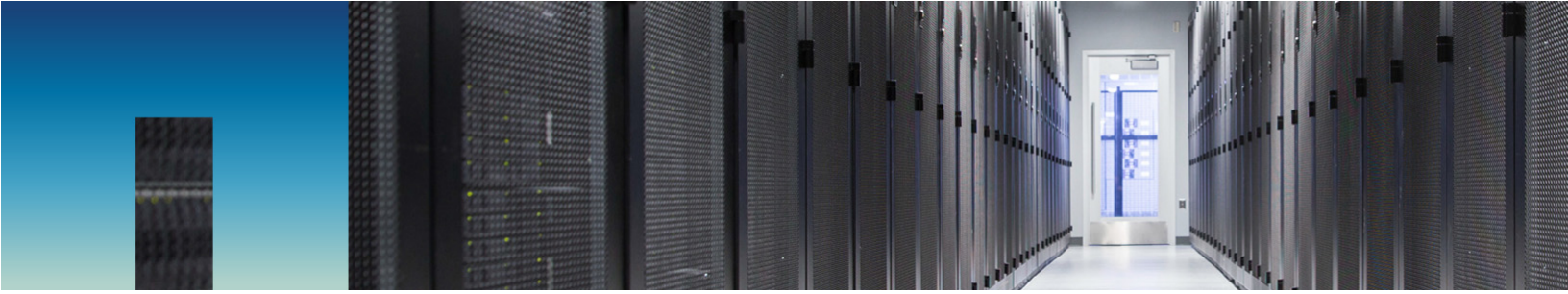


Application Development Procedure

Emily Schleier, Rajesh Sunkara, Sree Lakshmi Lanka, NetApp
December 2020

Abstract

This document provides a step-by-step tutorial for designing, developing and implementing a mobile application on Android Mobile devices. It is intended to educate the students and teachers of the Gashora Girls Academy to create a e-learning platform and share educational resources with the students using the mobile application.

NetApp®

TABLE OF CONTENTS

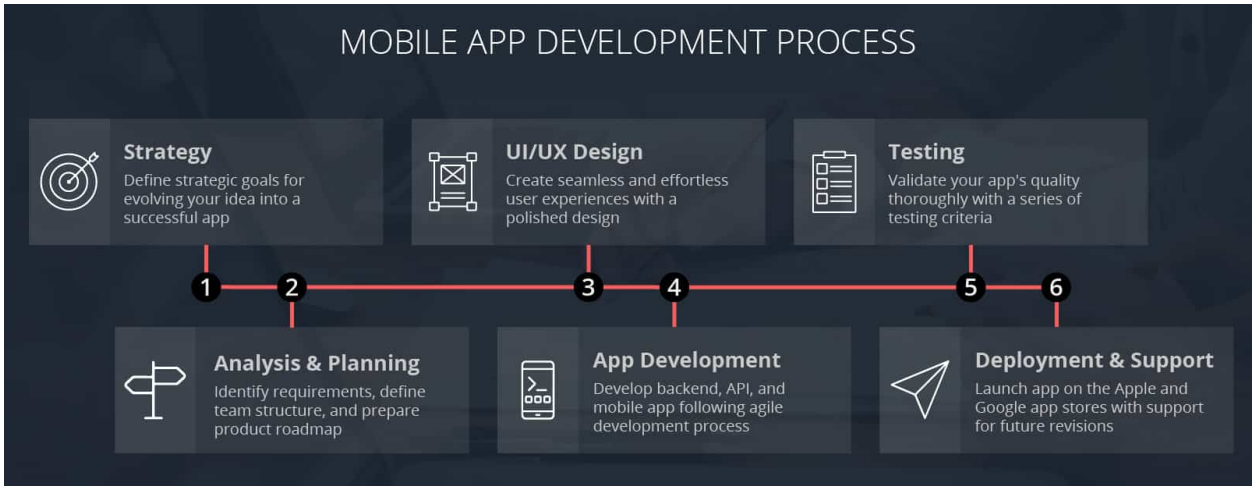
1 Introduction

The COVID-19 has resulted in schools shut all across the world. Globally, over 1.2 billion children are out of the classroom. As a result, education has changed dramatically, with the distinctive rise of e-learning, whereby teaching is undertaken remotely and on digital platforms. Gashora Girls Academy management sees this as an opportunity to update their infrastructure to serve On-line learning to each student at the remotest locations with minimal expense.

This document provides a step-by-step tutorial for designing, developing and implementing a mobile application on Android Mobile devices. It is intended to educate the students and teachers of the Gashora Girls Academy to create a e-learning platform and share educational resources with the students using the mobile application.

1.1 Development Process

An effective app development process flow spans over six key phases. Regardless of the size and scope of your project, following this development process will make your mobile app development initiative a success.



1.1.1 Strategy

The first phase of the mobile app development process is defining the strategy for evolving your idea into a successful app. You may include a more significant part of this in your overall enterprise mobility strategy. As one app's objectives may differ from another, there is still an app-specific impact to the mobility strategy to address during the development process.

In this phase, you will:

- Identify the app users
- Research the competition
- Establish the app's goals and objectives
- Select a mobile platform for your app

Your strategy helps focus your vision on a clear picture of your app idea. With this in mind, you can go deeper into the next phase of the mobile application development process.

1.1.2 Analysis and Planning

Analysis and planning begin with defining use cases and capturing detailed functional requirements.

After you have identified the requirements for your app, prepare a product roadmap. This includes prioritizing the mobile app requirements and grouping them into delivery milestones. If time, resources or costs are a concern, then define your minimum-viable-product (MVP) and prioritize this for the initial launch. Have you selected the name of your app yet? Mobile app names are like domain names and have to be unique within each app store. Research each app store ensuring your app's name isn't already in use!

1.1.3 UI / UX Design

The purpose of an app's design is to deliver seamless and effortless user experiences with a polished look. The success of a mobile app is determined based on how well users are adopting and benefiting from all its features. The goal for mobile app UI / UX design is creating excellent user experiences making your app interactive, intuitive, and user-friendly. While polished UI designs will help with early adoption, your app must have intuitive user experiences to keep app users engaged. We will discuss this in detail in the Design section.

1.1.4 Application Development

Planning remains an integral part of this phase in the mobile app development process. Before actual development/programming efforts start, you will have to:

- define the technical architecture,
- pick a technology stack, and
- define the development milestones.

A typical mobile app project is made up of three integral parts: back-end/server technology, API(s) and the mobile app front-end.

1.1.5 Testing

Performing thorough quality assurance (QA) testing during the mobile app development process makes applications stable, usable, and secure. To ensure comprehensive QA testing of your app, you first need to prepare test cases that address all aspects of app testing.

Similar to how use cases drive the process of mobile app development, test cases drive mobile app testing. Test cases are for performing test steps, recording testing results for software quality evaluation, and tracking fixes for retesting. A best practice approach is involving your QA team in the Analysis and Design stages. The familiarity with your app's functional requirements and objectives will help produce accurate test cases.

1.1.6 Deployment & Support

Releasing a native mobile app requires submitting your app to the app stores, Apple App Store for iOS apps and Google Play for Android apps. However, you will need a developer account with Apple App Store and Google Play Store before launching your mobile app.

An app's release in the app store requires preparing metadata including:

- Your app's title
- Description
- Category
- Keywords

- Launch icon
- App store screenshots

There isn't any review process with Android apps, and they become available in the app store within a few hours of submission.

After your app becomes available in the app stores, monitor its usage through mobile analytics platforms and track Key Performance Indicators (KPIs) for measuring your app's success. Frequently check crash reports, or other user reported issues.

2 Design

1. Intro to User Experience Design
    - What is UX Design?
    - What is Human-Centered Design?
    - Design Thinking
2. User Experience Design (UX)
    - Research
        1. Interviews
        2. Research synthesis
    - Creating User Personas
3. Ideation + Prototyping
    - Wireframing
    - Prototyping
4. Visual Design
    - Branding

2.1 Introduction to UX

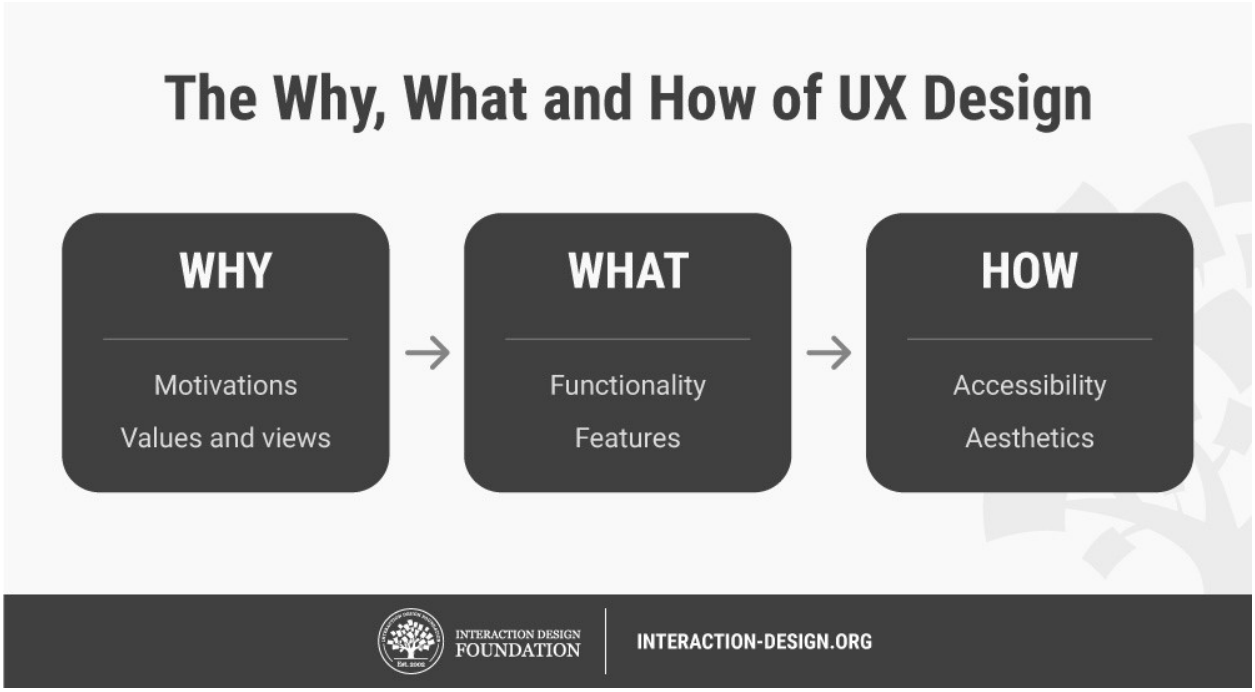
What is User Experience Design (UX)?

"A holistic, multidisciplinary approach to the design of user interfaces for digital products, defining their form, behavior, and content. User experience design integrates interaction design, industrial design, information architecture, information design, visual interface design, user assistance design, and user-centered design, ensuring coherence and consistency across all of these design dimensions."

— Pabini Gabriel-Petit

Assignment: Read "What is User Experience Design? Overview, Tools, and Resources" https://www.smashingmagazine.com/2010/10/what-is-user-experience-design-overview-tools-and-resources/

Extra credit: Watch the videos from the Interaction Design Foundation about UX https://www.interaction-design.org/literature/topics/ux-design



Let's look at Human-Centered Design (basically another name for UX Design)

Definition: Human-centered design sits at the intersection of empathy and creativity.

Assignment: Watch IDEO's videos about "What is human-centered design? Inspiration. Ideation. Implementation."

https://vimeo.com/106505300

More resources:

IDEO's Field Guide to Human Centered Design free PDF book download. Focuses on human-centered design for social innovation.

https://www.designkit.org/resources/1

***Maybe we can utilize this course Mr. Benjamin?* IDEO's Intro to Human Centered Design Course. Free. Scheduled: https://acumenacademy.org/course/design-kit-human-centered-design

2.2 Design Thinking Approach to App Design

Design Thinking is an iterative process in which we seek to understand the user, challenge assumptions, and redefine problems in an attempt to identify alternative

strategies and solutions that might not be instantly apparent with our initial level of understanding. At the same time, Design Thinking provides a solution-based approach to solving problems. It is a way of thinking and working as well as a collection of hands-on methods.

Source: *https://www.interaction-design.org/literature/article/what-is-design-thinking-and-why-is-it-so-popular*

Watch: IBM's explanation of their Design Thinking approach.

https://www.youtube.com/watch?v=pXtN4y3O35M

2.2.1 The Design Thinking Process

You can utilize design thinking to approach problem-solving for users in an interactive design.

The main stages of this process, while it varies from company to company, roughly breaks down as follows:

1. **Empathize/Research - Understanding your Users**
   Empathy is the centerpiece of a human-centered design process. The Empathize mode is the work you do to understand people, within the context of your design challenge. It is your effort to understand the way they do things and why, their physical and emotional needs, how they think about the world, and what is meaningful to them.
2. **Define - What problem are you hoping to solve?**
   The Define mode of the design process is all about bringing clarity and focus to the design space. It is your chance, and responsibility, as a design thinker to define the challenge you are taking on, based on what you have learned about your user and about the context. After becoming an instant-expert on the subject and gaining invaluable empathy for the person you are designing for, this stage is about making sense of the widespread information you have gathered.
3. **Ideate - How can you possibly solve this problem?**
   Ideate is the mode of the design process in which you concentrate on idea generation. Mentally it represents a process of "going wide" in terms of concepts and outcomes. Ideation provides both the fuel and also the source material for building prototypes and getting innovative solutions into the hands of your users.
4. **Prototype - Begin to conceptualize your ideas.**
   The Prototype mode is the iterative generation of artifacts intended to answer questions that get you closer to your final solution. A prototype can be anything that a user can interact with – be it a wall of post-it notes, a gadget you put together, a role-playing activity, or even a storyboard. Ideally you bias toward something a user can experience.
5. **Test - Time to test those prototypes!**
   The Test mode is when you solicit feedback, about the prototypes you have created, from your users and have another opportunity to gain empathy for the people you are designing for. Testing is another opportunity to understand your user, but unlike your initial empathy mode, you have now likely done more framing of the problem and created prototypes to test. Ideally you can test within a real context of the user's life.
6. **Implement/Iterate - Take what you have learned and update your designs.**
   Iteration is fundamental to good design. Iterate both by cycling through the steps 1-5 multiple times, and also by iterating within a step—for example by creating multiple prototypes or

trying variations of a brainstorming topics with multiple groups. Generally, as you take multiple cycles through the design process your scope narrows and you move from working on the broad concept to the nuanced details, but the process still supports this development.

User Experience Research (Empathize + Define)

What problem are you trying to solve for your users? The user-centered design process helps you design for their needs first.

Video: Empathy: https://www.designkit.org/mindsets/4 Stanford Design School - 1 minute

Activity and Video: Interviewing Your Potential Users - IDEO: https://www.designkit.org/methods/interview

Article: You are not like your users: https://uxmyths.com/post/715988395/myth-you-are-like-your-users

Video: Three different interview styles - Nielsen Norman Group - 3 minutes: https://www.nngroup.com/videos/3-types-user-interviews/?lm=interviewing-users&pt=article

Article: How to Do a User Interview: Nielsen Norman Group

https://www.nngroup.com/articles/user-interviews/

Extra credit for User Research:

- Card sorting activity: https://blog.optimalworkshop.com/why-card-sorting-loves-tree-testing/
- Surveys are great for quickly collecting large amounts of data about your users. https://www.invisionapp.com/inside-design/how-to-create-a-survey/

Problem definition and description.

After a round of research, you're ready to define your users and the problem you're hoping to solve.

Activity: Problem definition. How Might We. Every problem is an opportunity for design. By framing your challenge as a How Might We question, you'll set yourself up for an innovative solution.

https://www.designkit.org/methods/3

Activity: Build user personas from your research.

Video: How to create realistic User Personas - UX Mastery - 3 minutes

https://www.youtube.com/watch?v=B23iWg0koi8

Video: Creating Customer Personas - Circus Street - 2 minutes

https://www.youtube.com/watch?v=Av-1Htt7sOA

Tools: Photos for user personas:

https://www.flickr.com/photos/jasontravis/sets/72157603258446753/

2.2.2 Ideate + Prototyping + Testing

Wireframing and prototyping is an integral part of the design process. Here we are beginning to give form to our design solution.

Article: Everything you wanted to know about Prototyping:

https://www.smashingmagazine.com/2017/12/prototyping-driven-process/

Activity: Build a sitemap for your app:

Create your website map or application structure, add notes, specify page content, and use color schemes to improve your site map design or content planning. https://octopus.do

Article: User Research with Prototypes:

https://medium.com/eightshapes-llc/user-research-with-prototypes-asking-the-right-questions-d38bdaf074bc

Activity: Build a prototype of your app with Adobe XD and get feedback from users. Adobe XD is a design tool (there are many that you can choose from!) that will take some time to learn, but it will help you design your app and user experience before moving to the implementation phase.

Adobe XD tutorials: https://helpx.adobe.com/xd/tutorials.html

Extra credit: 10 Usability Heuristics for User Interface Design

https://www.nngroup.com/articles/ten-usability-heuristics/

Visual Design

Activity: Create a mood board for your apps branding. “A mood board is a collection of like-minded design examples, organized and presented to accomplish a task.” Mood boards are used for three main purposes: Definition, Inspiration, Direction. - In Vision: https://www.invisionapp.com/inside-design/mood-board-examples/

Explore: Google’s Material Design is a great place to learn about common interaction design patterns.

https://material.io

Explore: Need help with your color palette? Try this fun tool.

Colors: https://coolors.co

3 Implementation with Flutter

3.1 Flutter – Introduction

Flutter – a simple and high-performance framework based on Dart language, provides high performance by rendering the UI directly in the operating system's canvas rather than through native framework

Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environment and designing the application using widgets is as simple as designing HTML. Flutter widgets also supports animations and gestures.

Features of Flutter: Flutter framework offers the following features to developers:

- Modern and reactive framework.
- Uses Dart programming language and it is very easy to learn.
- Fast development.
- Beautiful and fluid user interfaces.
- Huge widget catalog.
- Runs same UI for multiple platforms.
- High performance application.

Advantages of Flutter

Flutter comes with beautiful and customizable widgets for high performance and outstanding mobile application. It fulfills all the custom needs and requirements. Besides these, Flutter offers many more advantages as mentioned below:

- Dart has a large repository of software packages which lets you to extend the capabilities of your application.
- Developers need to write just a single code base for both applications (both Android and iOS platforms). Flutter may to be extended to other platform as well in the future.
- Flutter needs lesser testing. Because of its single code base, it is sufficient if we write automated tests once for both the platforms.
- Flutter's simplicity makes it a good candidate for fast development. Its customization capability and extendibility make it even more powerful.
- With Flutter, developers have full control over the widgets and its layout.
- Flutter offers great developer tools, with amazing hot reload.

3.1.1 Flutter – Installation

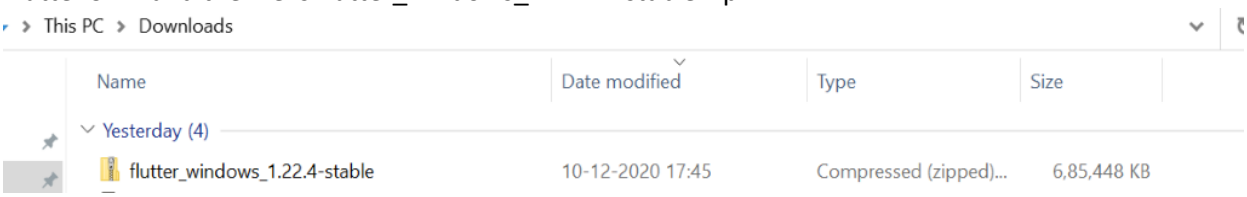
To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems**: Windows 7 SP1 or later (64-bit), x86-64 based

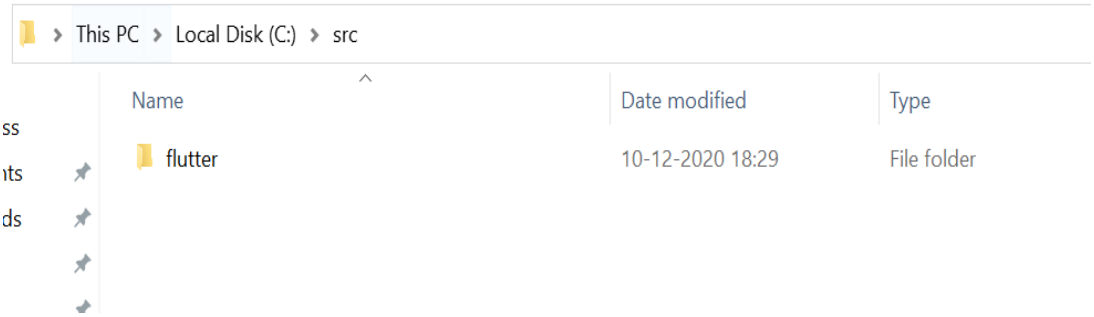
- **Disk Space**: 1.32 GB (does not include disk space for IDE/tools).

Installation in Windows In this section, let us see how to install Flutter SDK and its requirement in a windows system.

Step 1: Go to URL, https://flutter.dev/docs/get-started/install/windows and download the latest Flutter SDK. and the file is flutter_windows_v1.22.4-stable.zip.



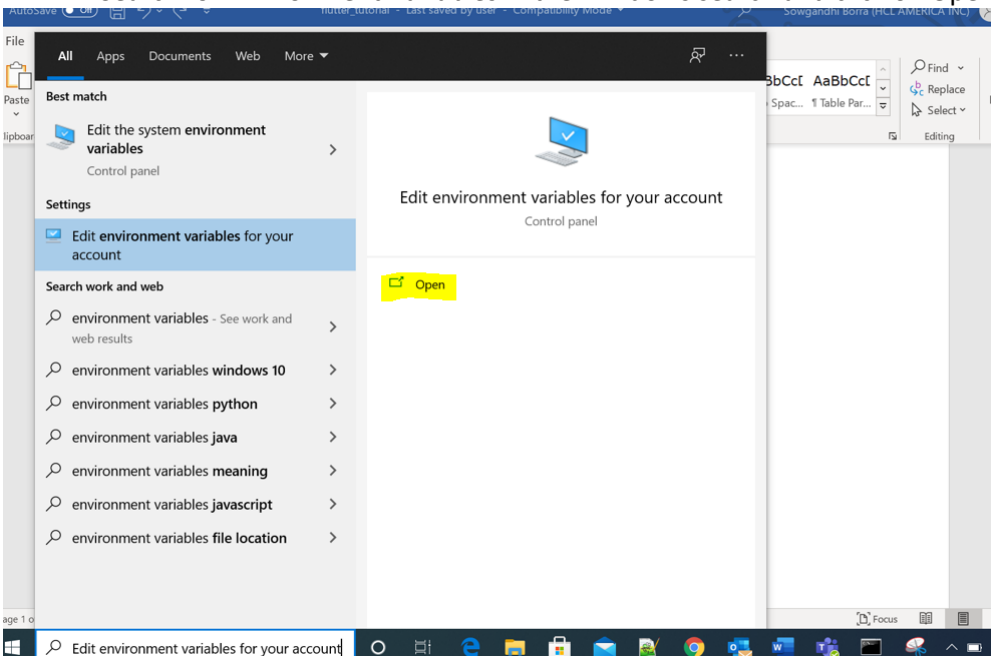
Step 2: Unzip the zip archive in a folder, say C:\src\flutter\



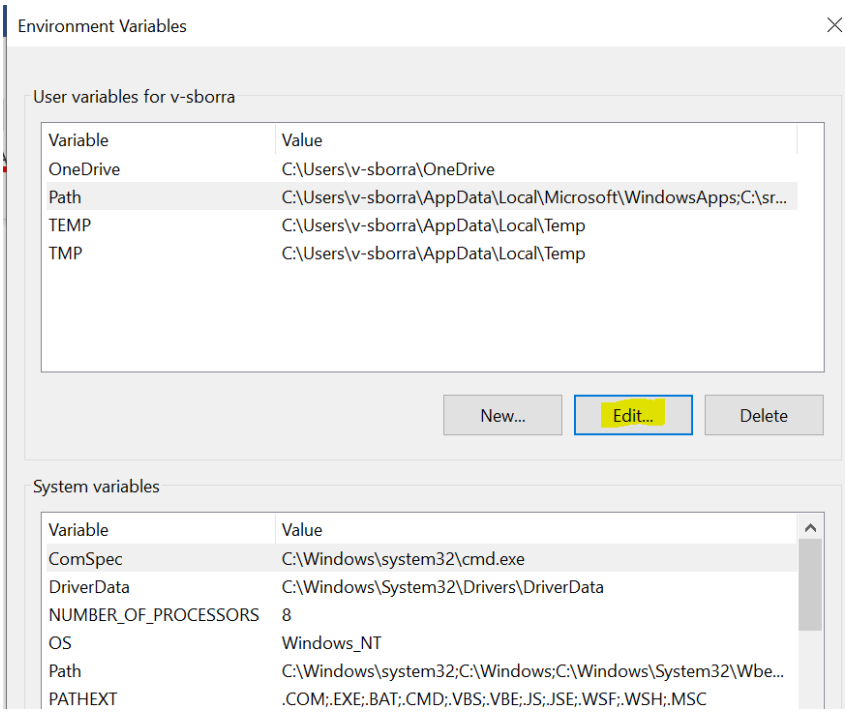
Step 3: Update the system path to include flutter bin directory.

> export PATH="$PATH:/path/to/flutter/bin" follow the steps below.

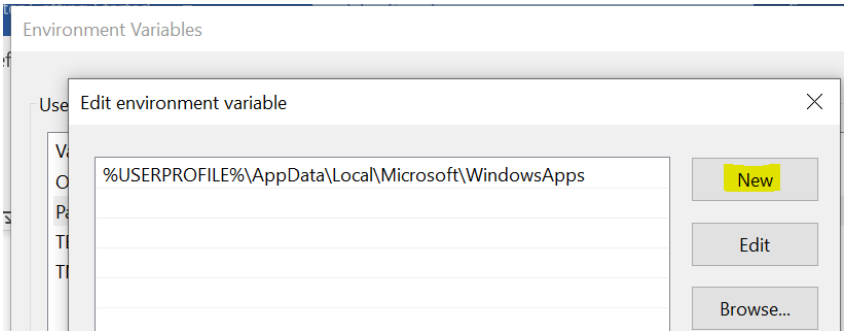
- Search for Environment Variables in the Windows search and click on Open



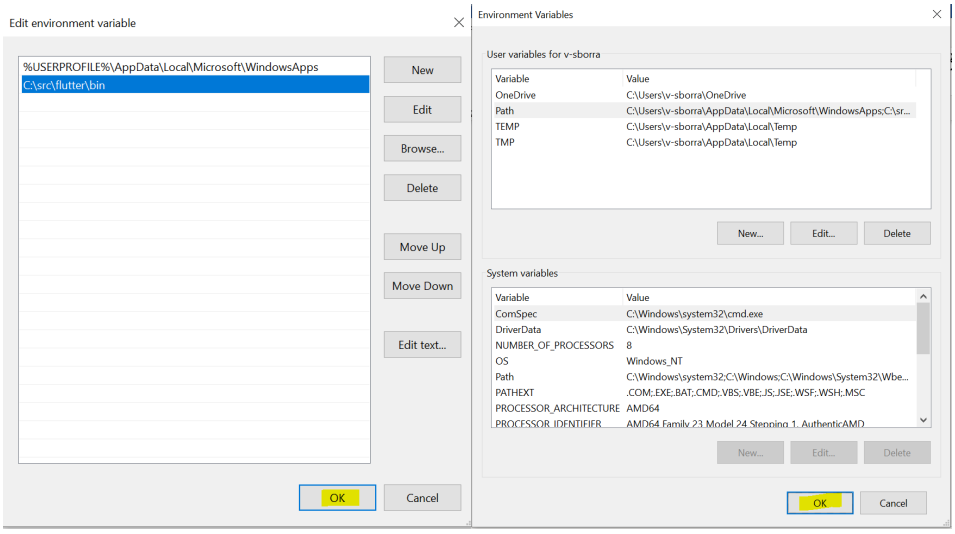
- Select Path and click on Edit button.



- Click on New Button

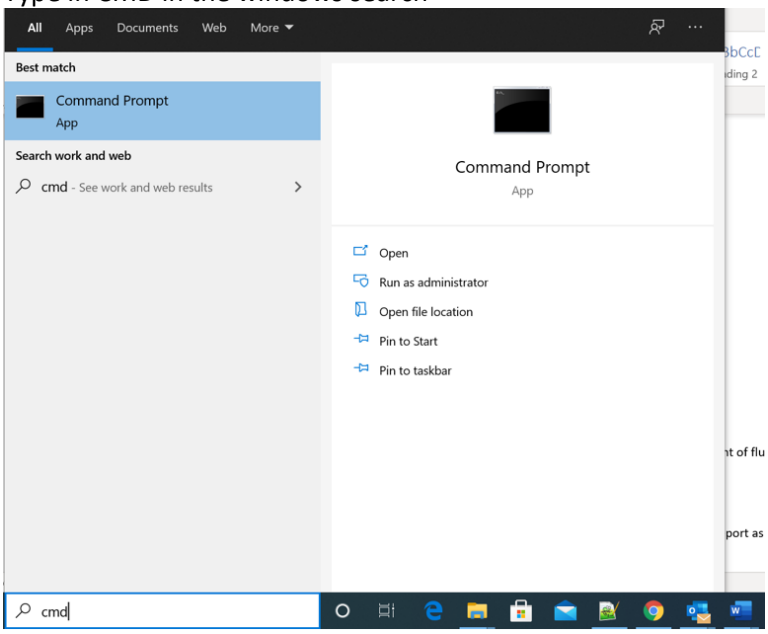


- Enter the “C:\src\flutter\bin” and click on OK button on Edit Environment Variables .
- Click on OK for the Environment Variables window also.

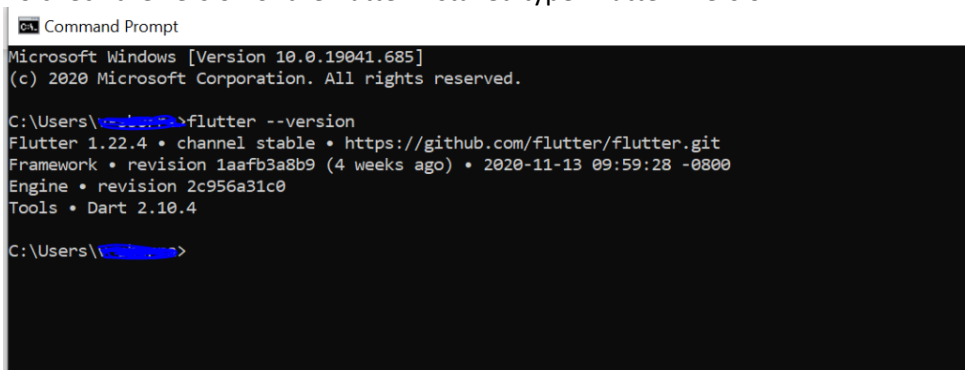


Step 4: Flutter provides a tool, flutter doctor to check that all the requirement of flutter development is met. Steps as below.

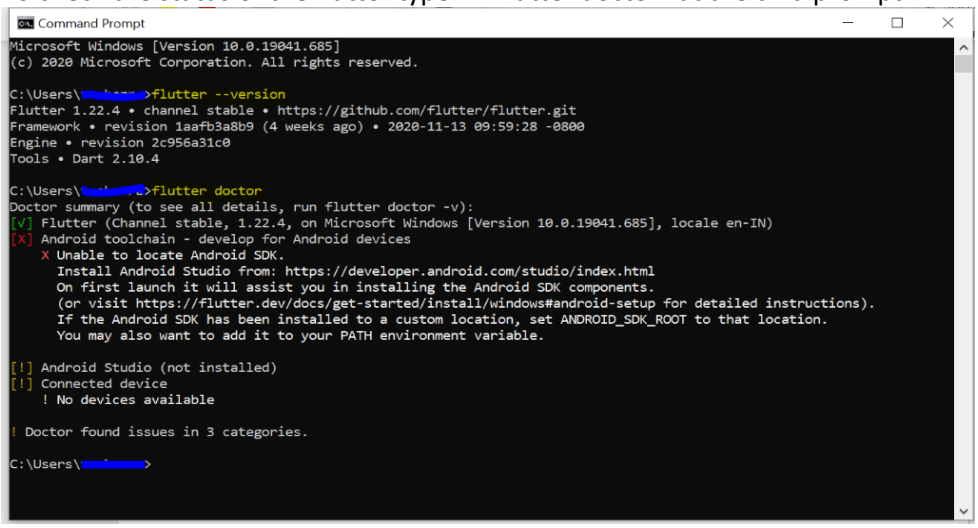
- Type in CMD in the windows search



- To check the version of the flutter installed type “flutter –version”



- To check the status of the flutter type in “flutter doctor” at the cmd prompt.



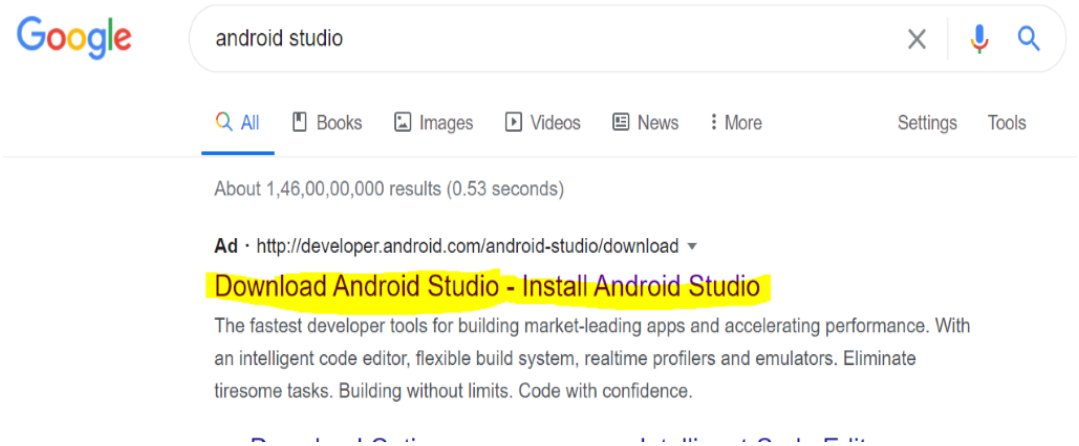
Step 5: Running the above command will analyze the system and show its report as shown below:

```
C:\Users\--------->flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 1.22.4, on Microsoft Windows [Version 10.0.19041.685], locale en-IN)
[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/windows#android-setup for detailed instructions).
      If the Android SDK has been installed to a custom location, set ANDROID_SDK_ROOT to that location.
      You may also want to add it to your PATH environment variable.

[!] Android Studio (not installed)
[!] Connected device
    ! No devices available

! Doctor found issues in 3 categories. The report says that all development tools are available but the device is not connected. We can fix this by connecting an android device through USB or starting an android emulator.
```

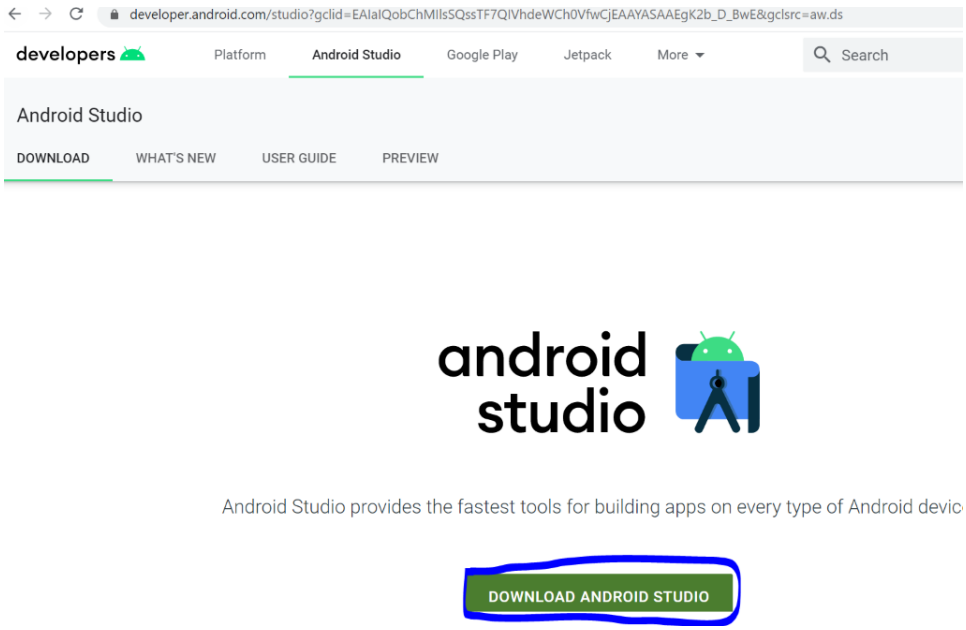
Step 6: Install the latest Android SDK, if reported by flutter doctor

Step 7: Install the latest Android Studio, if reported by flutter doctor

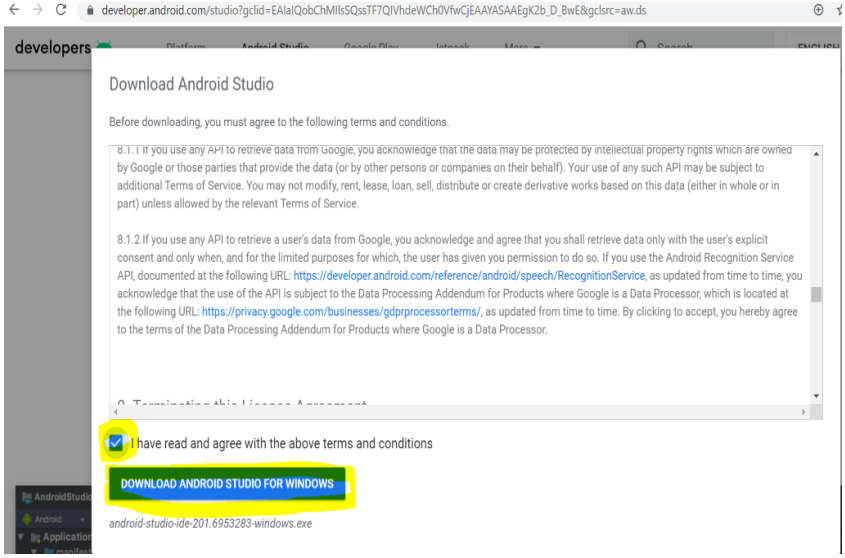
- Search for Android Studio on Google and click on the link mentioned below.
- https://developer.android.com/studio?gclid=EAIaIQobChMIlsSQssTF7QIVhdeWCh0VfwCjEAAYASAAEgK2b_D_BwE&gclsrc=aw.ds



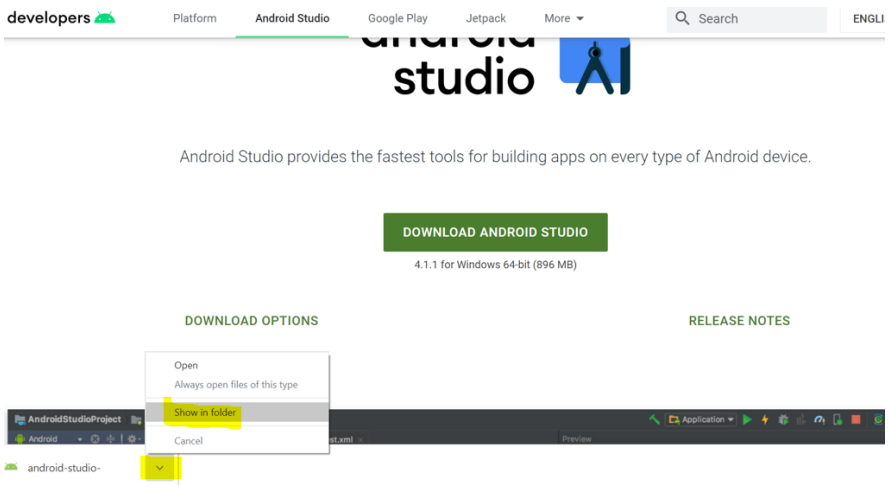
- Download Android Studio by clicking on the download button as shown below.



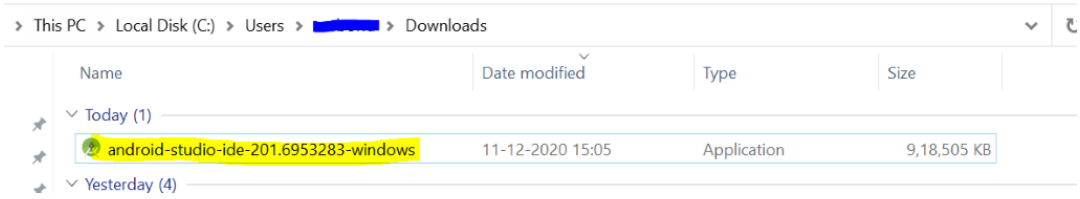
- Click the check box about the terms and conditions and click on Download Android Studio for Windows.



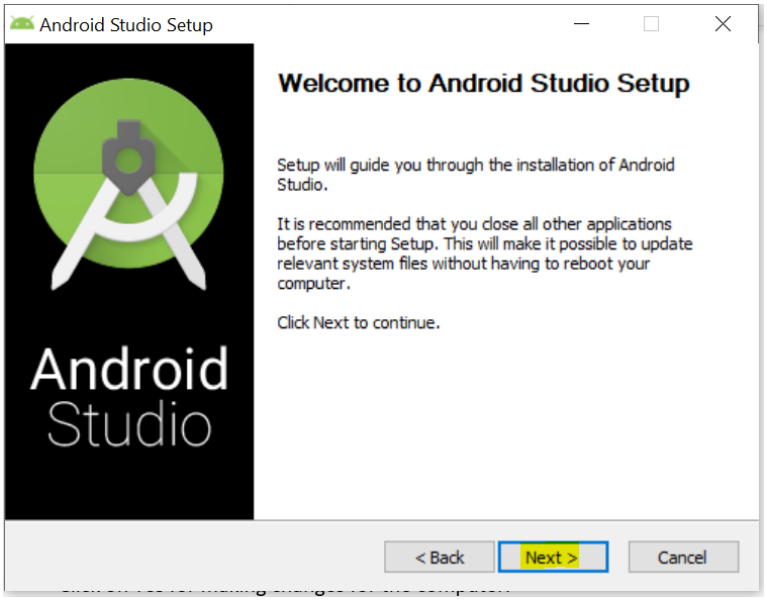
- Click on “Show in folder” as shown below

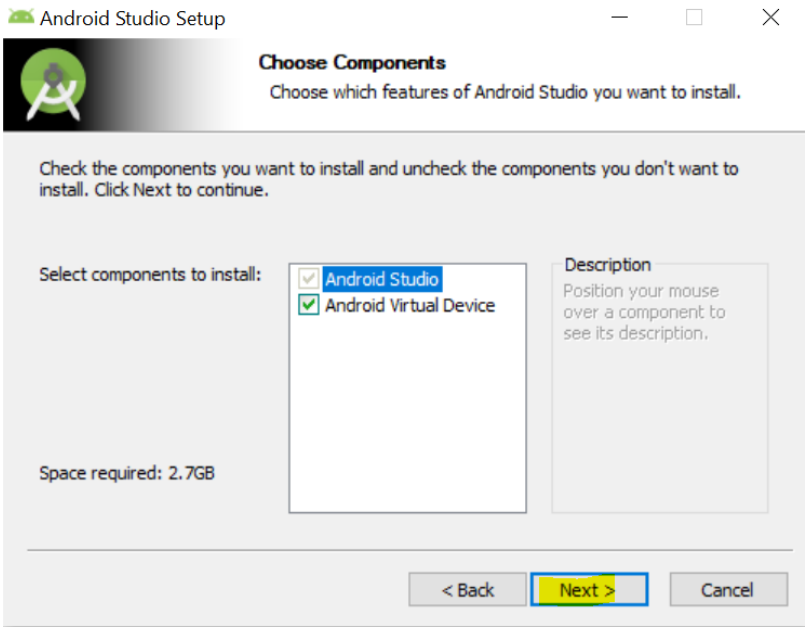


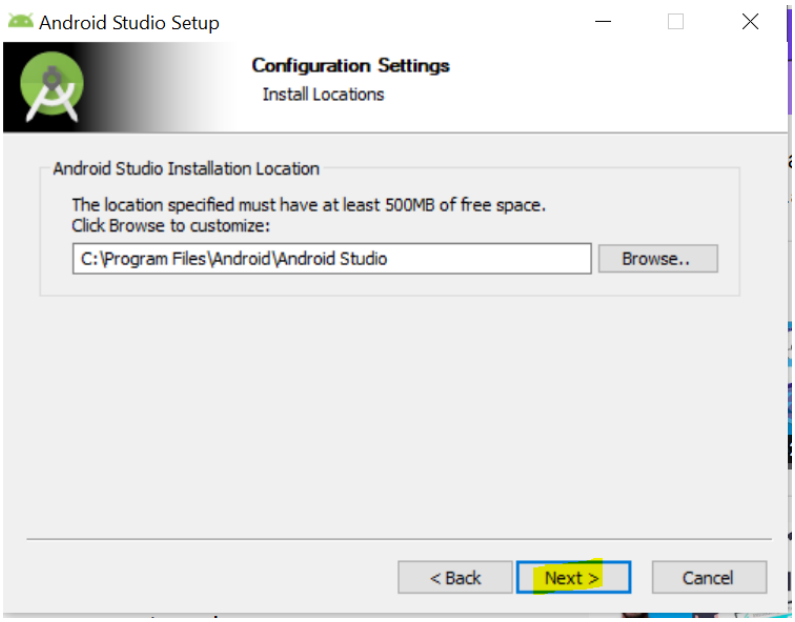
- Double click on the android-studio-ide Application .

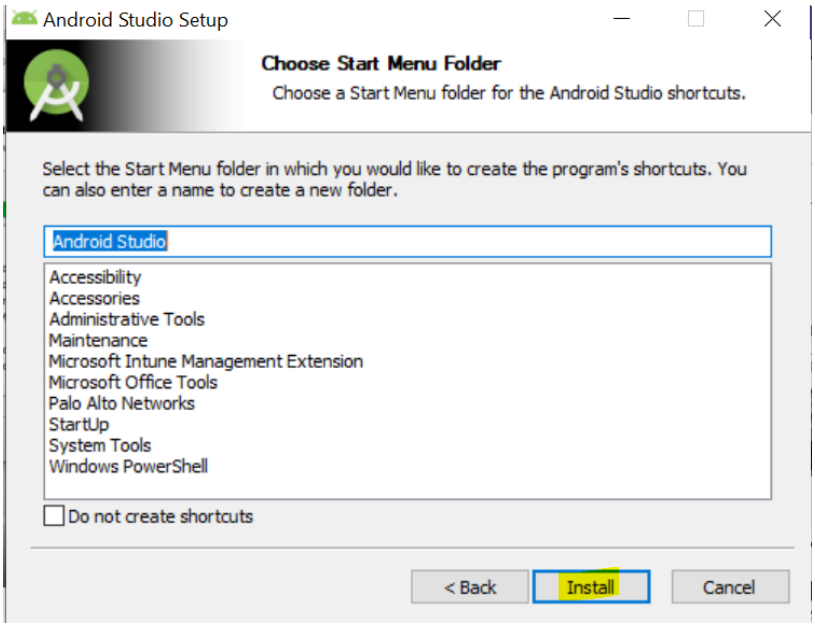


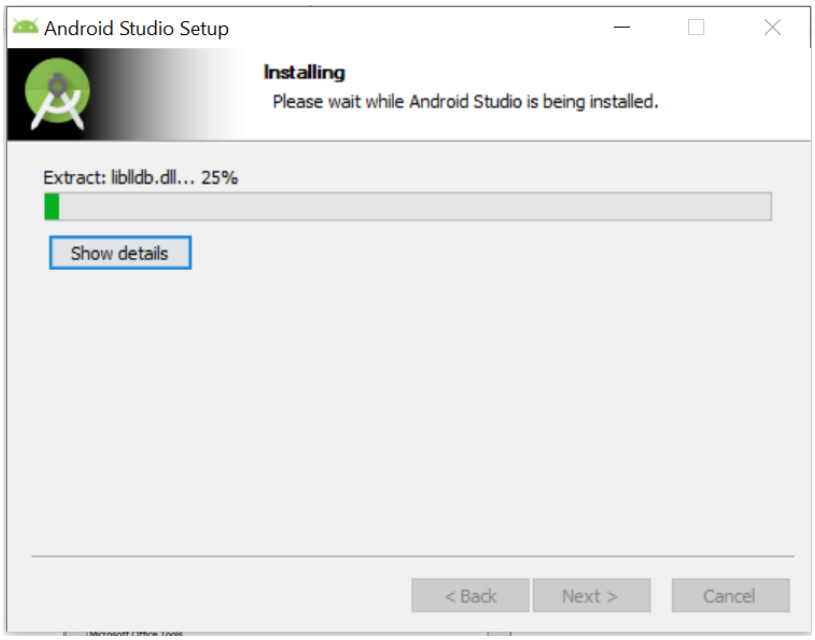
- Click on Yes for making changes for the computer.
- Click on Next button as shown below

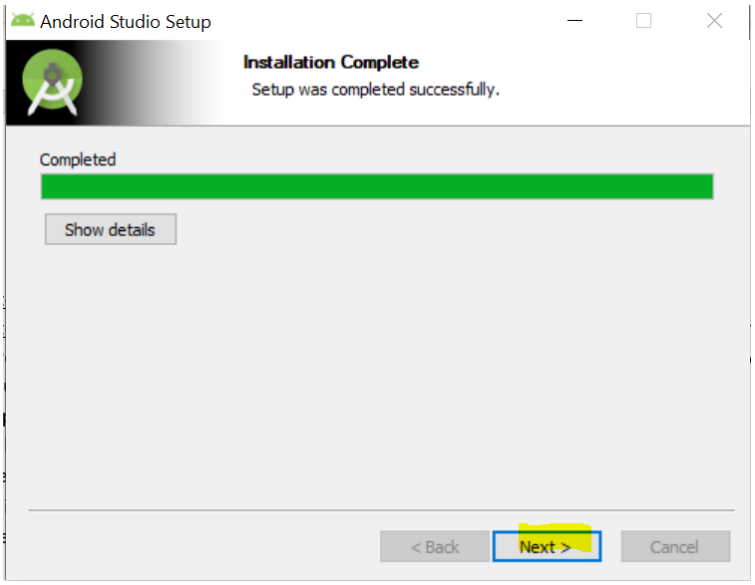


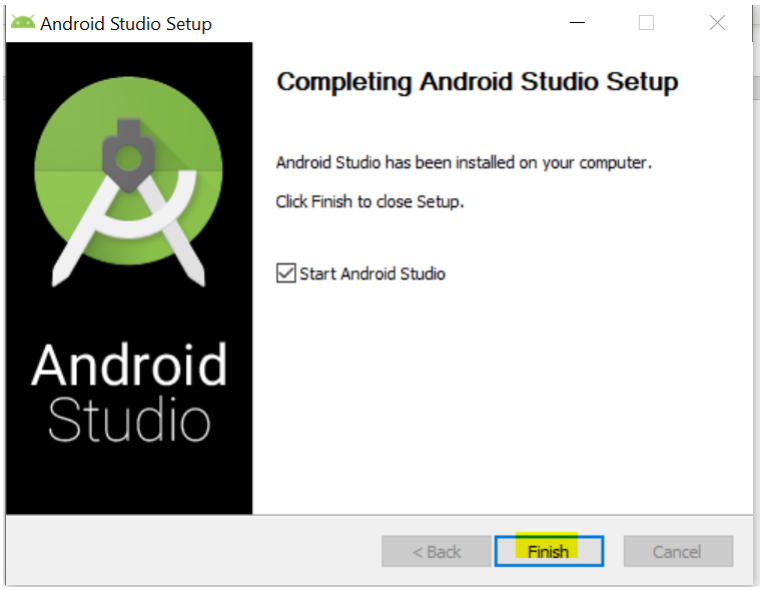




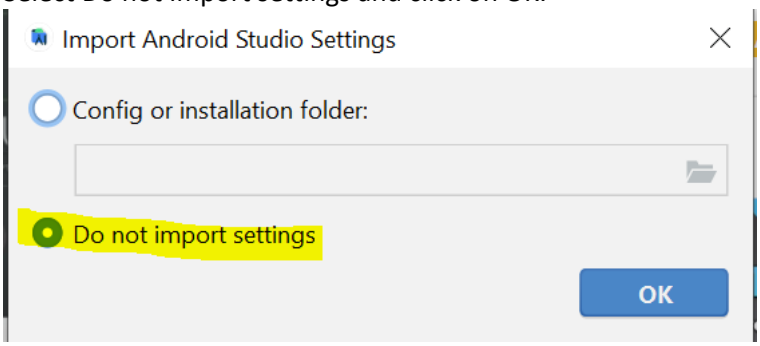




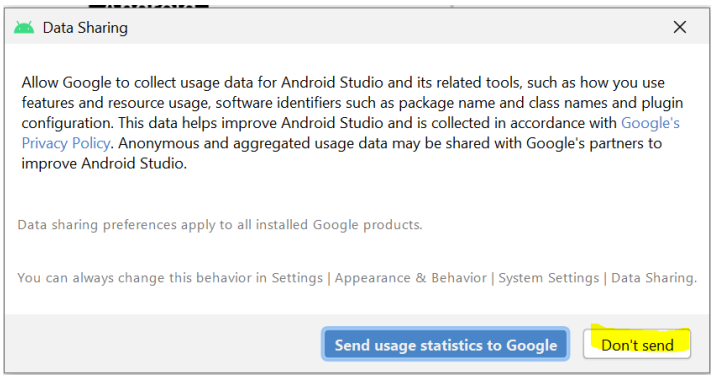




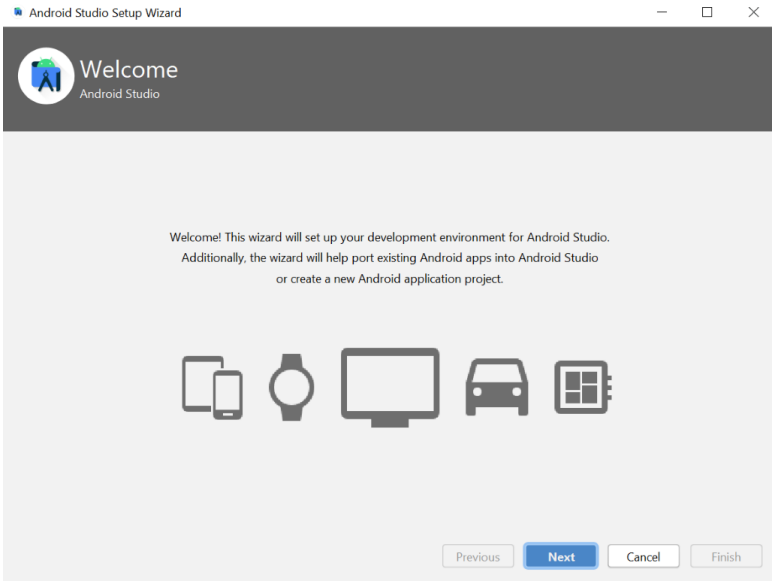
- Select Do not Import settings and click on OK.



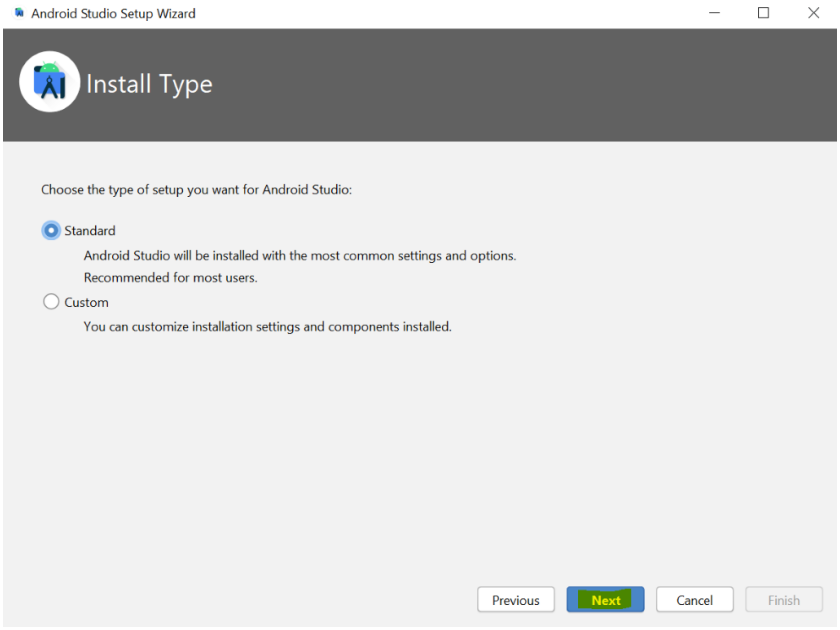
- The Application will start and click on Don’t send



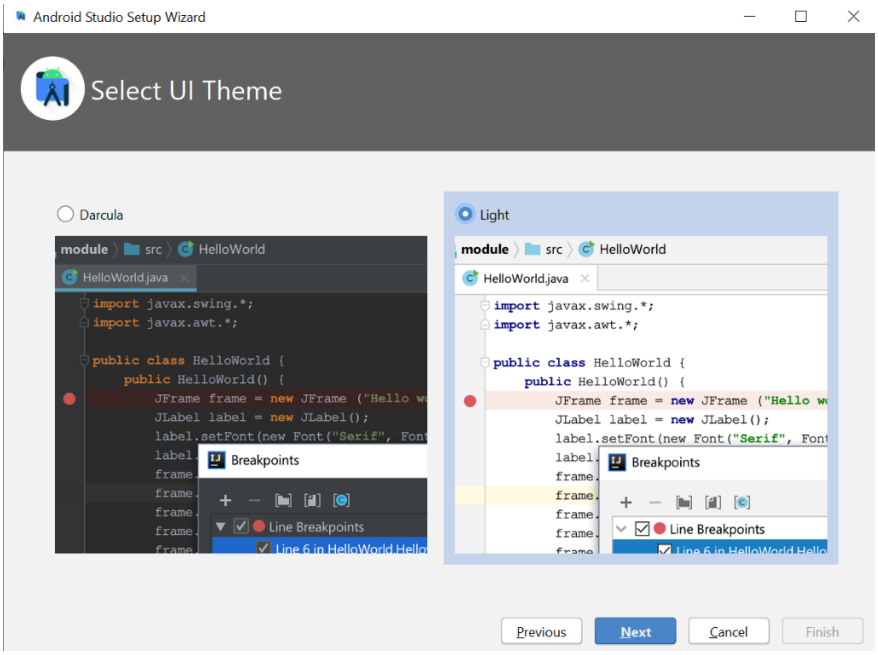
- Click on Next



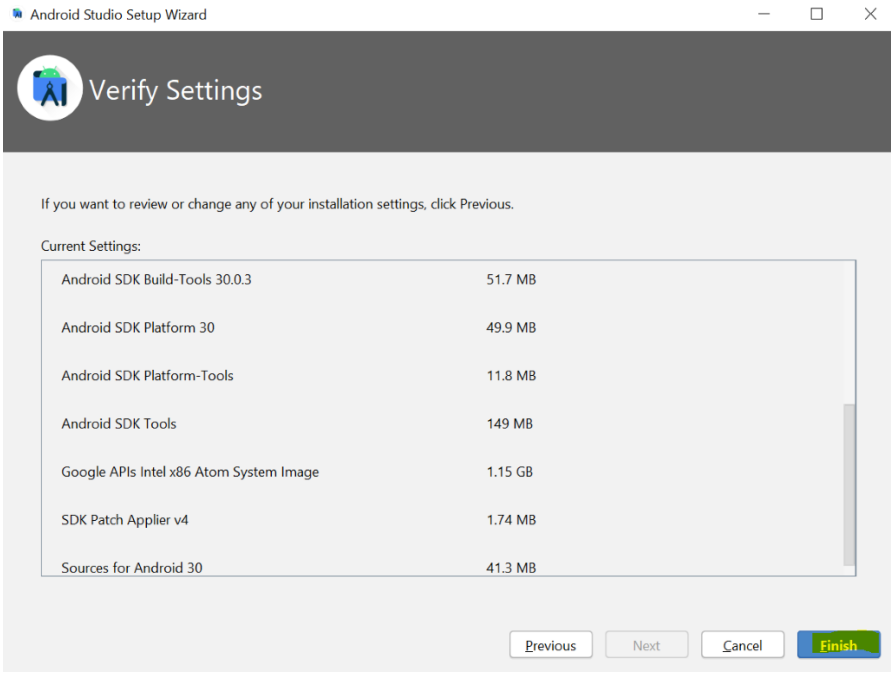
- Select standard and click on Next.



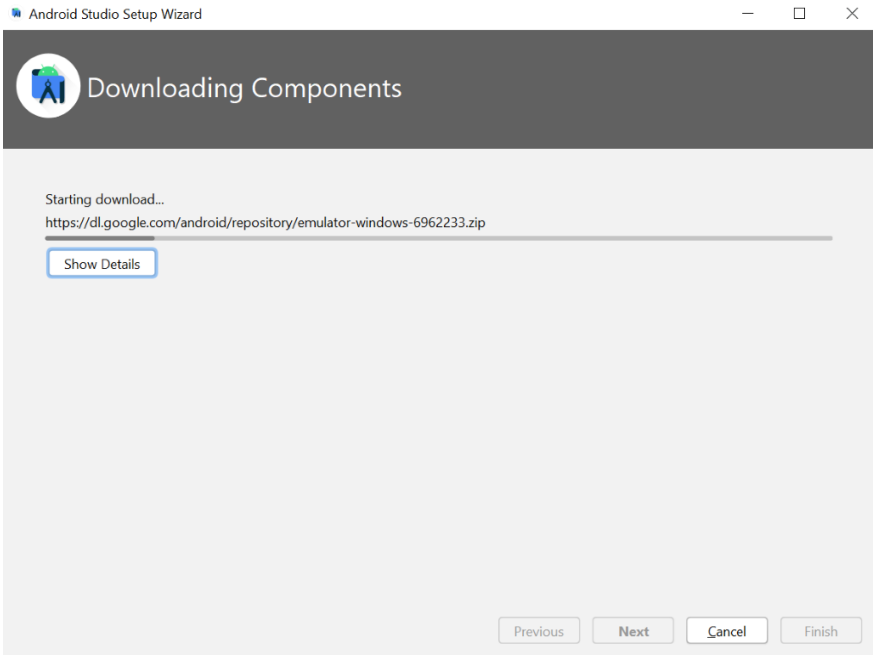
- Select the theme



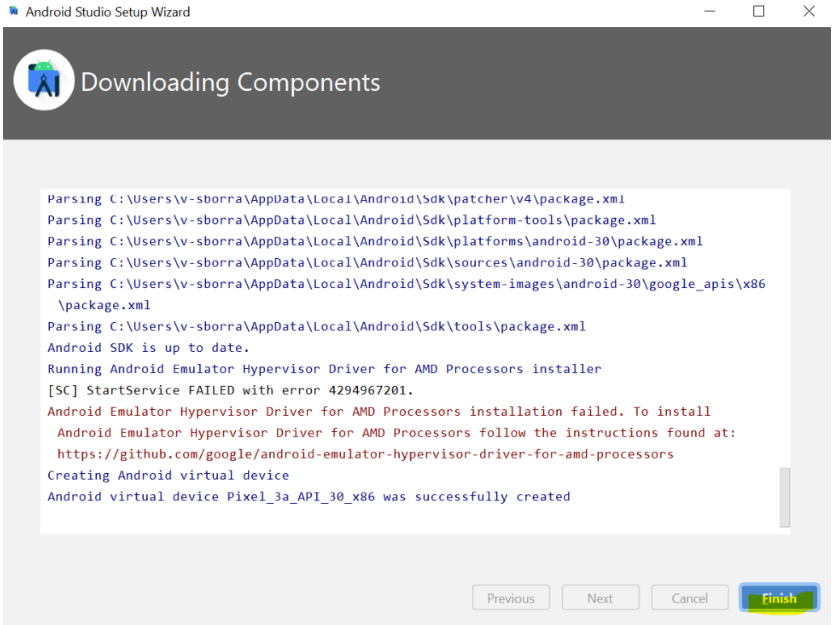
- Click on Finish button



- Will Install the packages as shown below

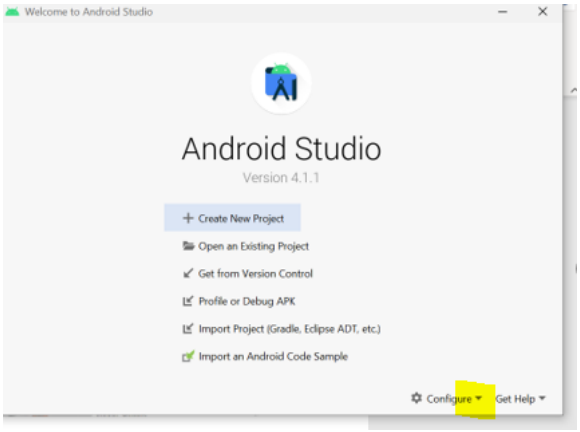


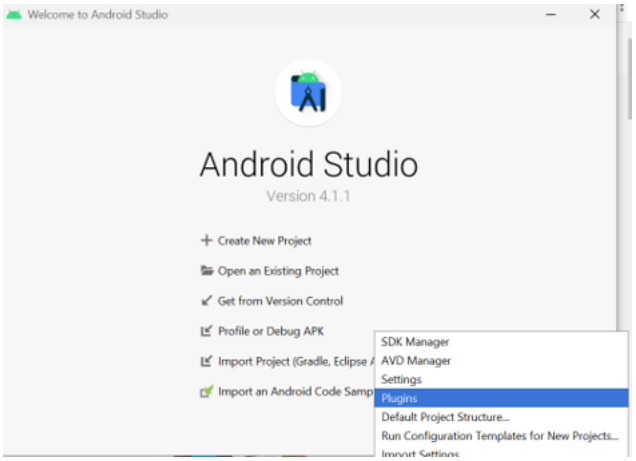
- Click on Yes for the prompt of Allow access to make changes to the device.
- Click on Finish Button



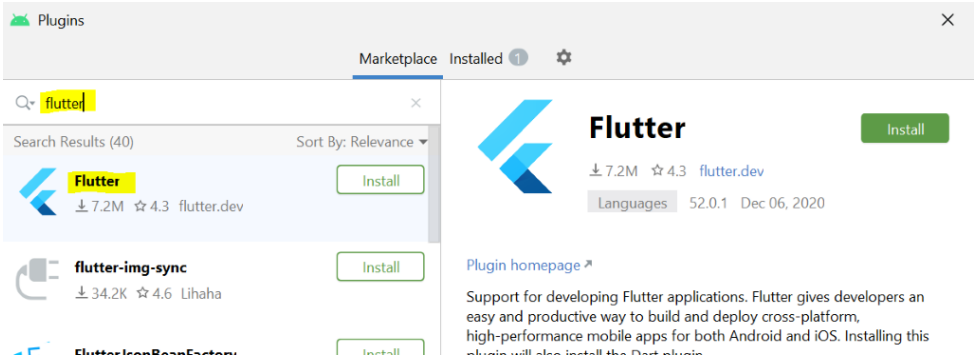
- When checked the “Flutter doctor” command in the cmd prompt .We see that the “Android License is not accepted” , “Flutter Plugin is not Installed” and “Dart Plugin is not installed”.

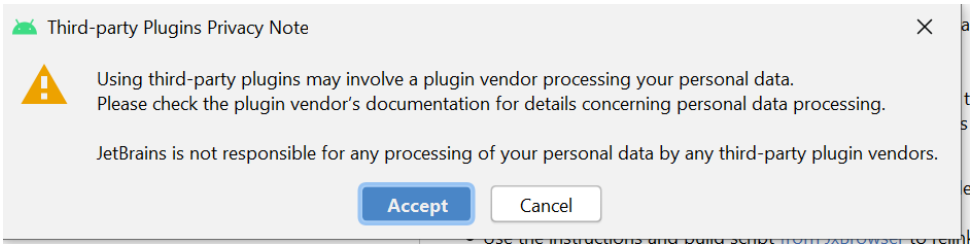
- To Install the Flutter plugin, click on Configure in the right corner of the Android Studio, from the drop down select Plugins as shows in the image below.



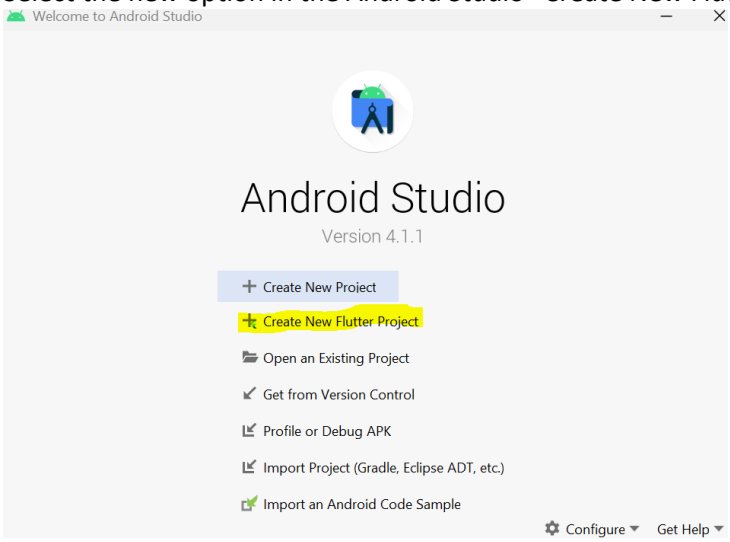


- Click on Install Button for the Flutter as shown in the screenshot. Click on Accept and Install. Click restart IDE to apply changes.

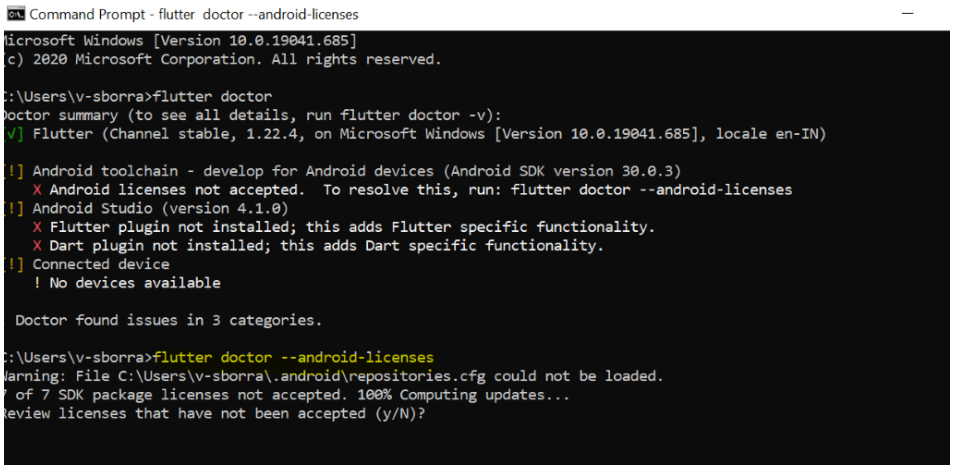




- Select the new option in the Android Studio "Create New Flutter Project"



- In the command prompt type `flutter doctor –android-licenses' as shown in the screen below and accept all licenses by entering `y' for all prompts.



- Type below command `flutter channel master && upgrade && flutter config –enable web` to take latest channel source to enable web/desktop development.

```
C:\Users\v-sborra>flutter channel master && upgrade
Switching to flutter channel 'master'...
git: From https://github.com/flutter/flutter
git:    1aafb3a8b9..7891006299  stable             -> origin/stable
git:  + 198df796aa...022b333a08 beta               -> origin/beta  (forced update)
git:    a0860f6a87..a12e2a473a  dev                -> origin/dev
```

```
C:\Users\v-sborra>flutter config --enable-web
Checking Dart SDK version...
Downloading Dart SDK from Flutter engine 827aa5d073f3551bad2702271b3f02c8178176be...
Building flutter tool...
Running pub upgrade...
```

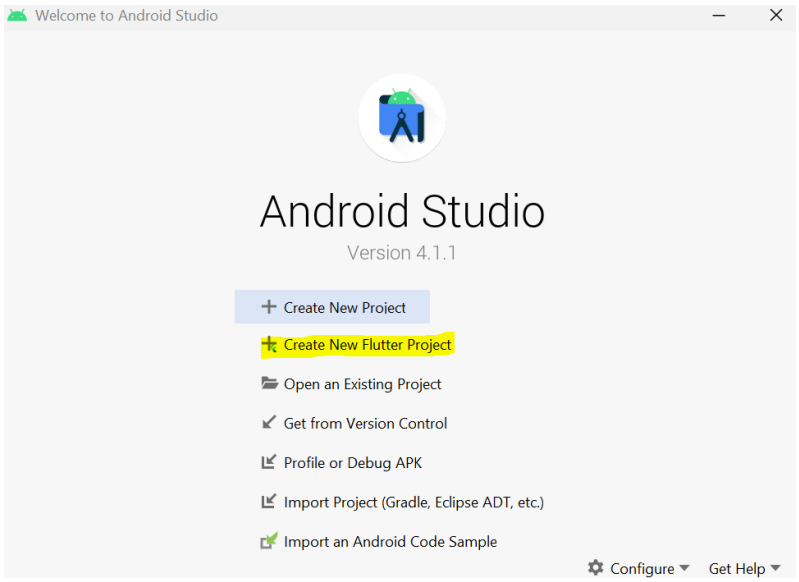
- Run `flutter doctor` again to see all check boxes green

```
C:\Users\v-sborra>flutter doctor
Downloading package sky_engine...                                922ms
Downloading flutter_patched_sdk tools...                       1,500ms
Downloading flutter_patched_sdk_product tools...               1,096ms
Downloading windows-x64 tools...                               2,355ms
Downloading windows-x64/font-subset tools...                     482ms
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel master, 1.25.0-9.0.pre.25, on Microsoft Windows [Version 10.0.19041.685], locale en-IN)
[√] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
[√] Chrome - develop for the web
[√] Android Studio (version 4.1.0)
[√] Connected device (1 available)

• No issues found!
```

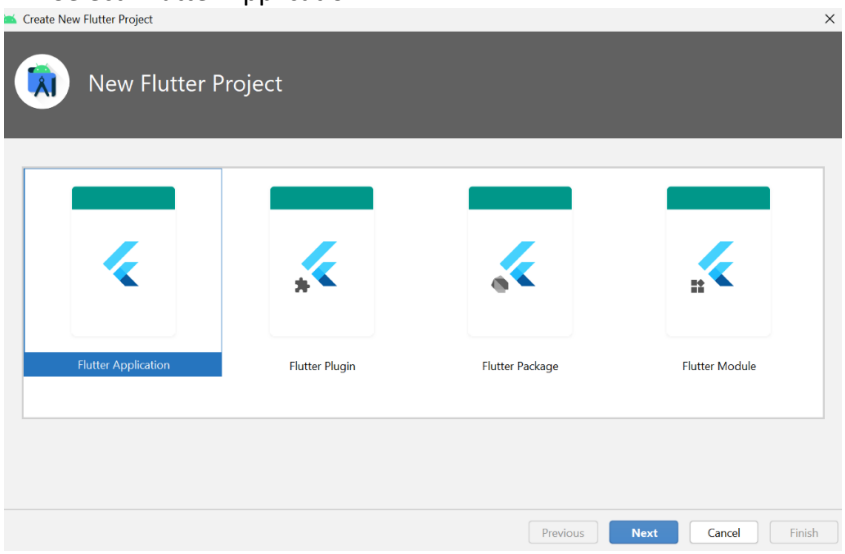
3.2 Flutter – Creating Simple Application in Android Studio

In this chapter, let us create a simple Flutter application to understand the basics of creating a flutter application in the Android Studio.

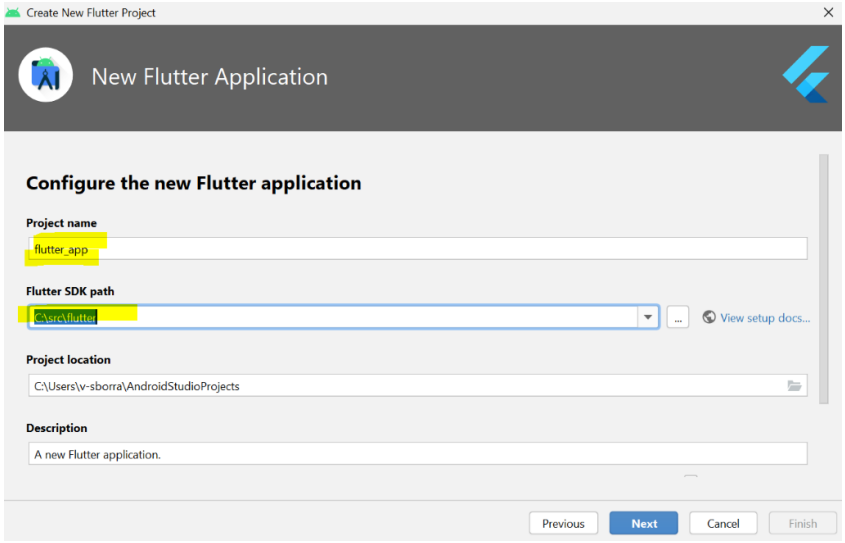
- Launch android studio and select ‘Create New Flutter Project`.



- Select `Flutter Application`.

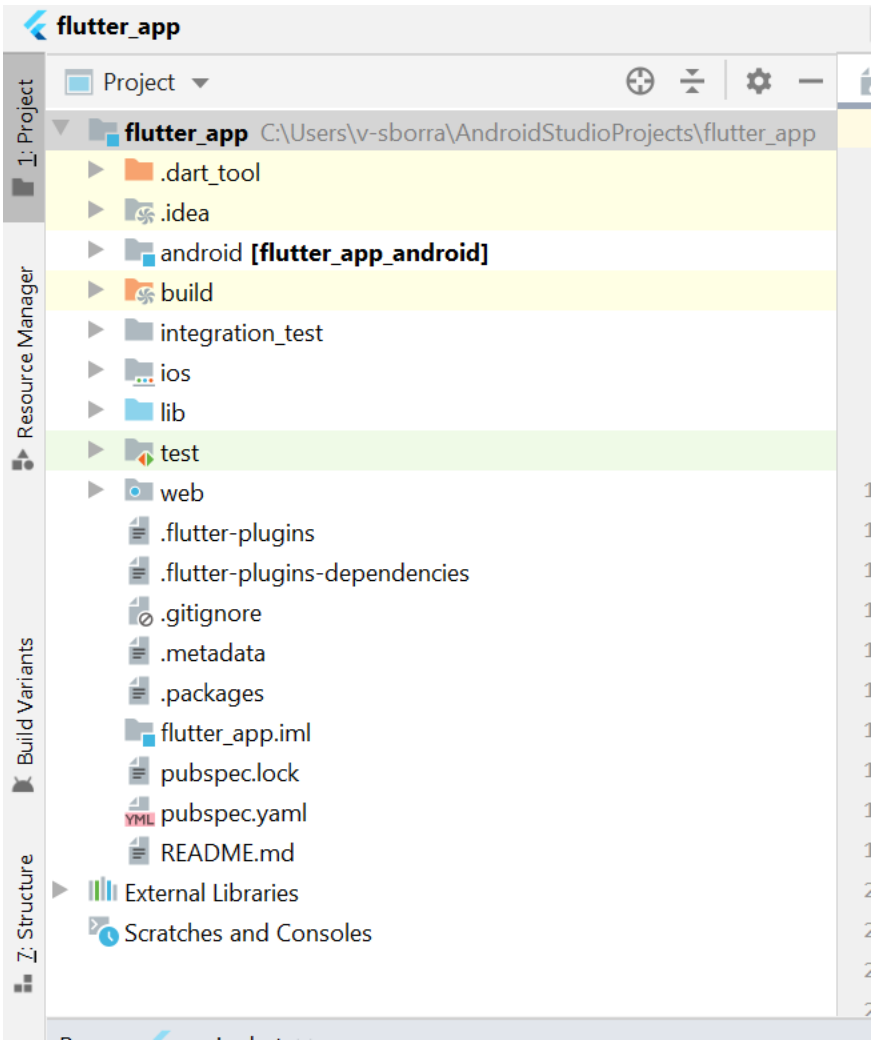


- Name application as `flutter_app` and chose flutter SDK location as below. Chose defaults and click on finish.



Various components of the structure of the application are explained here:

- **android** - Auto generated source code to create android application
- **ios** - Auto generated source code to create ios application
- **lib** - Main folder containing Dart code written using flutter framework
- **lib/main.dart** - Entry point of the Flutter application
- **test** - Folder containing Dart code to test the flutter application
- **test/widget_test.dart** - Sample code
- **.gitignore** - Git version control file
- **.metadata** - auto generated by the flutter tools
- **.packages** - auto generated to track the flutter packages
- **.iml** - project file used by Android studio
- **pubspec.yaml** - Used by **Pub**, Flutter package manager
- **pubspec.lock** - Auto generated by the Flutter package manager, **Pub**
- **README.md** - Project description file written in Markdown format



- Below default example the dart code is present in *lib/main.dart*:
- `import 'package:flutter/material.dart';`

```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app,
try
        // changing the primarySwatch below to Colors.green and then
invoke
```

```
        // "hot reload" (press "r" in the console where you ran "flutter
run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the
application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  // This widget is the home page of your application. It is stateful,
meaning
  // that it has a State object (defined below) that contains fields that
affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in
this
  // case the title) provided by the parent (in this case the App widget)
and
  // used by the build method of the State. Fields in a Widget subclass
are
  // always marked "final".

  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something
has
      // changed in this State, which causes it to rerun the build method
below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would
not be
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }

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

```
    // This method is rerun every time setState is called, for instance as
done
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunning build
methods
    // fast, so that you can just rebuild anything that needs updating
rather
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage object that was
created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ),
      body: Center(
        // Center is a layout widget. It takes a single child and positions
it
        // in the middle of the parent.
        child: Column(
          // Column is also a layout widget. It takes a list of children
and
          // arranges them vertically. By default, it sizes itself to fit
its
          // children horizontally, and tries to be as tall as its parent.
          //
          // Invoke "debug painting" (press "p" in the console, choose the
          // "Toggle Debug Paint" action from the Flutter Inspector in
Android
          // Studio, or the "Toggle Debug Paint" command in Visual Studio
Code)
          // to see the wireframe for each widget.
          //
          // Column has various properties to control how it sizes itself
and
          // how it positions its children. Here we
use mainAxisAlignment to
          // center the children vertically; the main axis here is the
vertical
          // axis because Columns are vertical (the cross axis would be
          // horizontal).
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
```

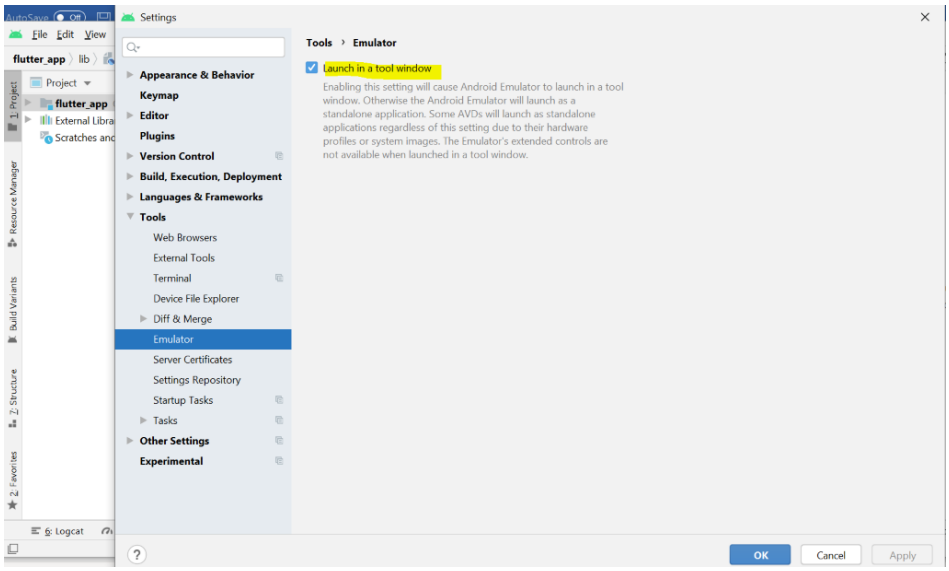
```
            onPressed: _incrementCounter,
            tooltip: 'Increment',
            child: Icon(Icons.add),
          ), // This trailing comma makes auto-formatting nicer for build
    methods.
        );
      }
    }
    •
```

[Text Wrapping Break]

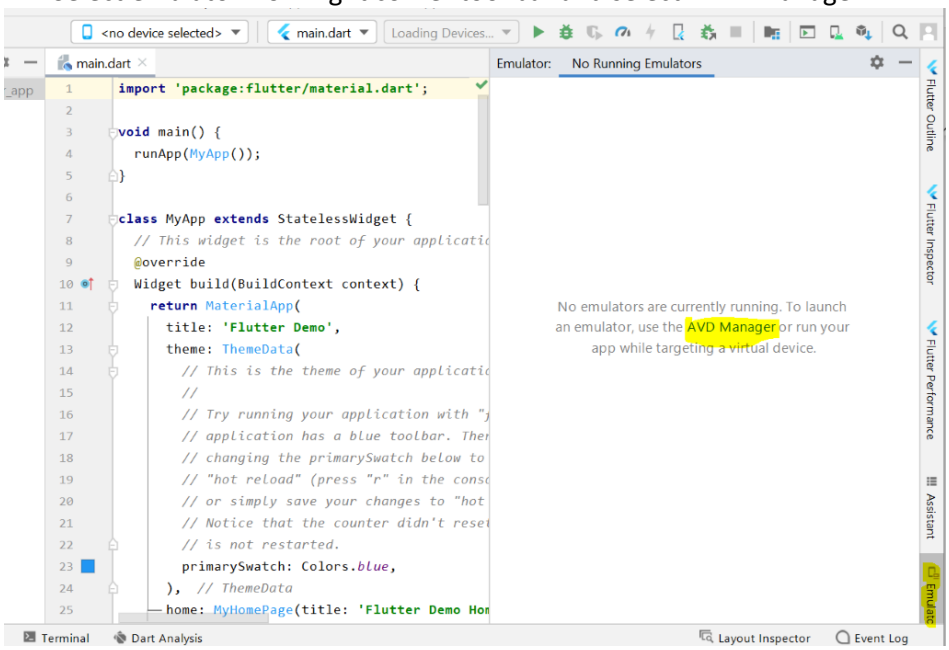
Let us understand the dart code line by line.

- imports the flutter package, *material*. The *material* is a flutter package to create user interface according to the Material design guidelines specified by Android.
- This is the entry point of the Flutter application. Calls *runApp* function and pass it an object of *MyApp* class. The purpose of the *runApp* function is to attach the given widget to the screen.
- *Widget* is used to create UI in flutter framework. *StatelessWidget* is a widget, which does not maintain any state of the widget. *MyApp* extends *StatelessWidget* and overrides its *build* method. The purpose of the *build* method is to create a part of the UI of the application. Here, *build* method uses *MaterialApp*, a widget to create the root level UI of the application. It has three properties - *title*, *theme* and *home*.
  - *title* is the title of the application.
  - *theme* is the theme of the widget. Here, we set *blue* as the overall color of the application using *ThemeData* class and its property, *primarySwatch*.
  - *home* is the inner UI of the application, which we set another widget, *MyHomePage*
- *MyHomePage* is same as *MyApp* except it returns *Scaffold* Widget. *Scaffold* is a top level widget next to *MaterialApp* widget used to create UI conforming material design. It has two important properties, *appBar* to show the header of the application and *body* to show the actual content of the application. *AppBar* is another widget to render the header of the application and we have used it in *appBar* property. In *body* property, we have used *Center* widget, which centers it child widget. *Text* is the final and inner most widget to show the text and it is displayed in the center of the screen.

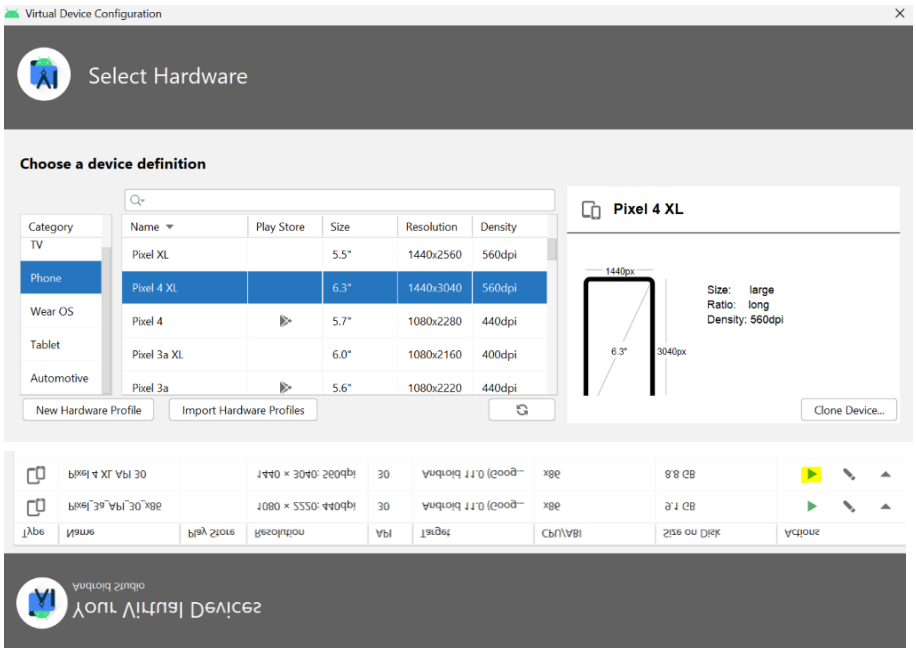
- Go to setting from `file → settings` and select `Launch in tools window` as shown below.



- Select emulator from right corner tool bar and select `AVD Manager`.



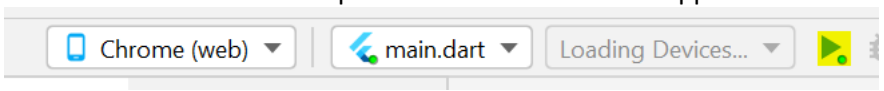
- Select one of the devices of your choice and click green run button.

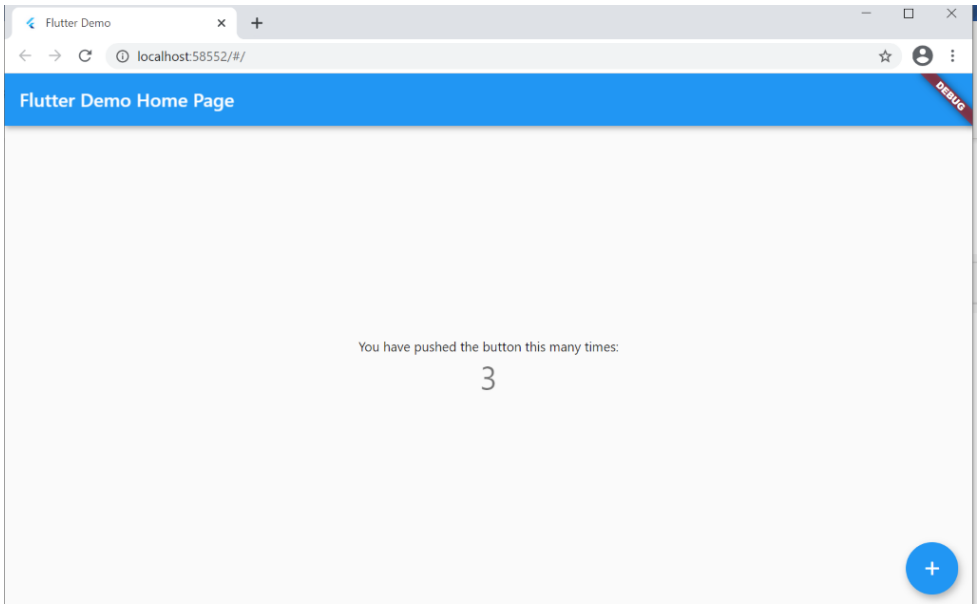


- Run flutter devices to show all available devices. These include web/desktop/mobile.

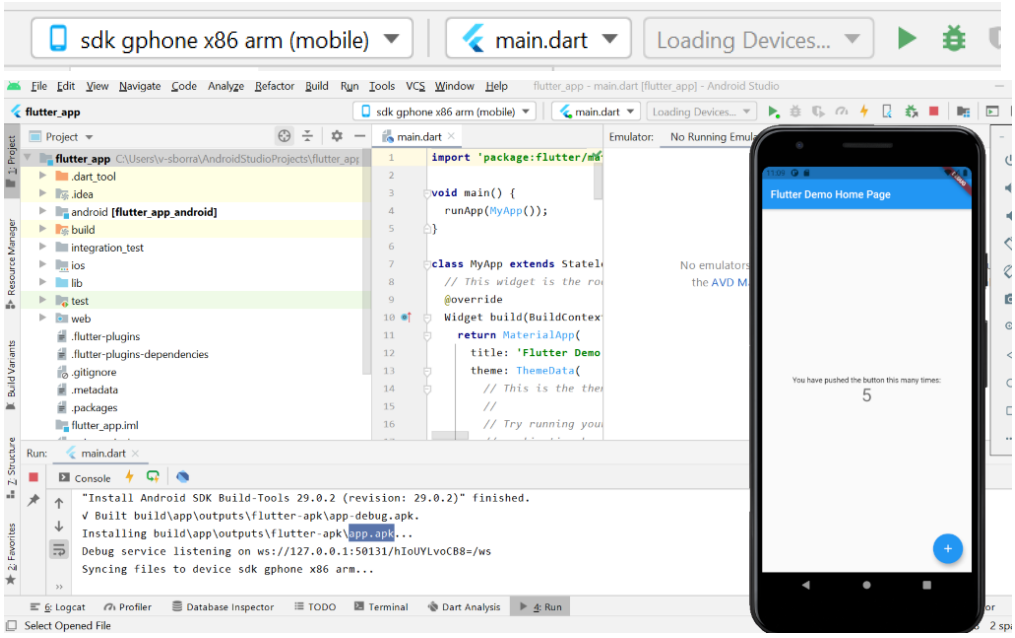
```
C:\Users\v-sborra>flutter devices
2 connected devices:

sdk gphone x86 arm (mobile) • emulator-5554 • android-x86    • Android 11 (API 30) (emulator)
Chrome (web)                • chrome        • web-javascript • Google Chrome 87.0.4280.88

C:\Users\v-sborra>
```

- Select Chrome from dropdown and click run to see application in browser.





- Select mobile device and click run.

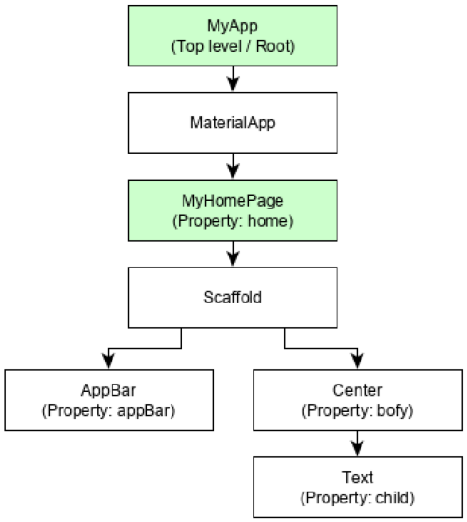


Step 8: (Optional steps) Start an android emulator or connect a real android device to the system.

3.3 Flutter – Architecture of Flutter Application

The core concept of the Flutter framework is In Flutter, Everything is a widget. Widgets are basically user interface components used to create the user interface of the application.

For example, the widget hierarchy of the hello world application (created in previous chapter) is as specified in the following diagram:



- MyApp is the user created widget and it is build using the Flutter native widget, MaterialApp.

- MaterialApp has a home property to specify the user interface of the home page, which is again a user created widget, MyHomePage.
- MyHomePage is build using another flutter native widget, Scaffold.
- Scaffold has two properties – body and appBar.
- body is used to specify its main user interface and appBar is used to specify its header user interface.
- Header UI is build using flutter native widget, AppBar and Body UI is build using Center widget.
- The Center widget has a property, Child, which refers the actual content and it is build using Text widget.

Appendix A: Application Development Requirements.

- App Development : Project Requirements

- GGast: Questions for Mr. Benjamin