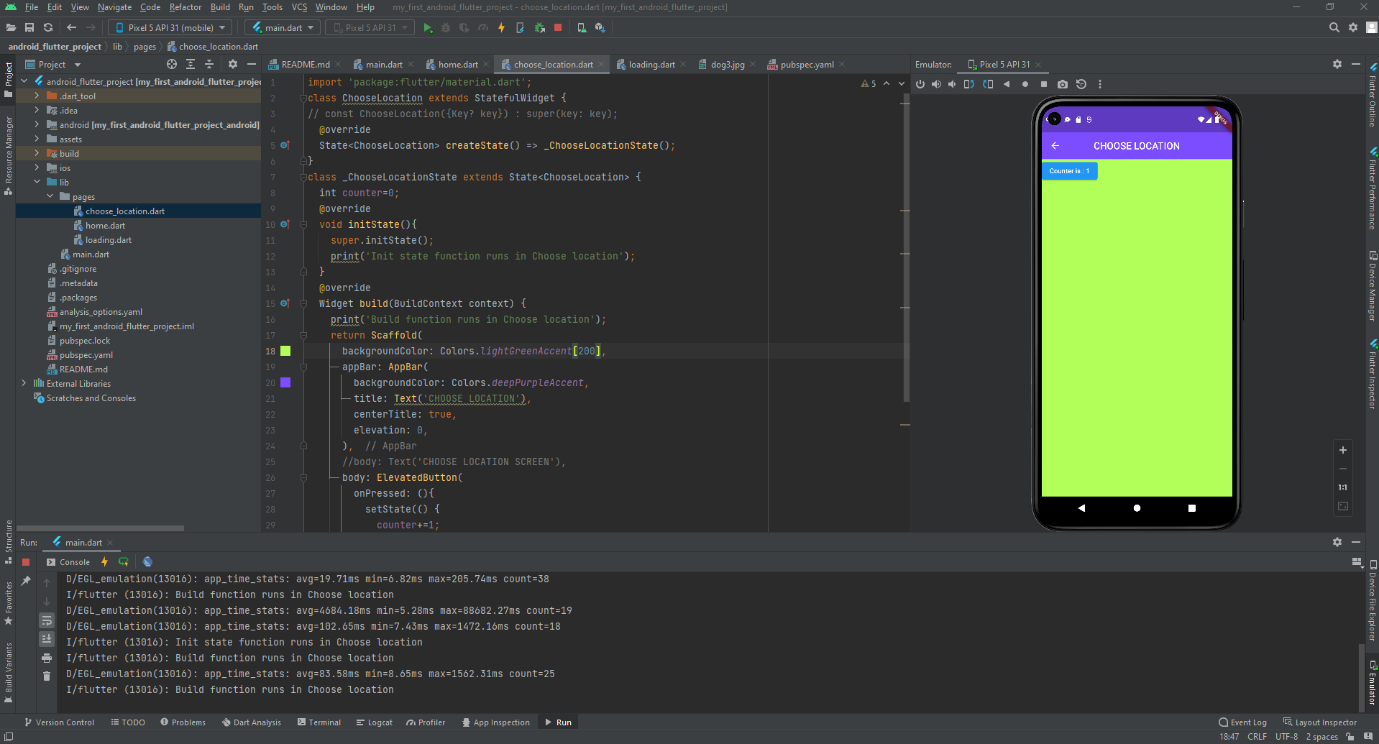


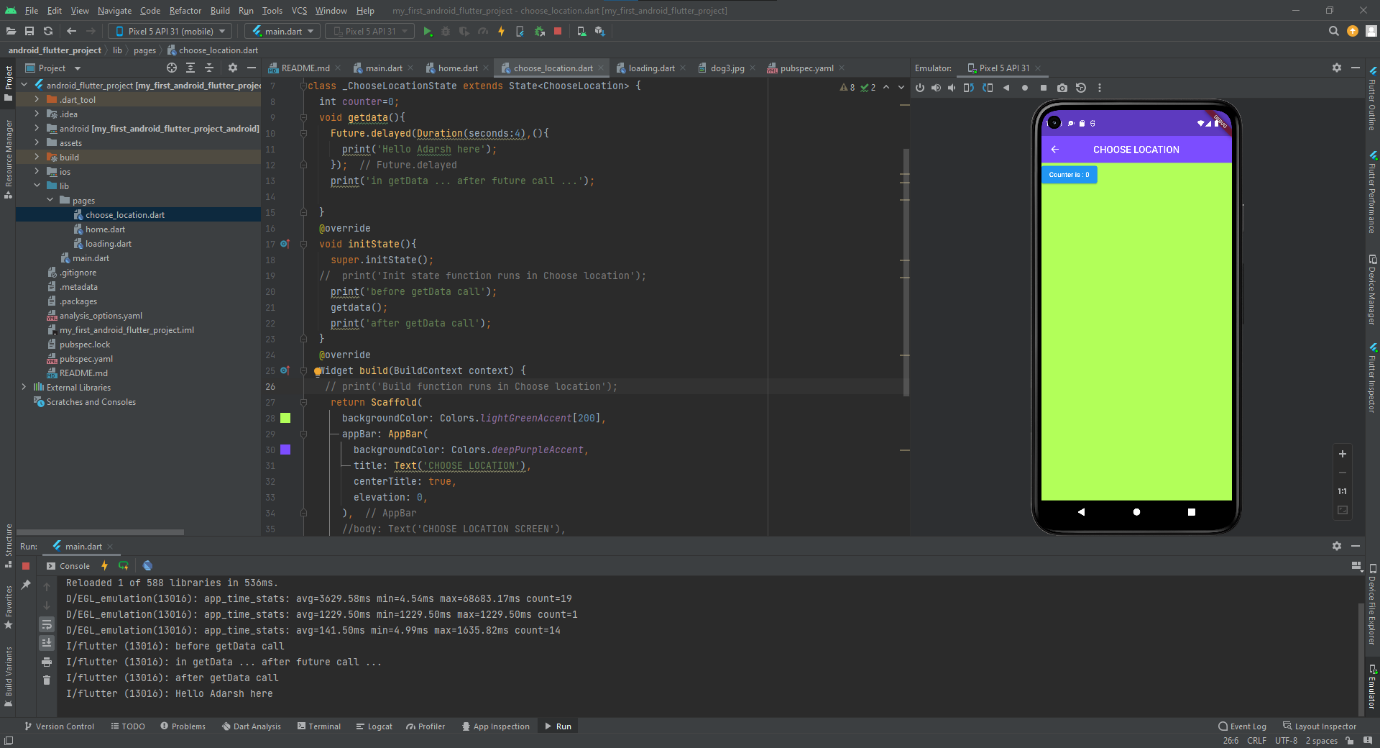
• When the first time we move to the “choose location” page ‘initState() runs.

• When we press the button, due to the ‘onPressed’ property every time setState() runs and it triggers the build() every time.

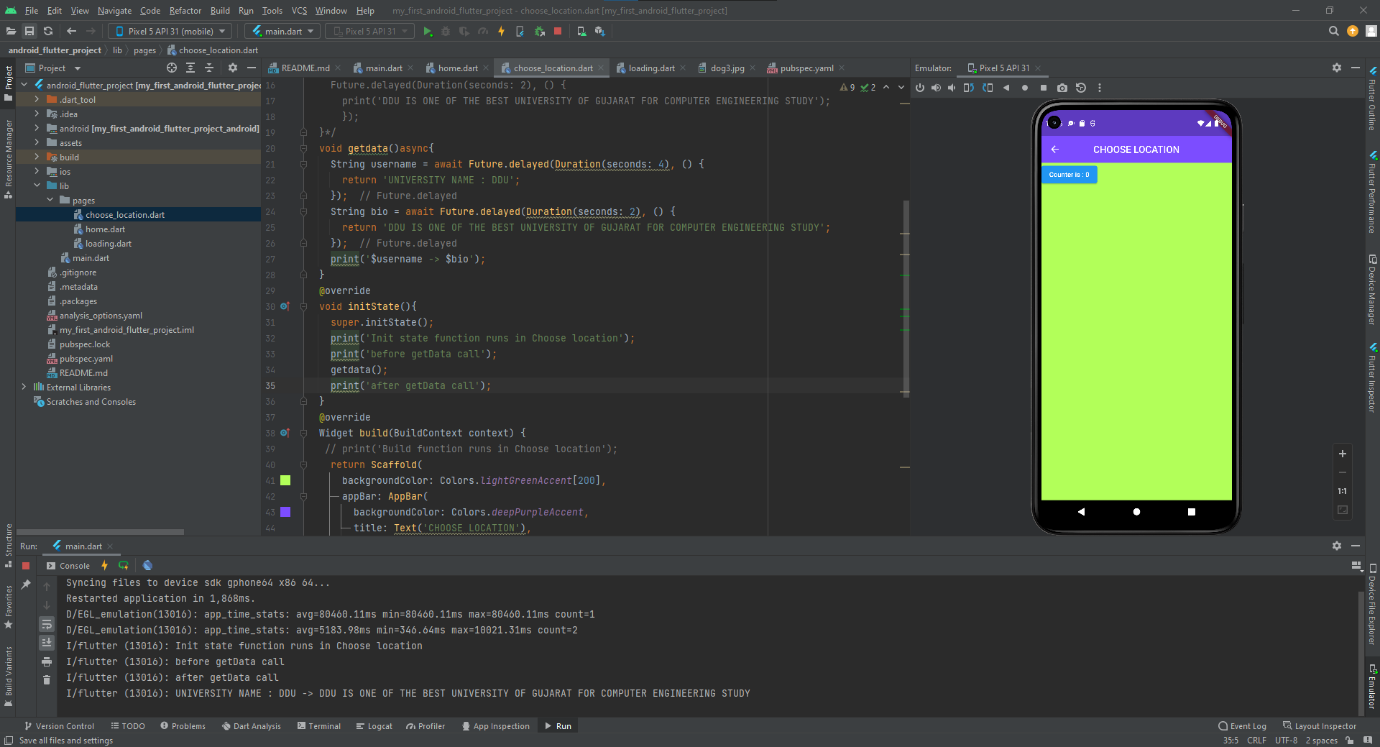
1. Code test 2 – Async in flutter

* Async:- Starts now and finishes in some time in future.
* Its non blocking code part
* If we request any api to execute/update some data…so we start the request but it  doesn’t finish at same time because it might required some time to complete request. In  the meantime, our code should not stop until the request is complete..
* Once the request is made, the rest of the code from file could carry on..





* Future: A future is an instance of the [Future](https://api.dart.dev/stable/dart-async/Future-class.html) class. A future represents the result of an asynchronous operation and can have two states: uncompleted or completed.
* Future.delayed: Creates a future that runs its computation after a delay. The computation will be executed after the given duration has passed, and the future is completed with the result of the computation.
* Duration: The [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) represents a single number of microseconds, which is the sum of all the individual arguments to the constructor.
* Sometimes new request depends on the data of the first request. So in such a situation, the second request must have to wait until the first one is not completed. The solution is to use async, await.



async: You can use the async keyword before a function’s body to mark it as asynchronous.

async function: An async function is a function labeled with the async keyword.

await: You can use the await keyword to get the completed result of an asynchronous expression. The await keyword only works within an async function. It will wait for the desired result.

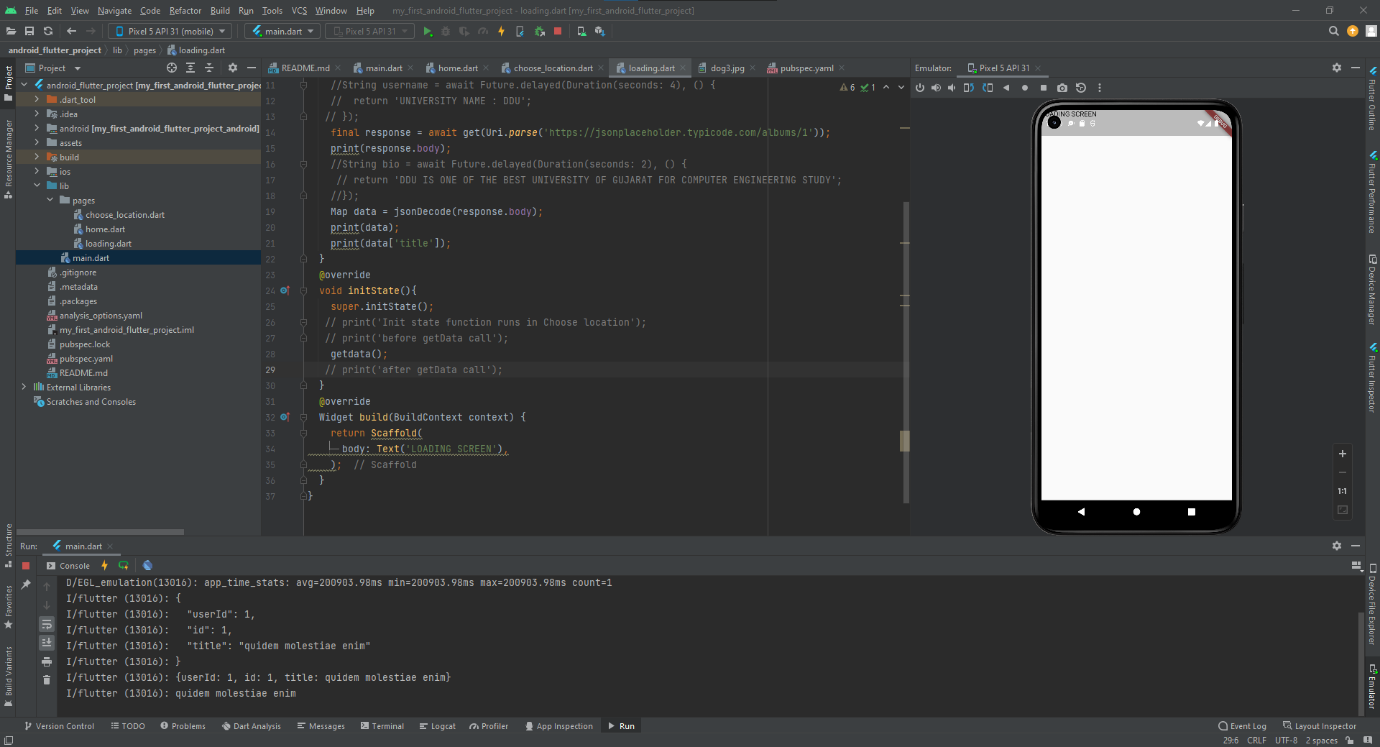
3)Code test 3: packages and API calling basic

A package is a namespace that contains a group of similar types of classes, interfaces,  and sub-packages. We can think of packages as similar to different folders on our  computers where we might keep movies in one folder, images in another folder,  software in another folder, etc. In Flutter, Dart organizes and shares a set of functionality  through a package. Flutter always supports shared packages, which is contributed by  other developers to the Flutter and Dart ecosystem. The packages allow us to build the  app without having to develop everything from scratch.

pubspec.yaml: It is the project's configuration file that will use a lot during working  with the Flutter project. This file contains:

Types of Packages:

1)Dart Package: It is a general package, which is written in the dart language, such as a  path package. This package can be used in both the environment, either it is a web or  mobile platform. It also contains some Flutter specific functionality and thus has a  dependency on the Flutter framework, such as fluro package.

2)Plugin Package: It is a specialized Dart package that includes an API written in Dart  code and depends on the Flutter framework. It can be combined with a platform specific implementation for an underlying platform such as Android (using Java or Kotlin), and iOS (using Objective C or Swift). The example of this package is the battery  and image picker plugin package.

Main.dart intiates the loading.dart .For testing you can use fake online Json Rest Api. Just copy the dependencies from [**https://pub.dev/packages?q=http**](https://pub.dev/packages?q=http)and paste them into pubspec.yaml.

4)Final code for 10.1

Main.dart

import 'package:flutter/material.dart';

import 'package:my\_first\_android\_flutter\_project/pages/choose\_location.dart';

import 'package:my\_first\_android\_flutter\_project/pages/loading.dart';

import 'pages/home.dart';

void main() => runApp(MaterialApp(

 // initialRoute: '/home',

  initialRoute: '/',

  routes: {

    '/':(context) => Loading(),

    '/home': (context) => Home(),

    '/location':(context)=>ChooseLocation(),

  },

));

Pages/loading.dart

import 'package:flutter/material.dart';

import 'package:http/http.dart';

import 'dart:convert';

class Loading extends StatefulWidget {

// const Loading({Key? key}) : super(key: key);

  @override

  State<Loading> createState() => \_LoadingState();

}

class \_LoadingState extends State<Loading> {

  void getdata()async{

    //String username = await Future.delayed(Duration(seconds: 4), () {

    //  return 'UNIVERSITY NAME : DDU';

   // });

    final response = await get(Uri.parse('https://jsonplaceholder.typicode.com/albums/1'));

    print(response.body);

    //String bio = await Future.delayed(Duration(seconds: 2), () {

     // return 'DDU IS ONE OF THE BEST UNIVERSITY OF GUJARAT FOR COMPUTER ENGINEERING STUDY';

    //});

    Map data = jsonDecode(response.body);

    print(data);

    print(data['title']);

  }

  @override

  void initState(){

    super.initState();

   // print('Init state function runs in Choose location');

   // print('before getData call');

    getdata();

   // print('after getData call');

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Text('LOADING SCREEN'),

    );

  }

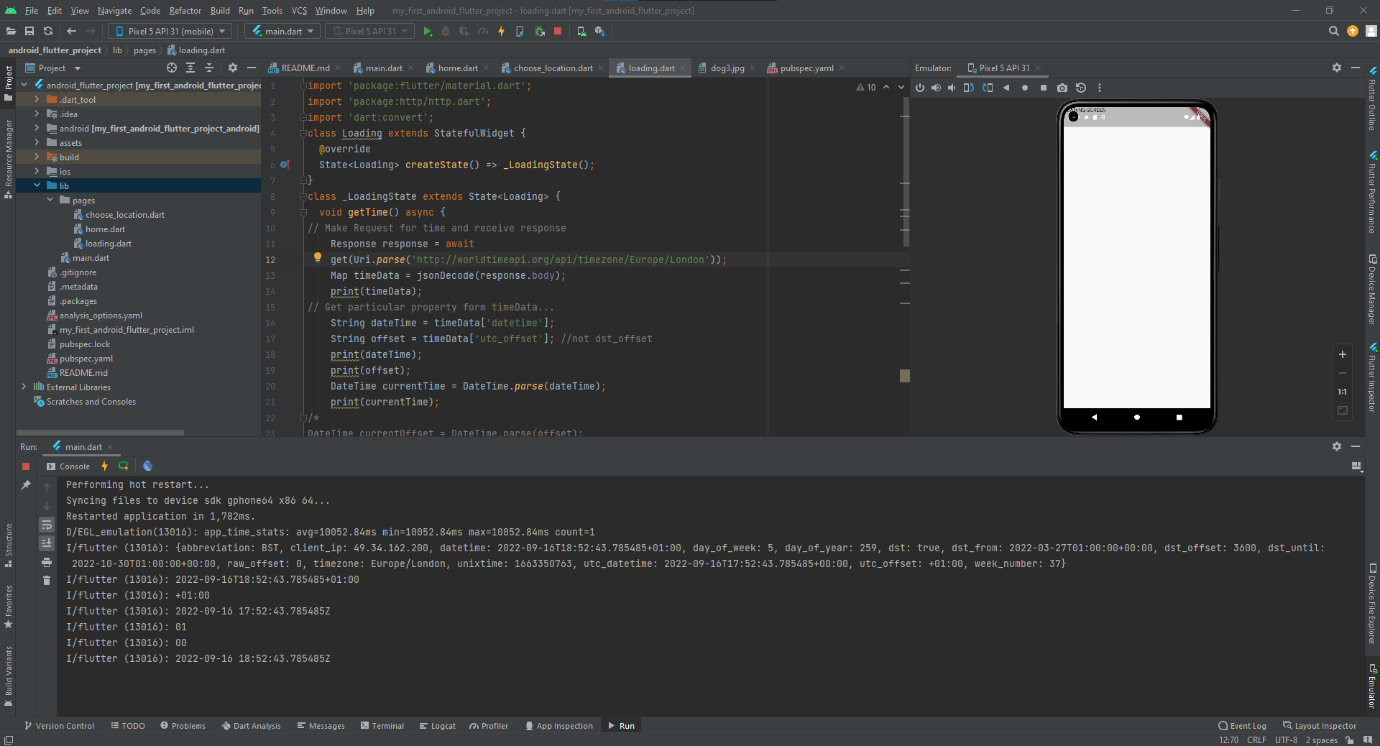
}

Tutorial 10.2

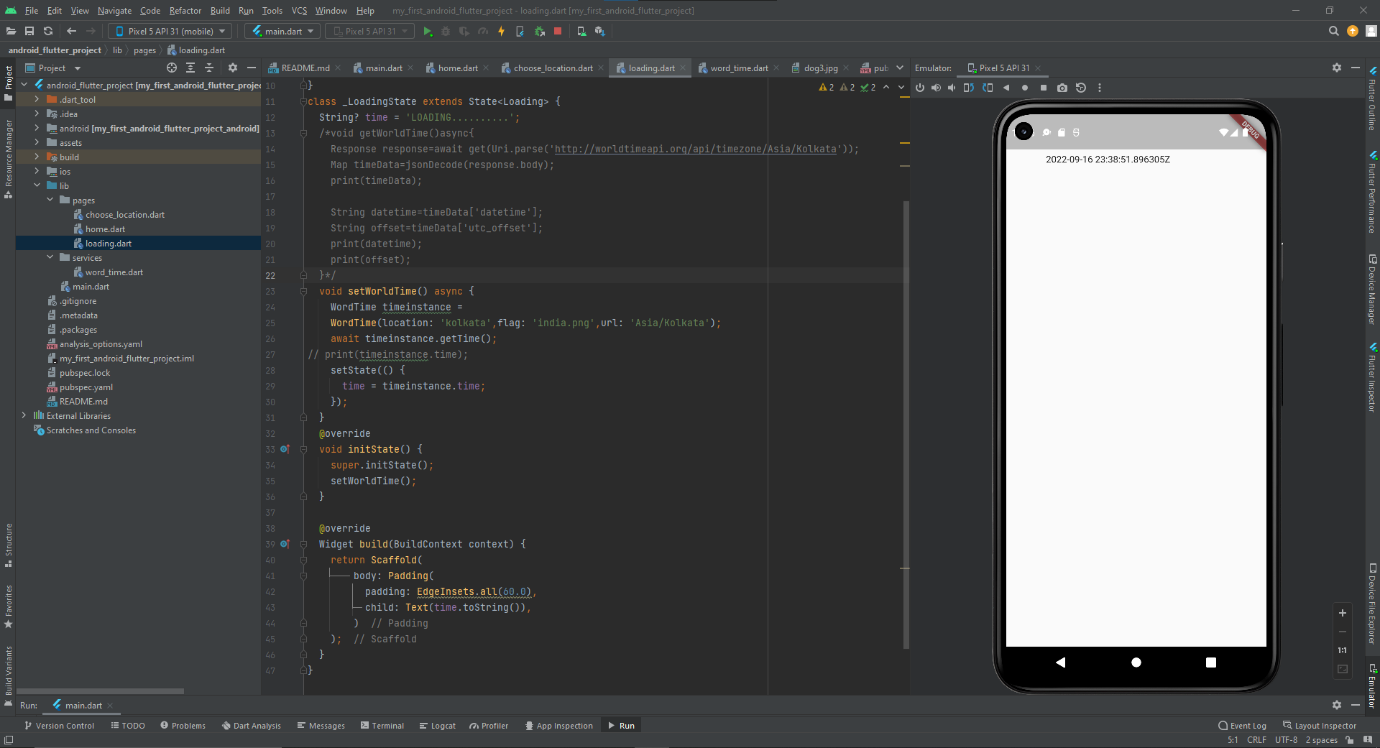
1) Code test 1: World time API and Date & Time

Plugin: Flutter plugins are thin Dart wrappers on top of native (Java, Kotlin, ObjC, Swift) mobile APIs and services. For instance, if you wanted to access a sensor on the phone, the only way is to write a plugin (or use one that’s already there).

Dart Package: This is exactly what it sounds like. You write a package entirely in pure Dart. Plugins are also (special) Dart packages. They get published to Pub and you interact with them via their Dart interface. The main difference between the two is that with a pure Dart package you don’t need to write any native code and testing is a breeze.



2) Code test 2:World time class widget service...



3)Final code for 10.2

loading.dart

import 'dart:convert';

import 'package:flutter/material.dart';

import '../services/word\_time.dart';

import 'package:http/http.dart';

class Loading extends StatefulWidget {

  @override

  State<Loading> createState() => \_LoadingState();

}

class \_LoadingState extends State<Loading> {

  String? time = 'LOADING..........';

  /\*void getWorldTime()async{

    Response response=await get(Uri.parse('http://worldtimeapi.org/api/timezone/Asia/Kolkata'));

    Map timeData=jsonDecode(response.body);

    print(timeData);

    String datetime=timeData['datetime'];

    String offset=timeData['utc\_offset'];

    print(datetime);

    print(offset);

  }\*/

  void setWorldTime() async {

    WordTime timeinstance =

    WordTime(location: 'kolkata',flag: 'india.png',url: 'Asia/Kolkata');

    await timeinstance.getTime();

// print(timeinstance.time);

    setState(() {

      time = timeinstance.time;

    });

  }

  @override

  void initState() {

    super.initState();

    setWorldTime();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        body: Padding(

          padding: EdgeInsets.all(60.0),

          child: Text(time.toString()),

        )

    );

  }

}

World\_time.dart

import 'package:http/http.dart';

import 'dart:convert';

class WordTime {

  String? location; // REAL LOCATION NAME FOR UI

  String? time; // the time in that location..

  String? flag; // flag images related to location country...do it yourself

  String? url; // end point of static url...which will change every timewhen location will change

  WordTime({ this.location,this.flag,this.url });

  Future<void> getTime() async {

// Make Request for time and receive response

    Response response = await

    get(Uri.parse('http://worldtimeapi.org/api/timezone/$url')); // Asia/Kolkata

    Map timeData = jsonDecode(response.body);

// Get particular property form timeData...

    String dateTime = timeData['datetime'];

    String offset = timeData['utc\_offset']; //not dst\_offset

    String offsetHours = offset.substring(1,3);

    String offsetMinutes = offset.substring(4,6);

// create DateTime object

    DateTime currenttime = DateTime.parse(dateTime);

    currenttime = currenttime.add(

        Duration(minutes:

        int.parse(offsetMinutes),hours:int.parse(offsetHours)));

//set the time property of class...

    time = currenttime.toString();

  }

}