Academic Year: 2021-22 **Semester: VI**

Name: Nency Batada, Rollno: 1, Class / Branch: TE IT
Subject: MAD & PWA Lab

EXPERIMENT NO. 1

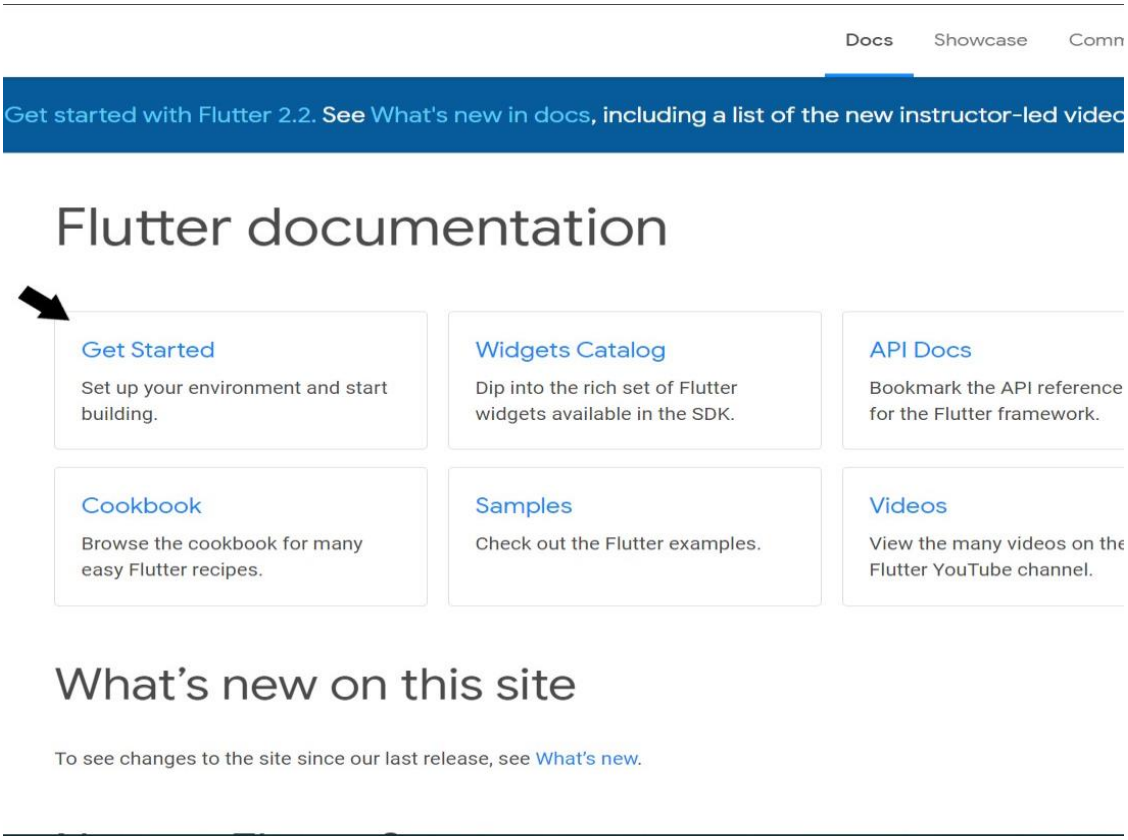
Aim: To install and configure Flutter Environment.

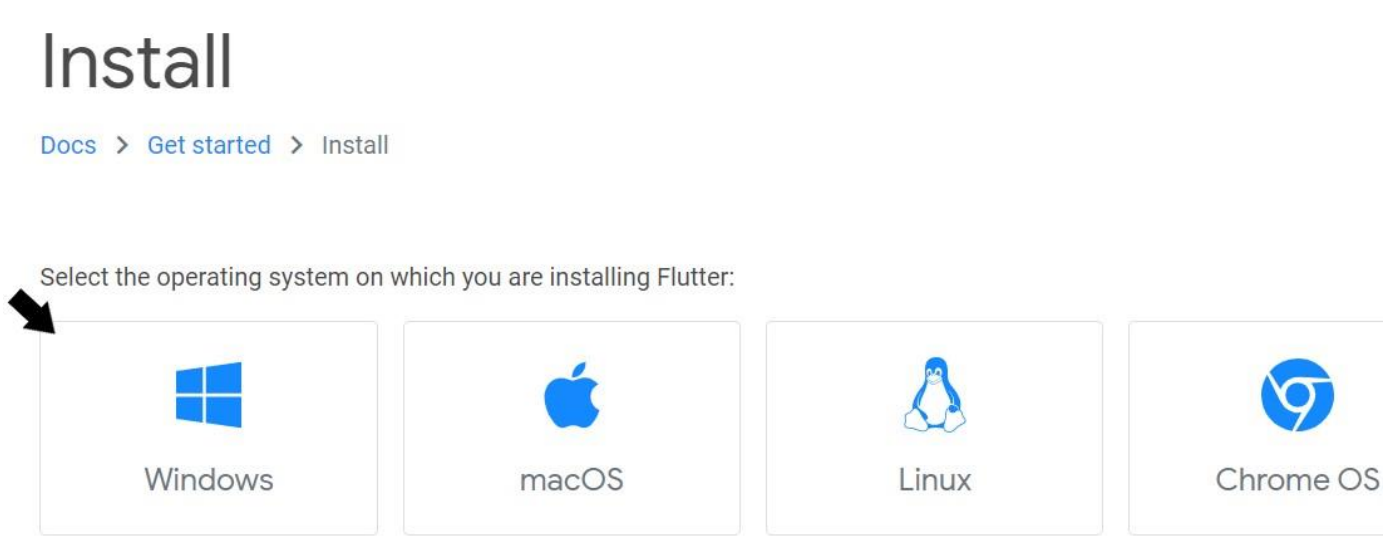
Theory:

Flutter is basically Google's portable user interface (UI) toolkit, used to build and develop eyecatching, natively-built applications for mobile, desktop, and web, from a single codebase. Flutter is free, open-sourced, and compatible with existing code. It is utilized by companies and developers around the world, due to its user-friendly interface and fairly simple, yet to-the-point commands.

Steps:

Step 1: Navigate to flutter.dev on your webpage. On the top menu bar, select Docs > Get Started > Install > Windows.





Step 2: Check for the System Requirements. Henceforth, you can begin the installation.

Windows install

Docs > Get started > Install > Windows

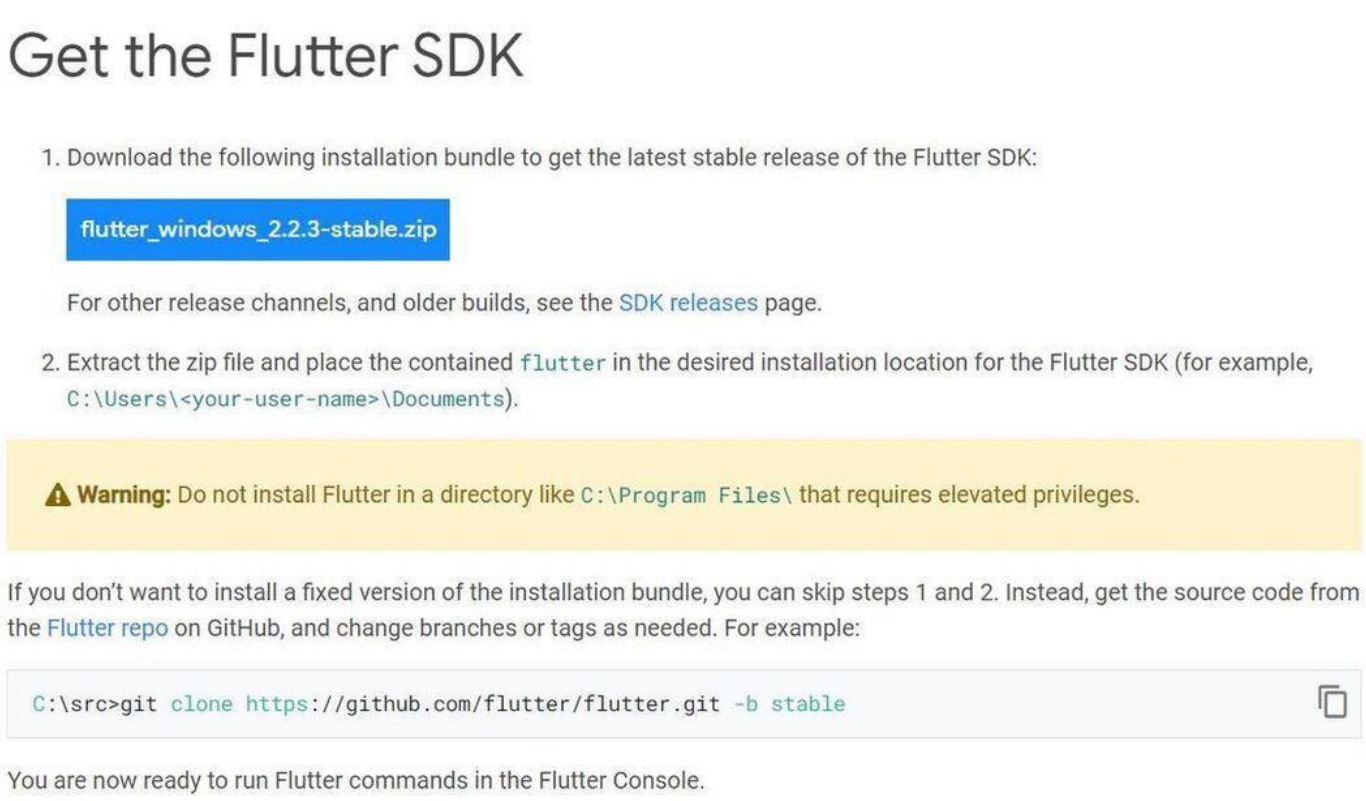
System requirements

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**: 1.64 GB (does not include disk space for IDE/tools).
- **Tools**: Flutter depends on these tools being available in your environment.
  - Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
  - Git for Windows 2.x, with the **Use Git from the Windows Command Prompt** option.

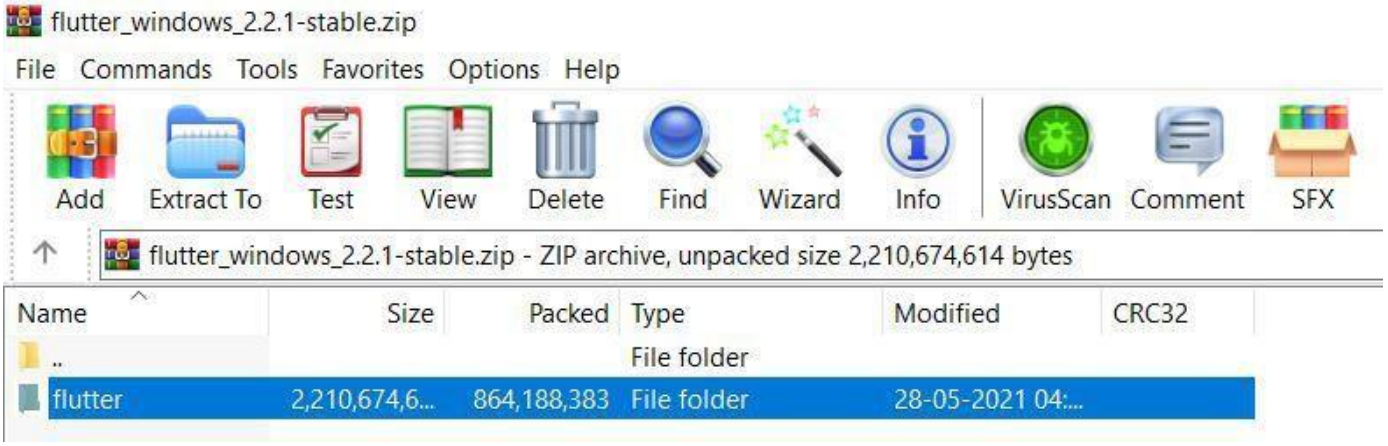
    If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell.

Step 3: Restart the system after installing Git on your windows. Once done, let's get to the installation of Flutter Software development Kit (Flutter SDK). Click on the download link for the latest version (as of today).

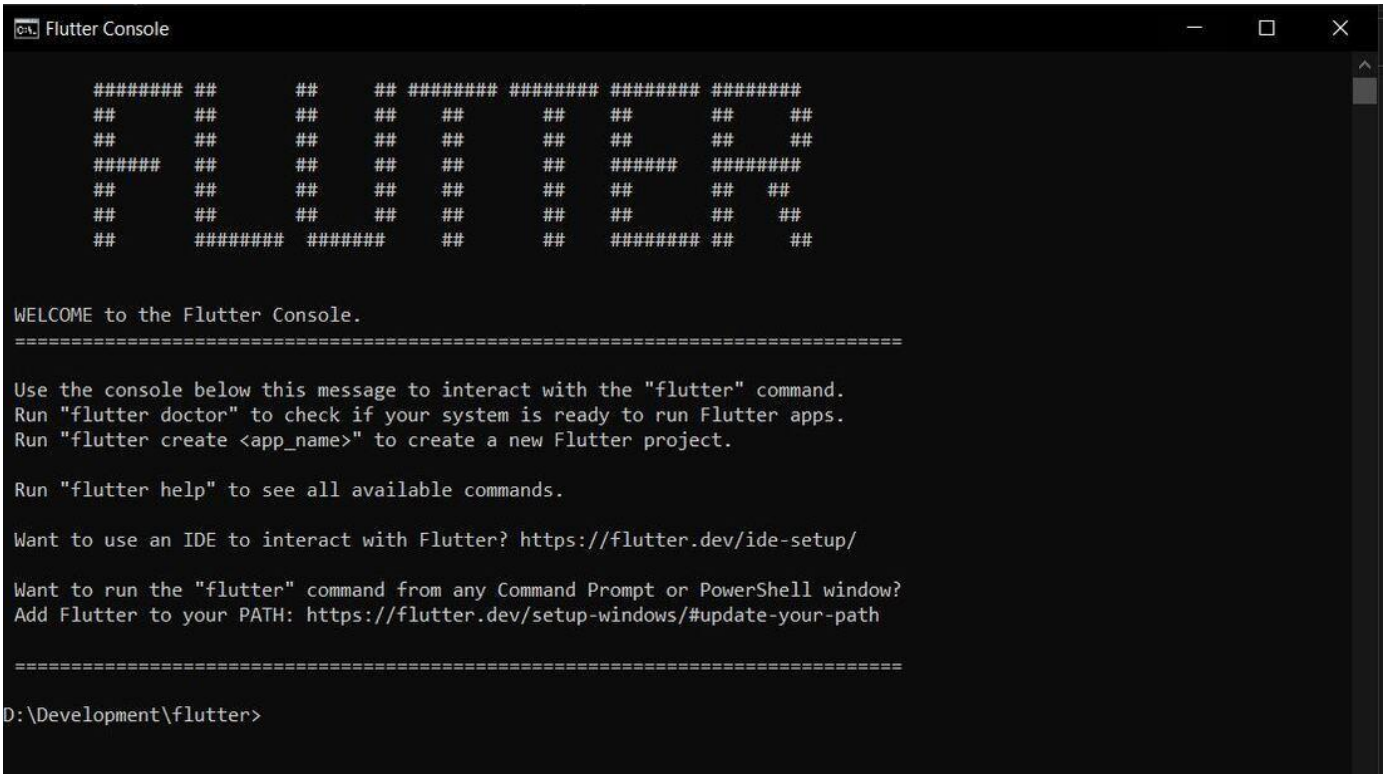


Flutter SDK is the tool that not only allows us to create flutter projects but also build those projects and transform them into native mobile applications. In simpler words, Flutter SDK is the core tool for building a flutter UI.

Once the zip file is downloaded, extract the 'flutter' folder (drag and drop) to any path/directory of the system where you get the read and write access. Typically, it is better to create a new folder in a separate directory apart from the system drive due to permission issues (In my case, the target destination is D: > development > flutter).



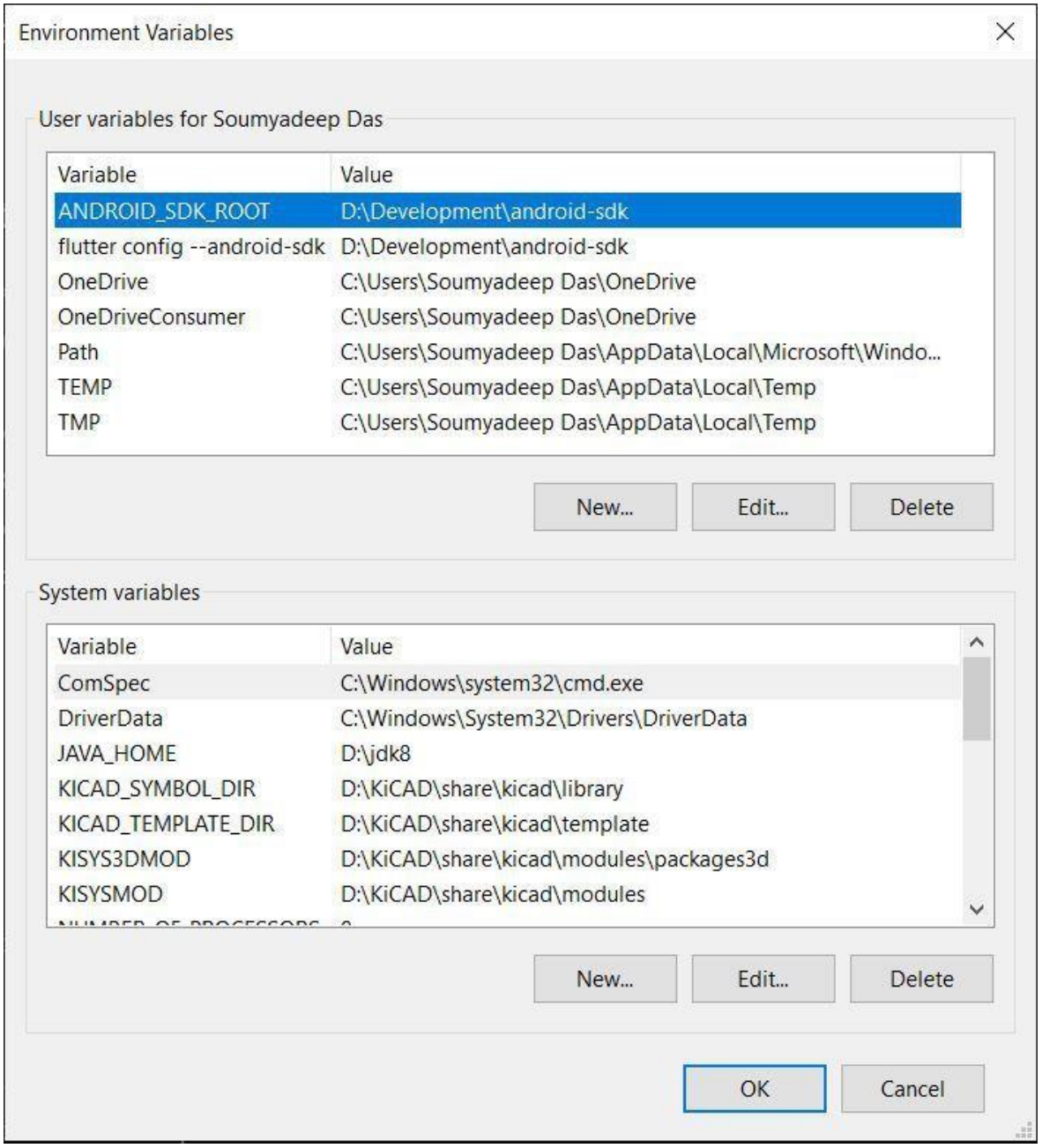
Now double-click on the 'flutter' folder. Go to 'flutter_console.bat' file and double-click to open a command prompt window. It should look something like this:



This console is actually a Windows terminal available for the developer to run flutter commands. Type in 'flutter' to get a list of all the flutter commands that can be run.

Whilst it is pretty good to have a terminal to execute flutter commands and create projects, it'd still be better and more convenient to store all our flutter projects somewhere else on our system for easy access. Let us steer over to the next step of our journey!

Step 4: Check and edit environment variables for global system access. For this, scroll down to 'Update your path' on the official Docs page of the flutter installation page. For this, go to Control Panel > System and Security > System > Advanced System Settings > Environment Variables… . A dialog box displaying a list of the available environment variables appears on your screen.

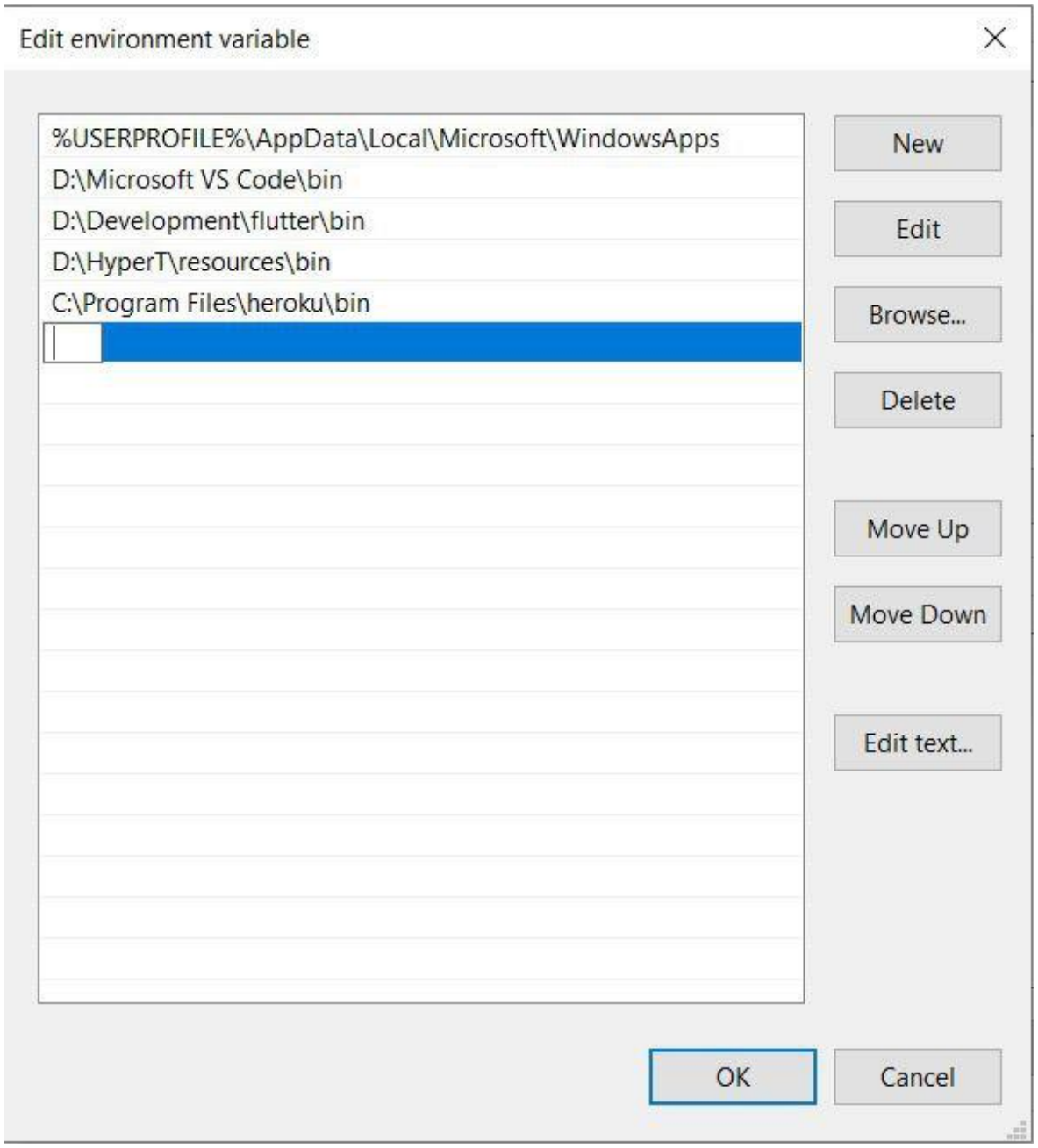


Environment Variables are global system variables present at the root level, which aids in configuring various aspects of Windows. We will now add the flutter tool as an environment variable for direct access (instead of running the .bat executable), and unlock it on the entire PowerShell and Command Prompt of your system.

To do this, glance through the following steps:

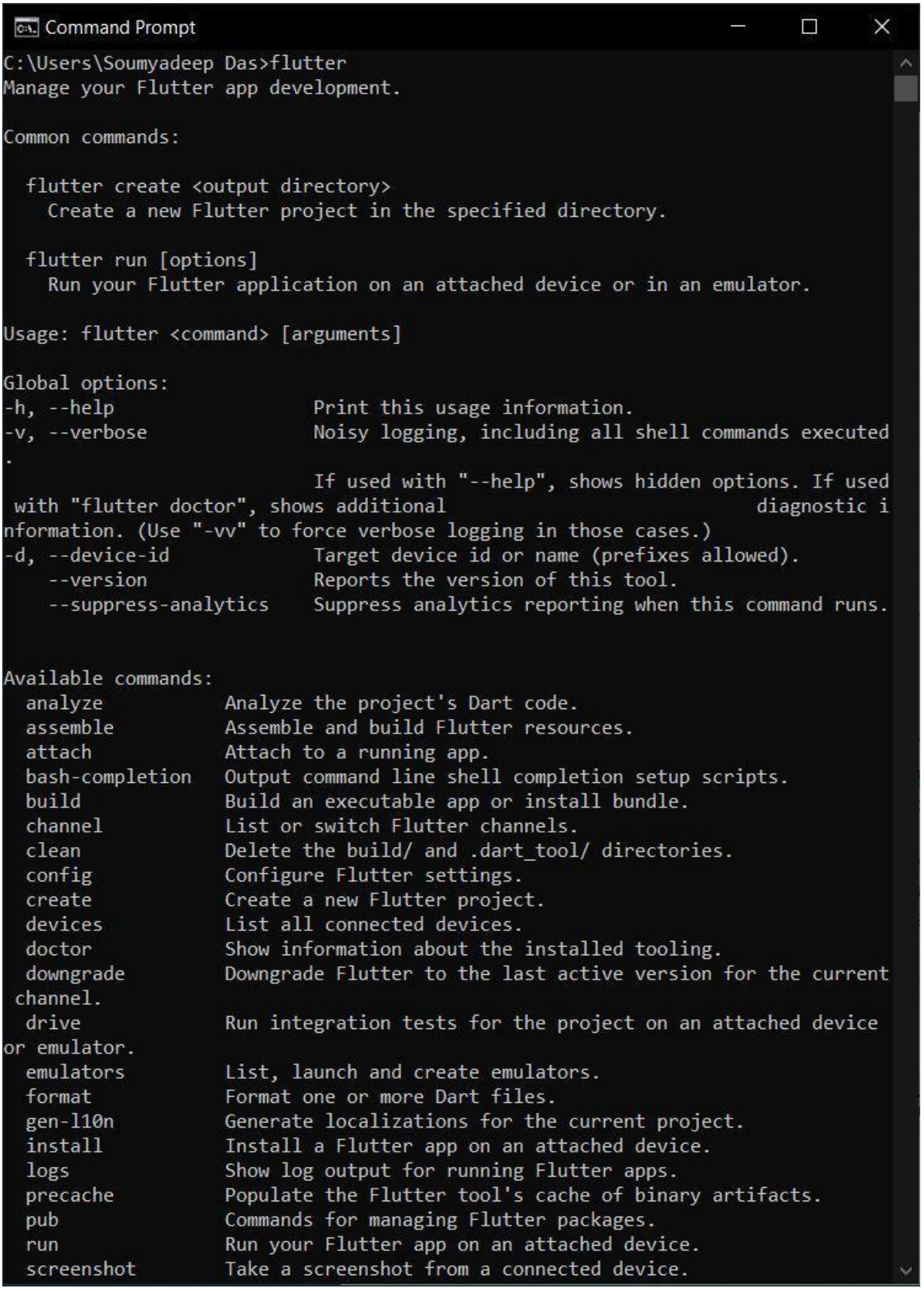
Check for 'Path' variable under User Variables list. If not already present, create a new variable ('New…') and assign the 'flutter\bin' directory as its value.

Now double-click on the 'Path' variable and add a new entry by double-clicking on a column below. It should look something like this:



In the path, copy the entire directory of flutter\bin folder and paste it. Click 'Ok' twice to complete the setup. Now, make sure that you have closed any existing Command Prompt/Windows PowerShell windows that are open.

Now, check whether your flutter framework can be accessed globally. To do this, open any terminal (say Command Prompt) and type in 'flutter' and see whether you get the same list of commands as you did get earlier from the .bat terminal. If yes, you have successfully completed setting up flutter on the root level in your system. If not, you might as well consider re-running the setup again.


Command Prompt
C:\Users\Soumyadeep Das>flutter
Manage your Flutter app development.
Common commands:
flutter create <output directory>
Create a new Flutter project in the specified directory.
flutter run [options]
Run your Flutter application on an attached device or in an emulator.
Usage: flutter <command> [arguments]
Global options:
-h, --help Print this usage information.
-v, --verbose Noisy logging, including all shell commands executed
.
If used with "--help", shows hidden options. If used
with "flutter doctor", shows additional diagnostic i
nformation. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id Target device id or name (prefixes allowed).
--version Reports the version of this tool.
--suppress-analytics Suppress analytics reporting when this command runs.
Available commands:
analyze Analyze the project's Dart code.
assemble Assemble and build Flutter resources.
attach Attach to a running app.
bash-completion Output command line shell completion setup scripts.
build Build an executable app or install bundle.
channel List or switch Flutter channels.
clean Delete the build/ and .dart_tool/ directories.
config Configure Flutter settings.
create Create a new Flutter project.
devices List all connected devices.
doctor Show information about the installed tooling.
downgrade Downgrade Flutter to the last active version for the current
channel.
drive Run integration tests for the project on an attached device
or emulator.
emulators List, launch and create emulators.
format Format one or more Dart files.
gen-l10n Generate localizations for the current project.
install Install a Flutter app on an attached device.
logs Show log output for running Flutter apps.
precache Populate the Flutter tool's cache of binary artifacts.
pub Commands for managing Flutter packages.
run Run your Flutter app on an attached device.
screenshot Take a screenshot from a connected device.

Step 5: Now, you have to analyze and check whether something is missing/has to be installed further. To do this, under the Command Prompt terminal, type in 'Flutter Doctor' to check for other requirements.

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 example:

```
[-] Android toolchain - develop for Android devices
    • Android SDK at D:\Android\sdk
    X Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    • Try re-installing or updating your Android SDK,
      visit https://flutter.dev/setup/#android-setup for detailed instructions.
```

The following sections describe how to perform these tasks and finish the setup process. Once you have installed any missing dependencies, you can run the `flutter doctor` command again to verify that you've set everything up correctly.

(Since a version has already been installed on my computer, below is an image shown from a previous version, to help you get an understanding of the 'errors' that appear after flutter doctor analysis.)

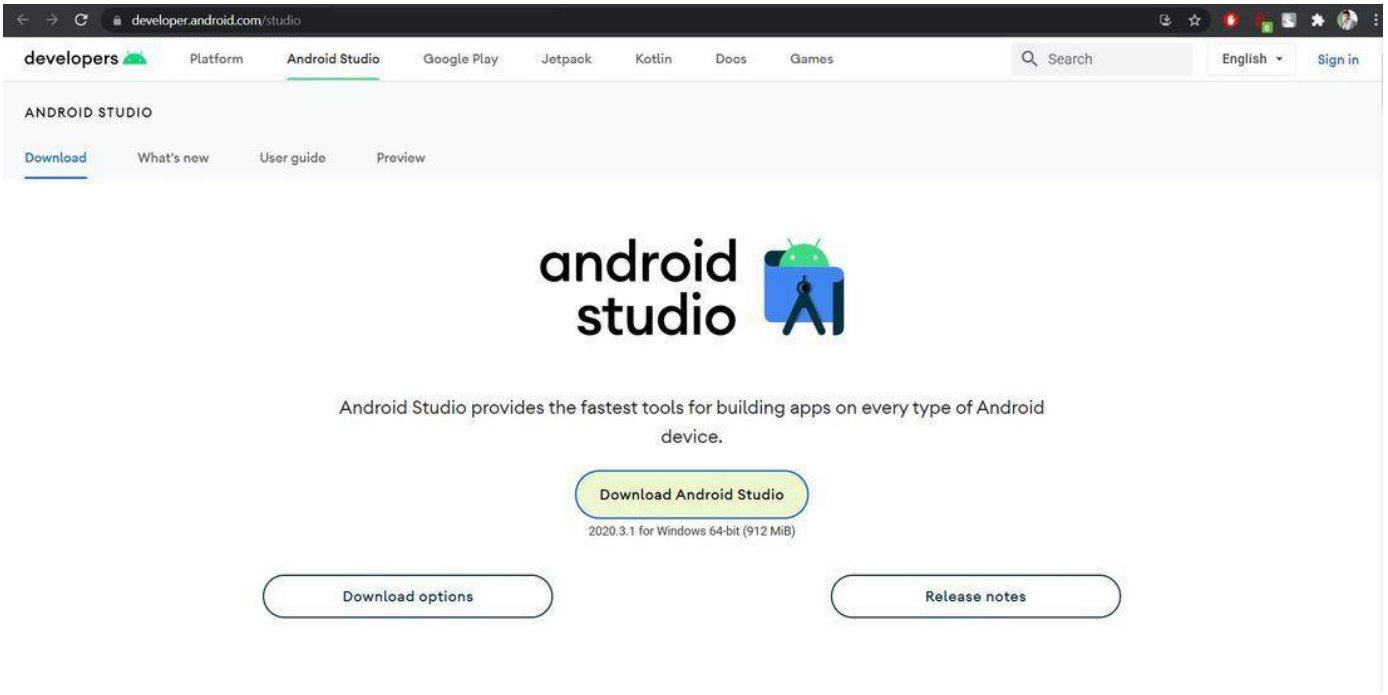
```
[√] Flutter (Channel stable, v1.17.3, on Microsoft Windows [Version
    10.0.18362.900], locale en-US)
[!] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it 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_HOME
      to that location.
      You may also want to add it to your PATH environment variable.

    X No valid Android SDK platforms found in
      C:\Users\maxschwarzmueller\AppData\Local\Android\sdk\platforms. Directory was
      empty.
[!] Android Studio (not installed)
```

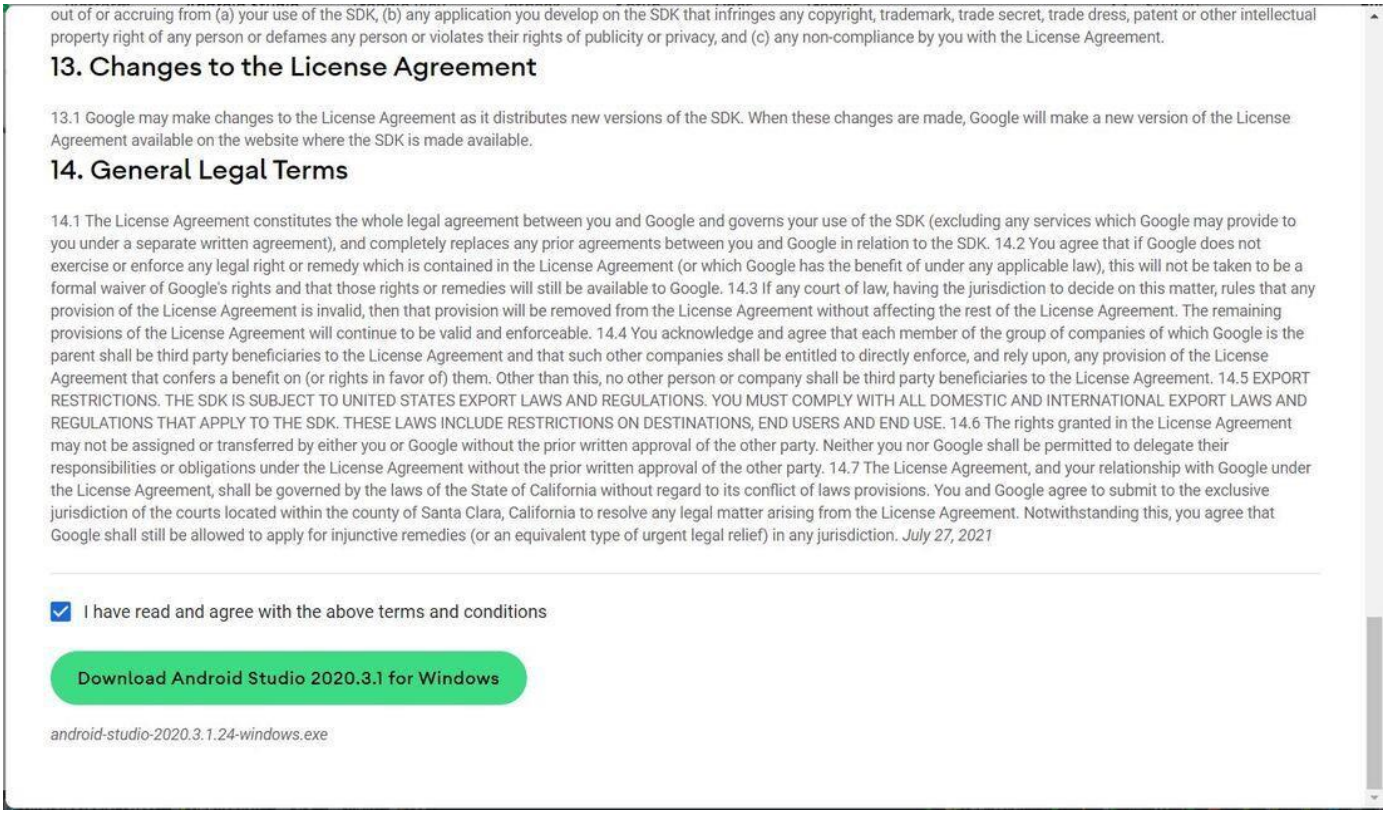
According to the flutter doctor check, we see that flutter was installed successfully in our system, but the Android tools are missing, and so is Android Studio. We also see that there are no connected devices too. Eventually, the next step is about setting up Android tools on your device, to execute the flutter apps built by you.

Step 6: Setting up Android tools and emulator for android devices.

The first step is to download and install Android Studio. To do this, navigate to the official page of Android Studio and click on 'Download Android Studio'.



After accepting the license agreements, you are good to go! Click on the final Download button to start downloading.


out of or accruing from (a) your use of the SDK, (b) any application you develop on the SDK that infringes any copyright, trademark, trade secret, trade dress, patent or other intellectual property right of any person or defames any person or violates their rights of publicity or privacy, and (c) any non-compliance by you with the License Agreement.

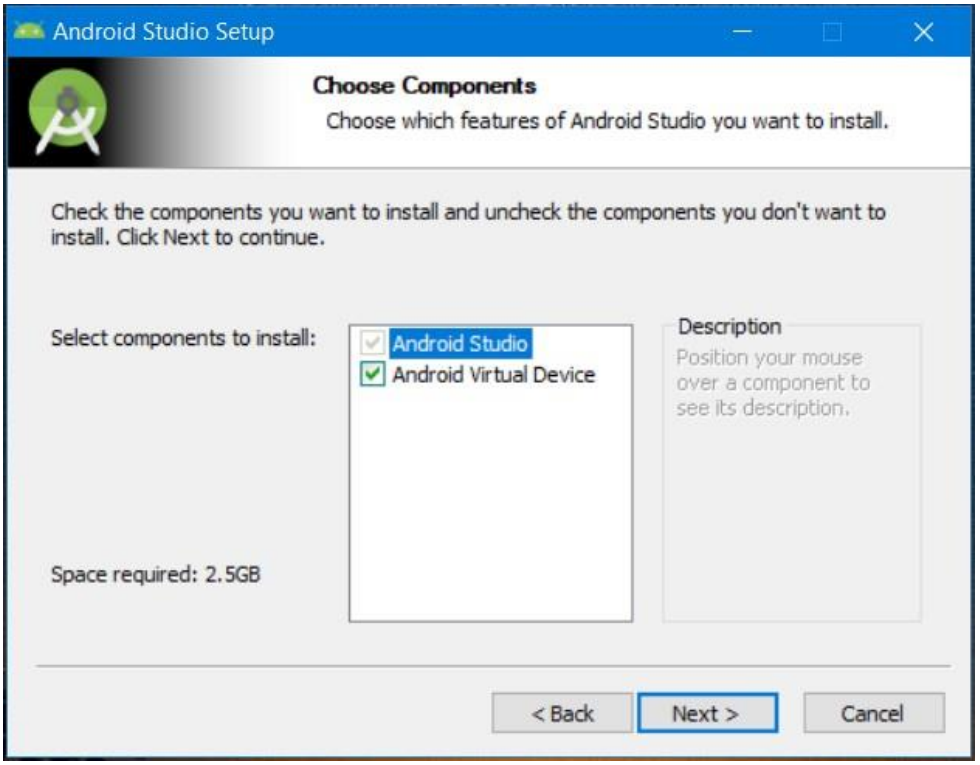
13. Changes to the License Agreement

13.1 Google may make changes to the License Agreement as it distributes new versions of the SDK. When these changes are made, Google will make a new version of the License Agreement available on the website where the SDK is made available.

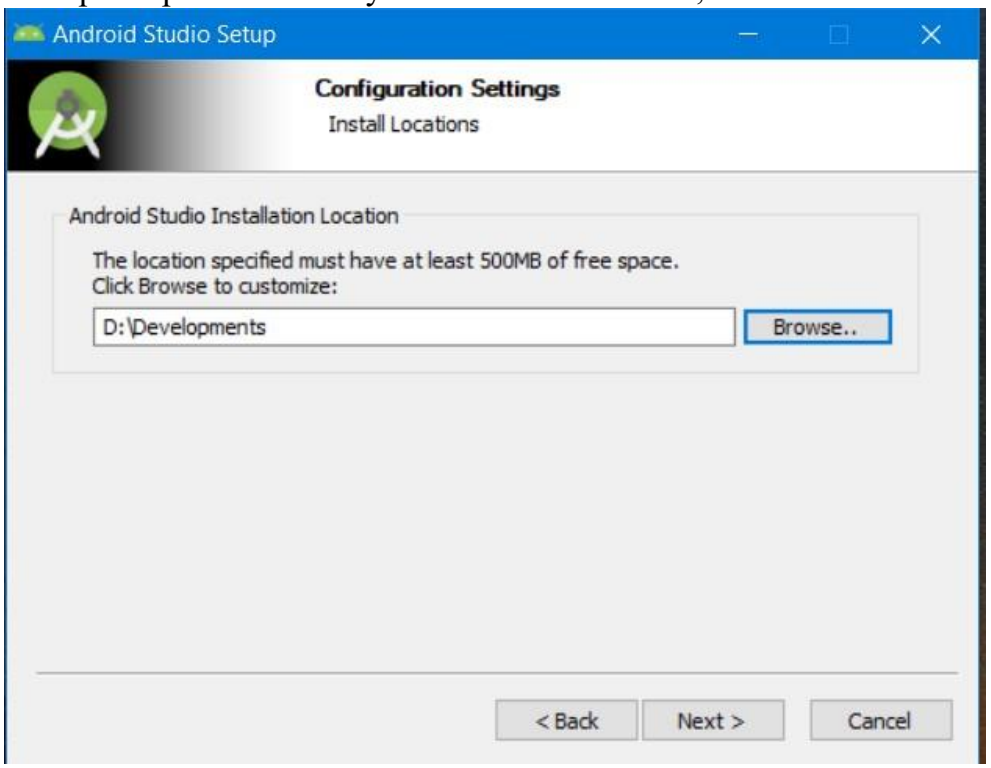
14. General Legal Terms

14.1 The License Agreement constitutes the whole legal agreement between you and Google and governs your use of the SDK (excluding any services which Google may provide to you under a separate written agreement), and completely replaces any prior agreements between you and Google in relation to the SDK. 14.2 You agree that if Google does not exercise or enforce any legal right or remedy which is contained in the License Agreement (or which Google has the benefit of under any applicable law), this will not be taken to be a formal waiver of Google's rights and that those rights or remedies will still be available to Google. 14.3 If any court of law, having the jurisdiction to decide on this matter, rules that any provision of the License Agreement is invalid, then that provision will be removed from the License Agreement without affecting the rest of the License Agreement. The remaining provisions of the License Agreement will continue to be valid and enforceable. 14.4 You acknowledge and agree that each member of the group of companies of which Google is the parent shall be third party beneficiaries to the License Agreement and that such other companies shall be entitled to directly enforce, and rely upon, any provision of the License Agreement that confers a benefit on (or rights in favor of) them. Other than this, no other person or company shall be third party beneficiaries to the License Agreement. 14.5 EXPORT RESTRICTIONS. THE SDK IS SUBJECT TO UNITED STATES EXPORT LAWS AND REGULATIONS. YOU MUST COMPLY WITH ALL DOMESTIC AND INTERNATIONAL EXPORT LAWS AND REGULATIONS THAT APPLY TO THE SDK. THESE LAWS INCLUDE RESTRICTIONS ON DESTINATIONS, END USERS AND END USE. 14.6 The rights granted in the License Agreement may not be assigned or transferred by either you or Google without the prior written approval of the other party. Neither you nor Google shall be permitted to delegate their responsibilities or obligations under the License Agreement without the prior written approval of the other party. 14.7 The License Agreement, and your relationship with Google under the License Agreement, shall be governed by the laws of the State of California without regard to its conflict of laws provisions. You and Google agree to submit to the exclusive jurisdiction of the courts located within the county of Santa Clara, California to resolve any legal matter arising from the License Agreement. Notwithstanding this, you agree that Google shall still be allowed to apply for injunctive remedies (or an equivalent type of urgent legal relief) in any jurisdiction. *July 27, 2021*

After the download is complete, let's move on to the next step, i.e. installation.
Under 'Components', make sure that both Android Studio and Android Virtual Device are checked, and only then proceed. The Android Virtual Device is an essential tool for running various types and sizes of android emulators to test your flutter project. Henceforth, click on 'Next'.

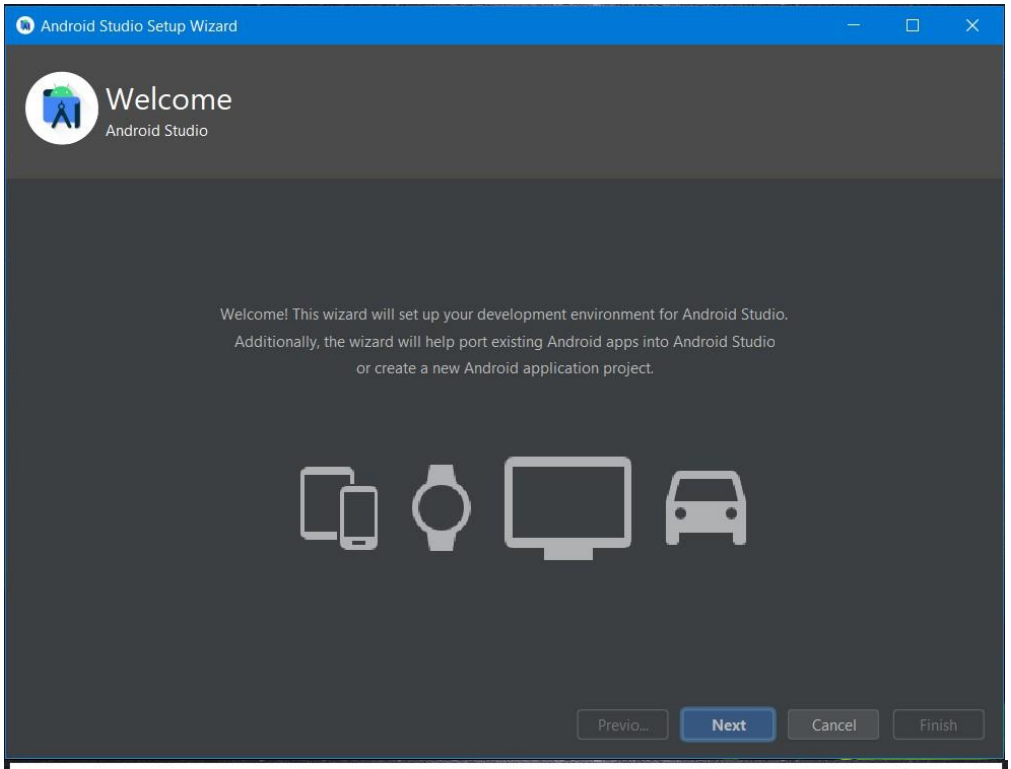


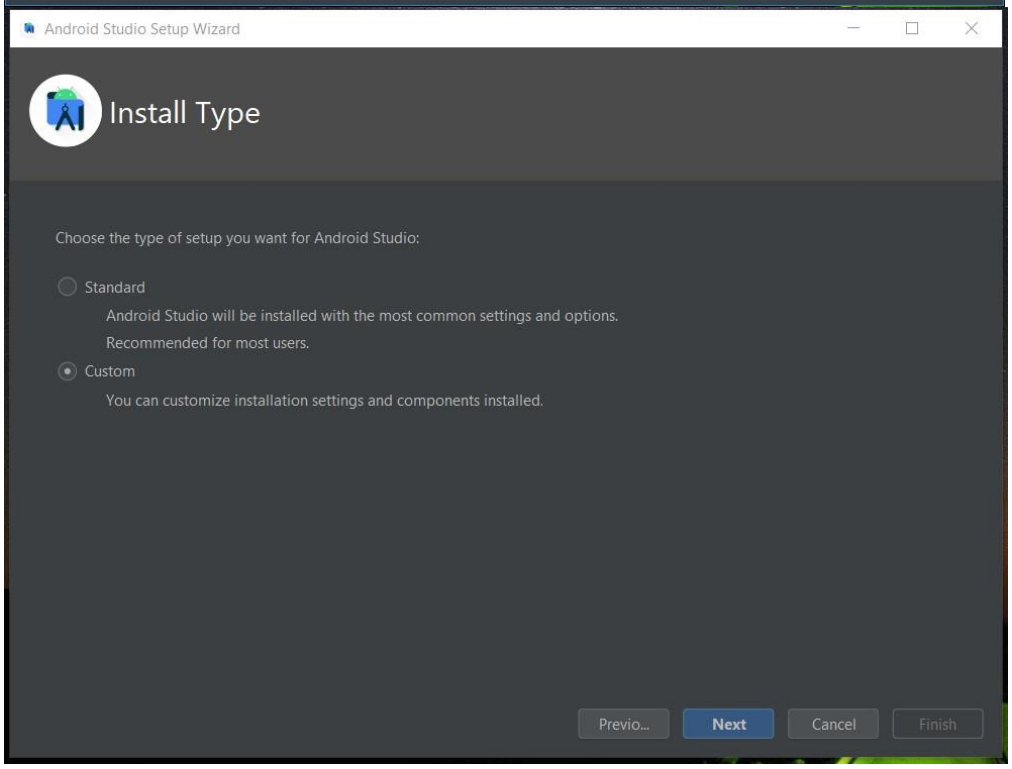
Select the directory you would want your file to be installed in. It is recommended to select some other path apart from the system drive. Once done, click on 'Next'.



Finally, click on 'Install'. Wait for a couple of seconds for the installation to complete. Check the box beside 'Launch Android Studio'. Click on 'Finish'.

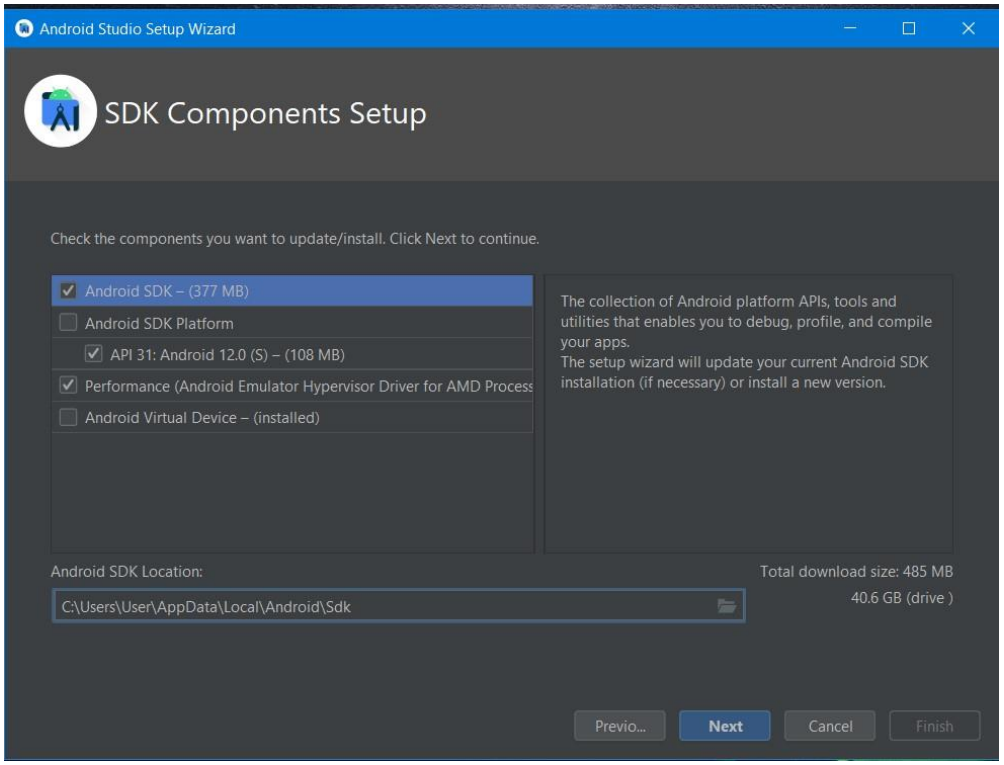
Wait for Android Studio to launch on your computer. On the home screen, click Next > Custom > Next.



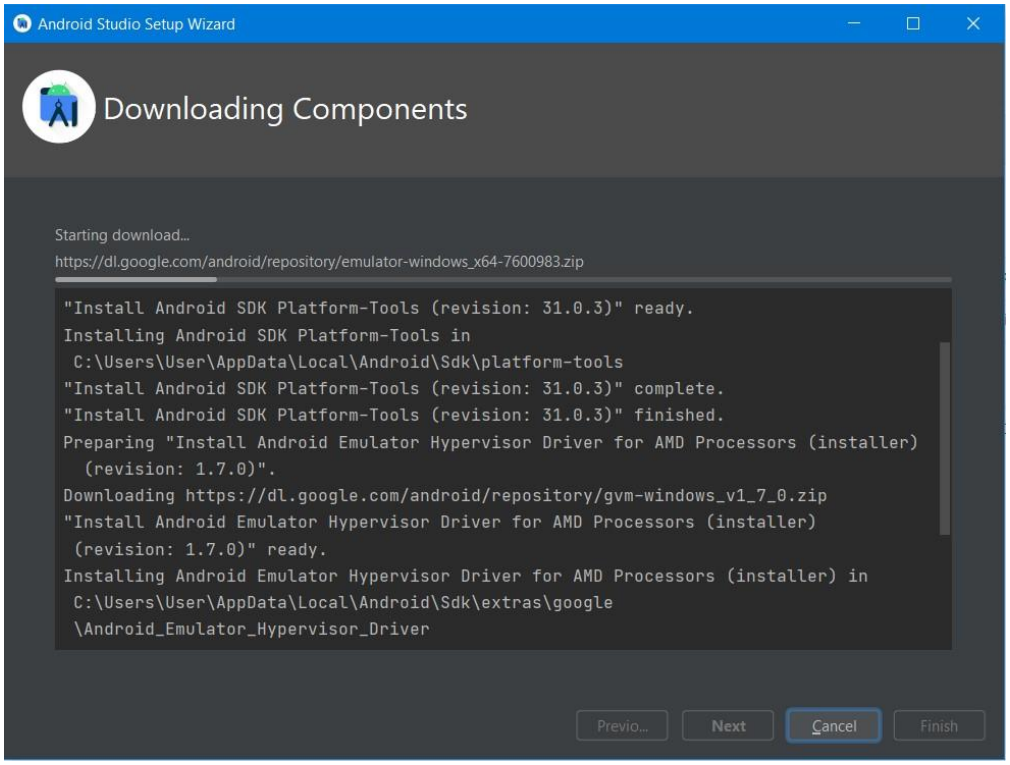


For the Java Development kit location in the next step, it is recommended to keep the default path it requires, to avoid the hassle. In the next step, choose the UI appearance you'd like for Android Studio. Click 'Next'.

This next step is a bit important. Remember to check the required boxes exactly as shown below. If kits have already been installed, you can ignore those and move on. Click 'Next'. Set your desired folder for Android SDK.

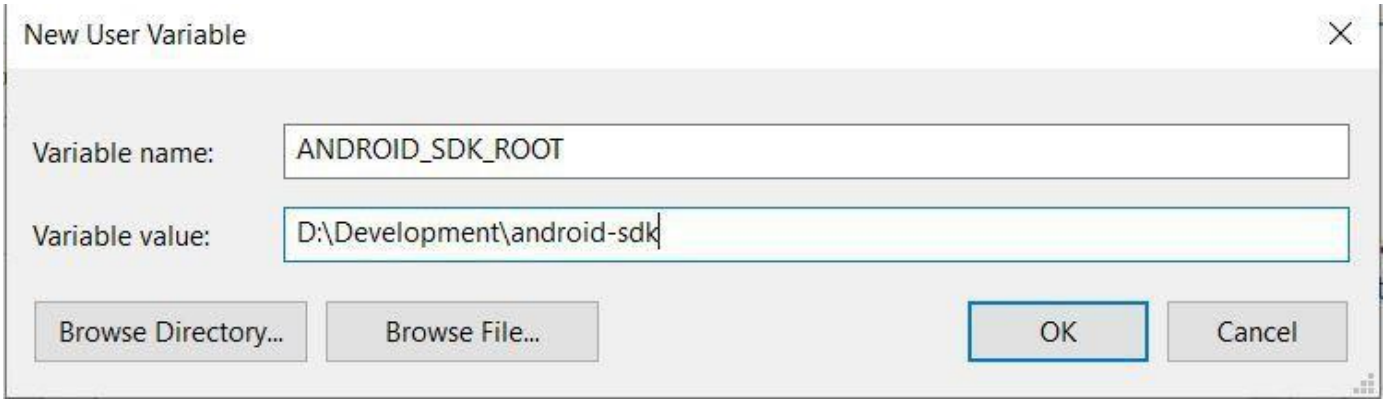


With that done, click on 'Finish'. Android Studio will now install all the necessary android tools required for the execution of your flutter projects. This may take a significant time – it's better to wait!



Now, we are ready to create and build flutter projects on Android Studio and run it on a real or a virtual Android device (emulator).

Step 6: Set SDK as an environment variable, for global access.
Now, open Command Prompt terminal and run 'flutter doctor' again. If you have installed Android SDK in the default directory suggested by Android Studio, there wouldn't be any problem that would appear. Nevertheless, if you have installed it in a non-default directory, flutter would not be able to detect it in your system. To help it able to do that, you guessed it…we would be assigning it as an environment variable, giving global access.
As discussed earlier in Step 4, go to environment variables and click 'New', and do the following (as recommended by flutter doctor). Click 'OK'.



Step 7: Accept required Android Licenses. On the Command Prompt terminal, type in: flutter doctor --android-licenses as suggested by flutter doctor. Hit Enter. To review licenses, type 'y' for Yes.

You'll see a couple of repeated prompts that look like this: Accept? (y/N):

Type 'y' whenever asked for.

Finally, after all the license agreements have been accepted, you should see a message that looks something like this:

All SDK package licenses accepted Step 7: Setup Android Emulator.

You have the option to choose between an Android Device or an Android Emulator to build your application on. It depends totally on you.

For setting up Android Device, go through the official docs page and follow the exact steps as mentioned. Download The Google USB Driver by following the link and install according to the instructions given. This can also be installed through Android Studio, which you can later connect to a real Android Device to build the application.

Set up your Android device

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) or higher.

1. Enable **Developer options** and **USB debugging** on your device. Detailed instructions are available in the Android documentation.
2. Windows-only: Install the Google USB Driver.
3. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.
4. In the terminal, run the `flutter devices` command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your `adb` tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the `ANDROID_SDK_ROOT` environment variable to that installation directory.

For setting up Android Emulator, you need to go through the following steps:

Open Android Studio.

On the topmost menu bar, click on Tools > SDK Manager.

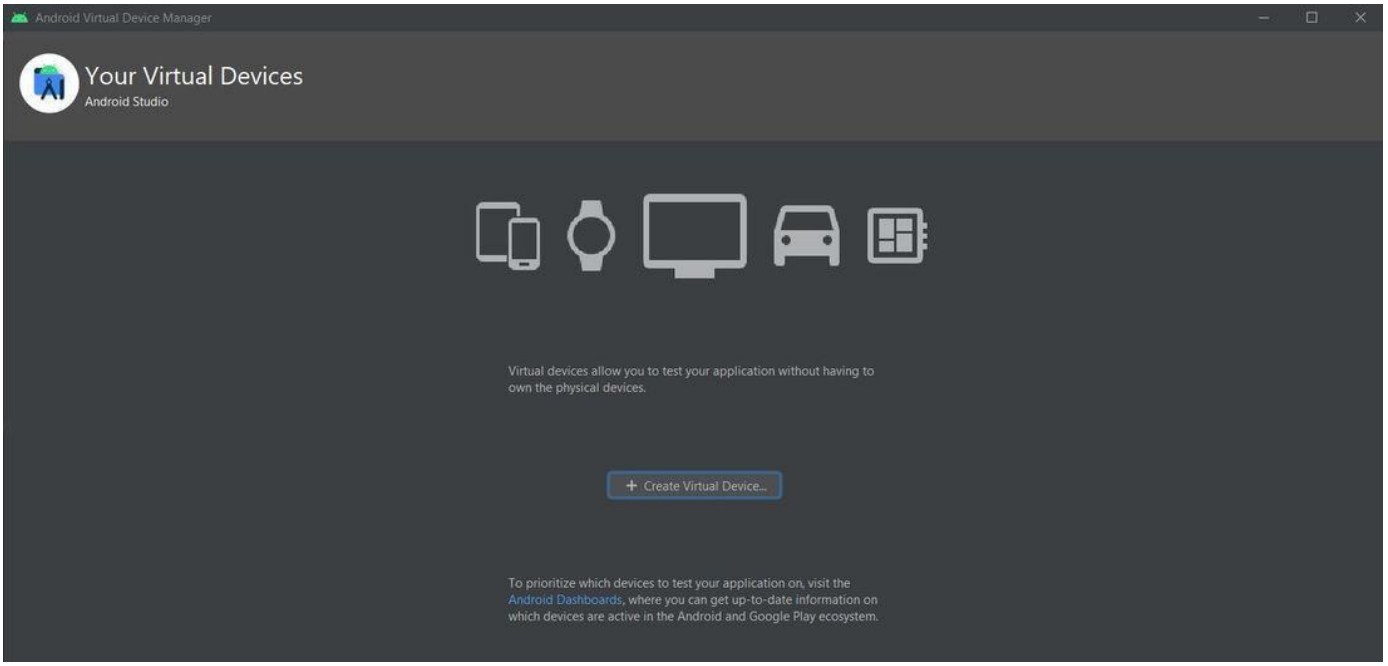
Verify whether you have the latest SDK installed. Remember to install the latest stable version too by checking on the box to the left.

In my case, it is 'Android 9.0 (Pie)'. You can even uncheck the latest version (if not stable), to not only save space but also run all your applications on the stable version itself.

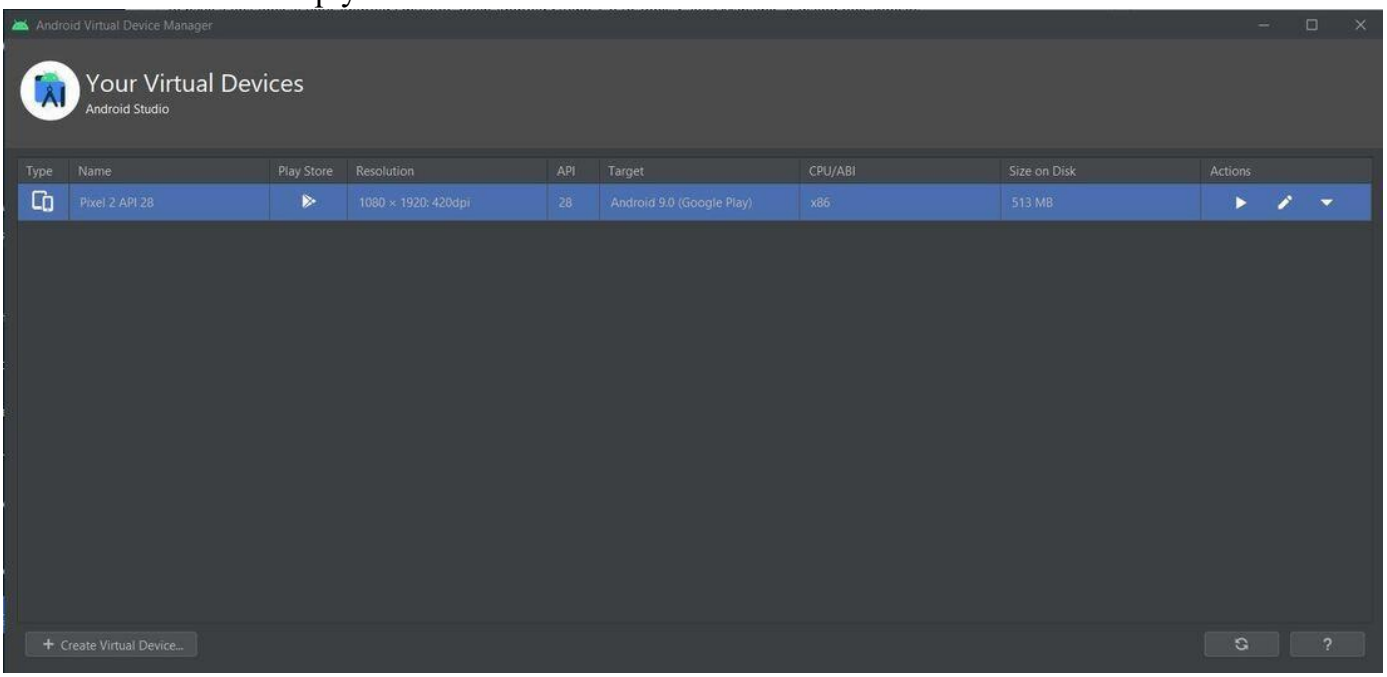
Under the 'SDK Tools' tab, don't forget to check Google USB Driver to later connect a real Android Device. With that, click 'Apply'. Click 'OK' to start SDK installation.

This might take a couple of minutes to complete. After the setup is done, click on 'Finish'. Your setup is now complete!

To have a first look at your Android Emulator, open Android Studio. Go to Tools > AVD Manager. A dialog box appears.



Click on 'Create Virtual Device…', select a device and its dimensions according to your preference, select a system image and lastly, under all default settings, click on 'Finish'. Click on the ' ' button to fire up your emulator.



Conclusion:

We understand how to install and configure Flutter Environment.

EXPERIMENT NO. 2

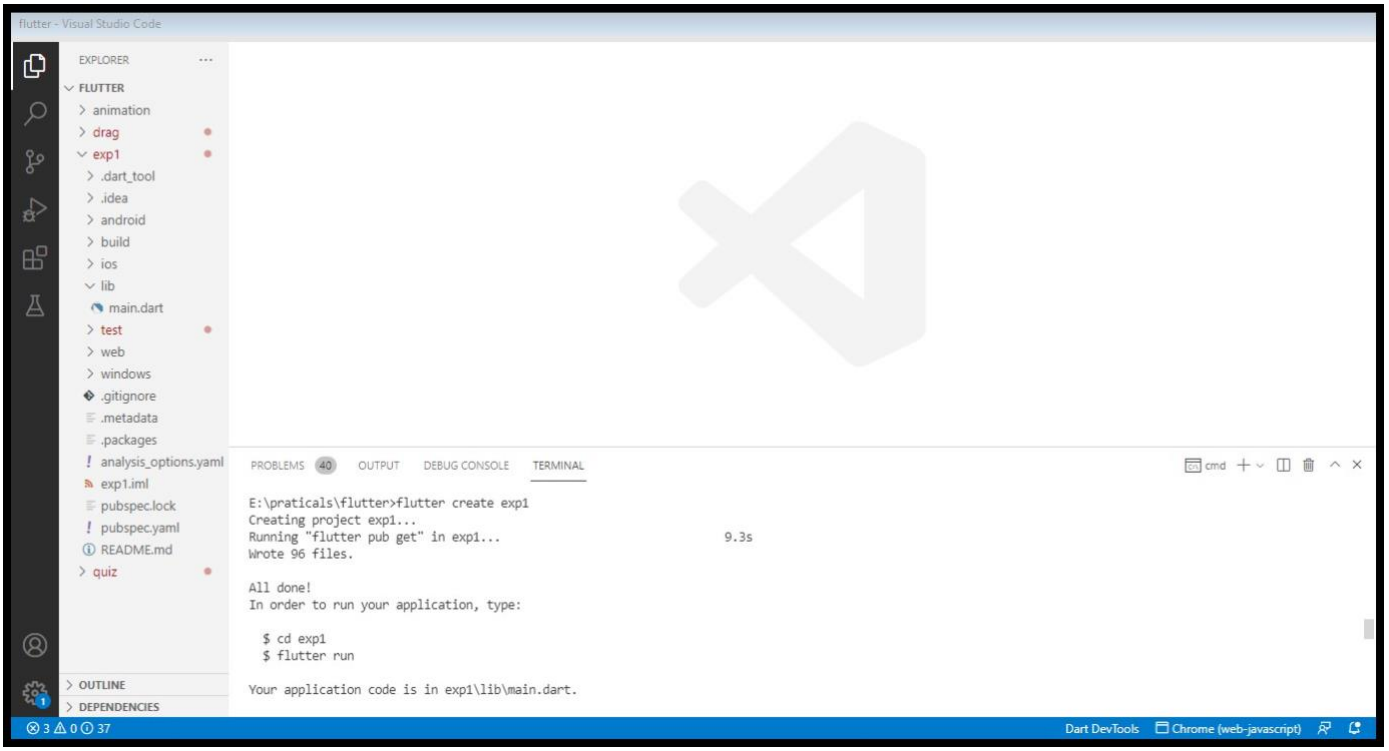
Aim: To design a layout of Flutter App using layout widgets.

Theory:

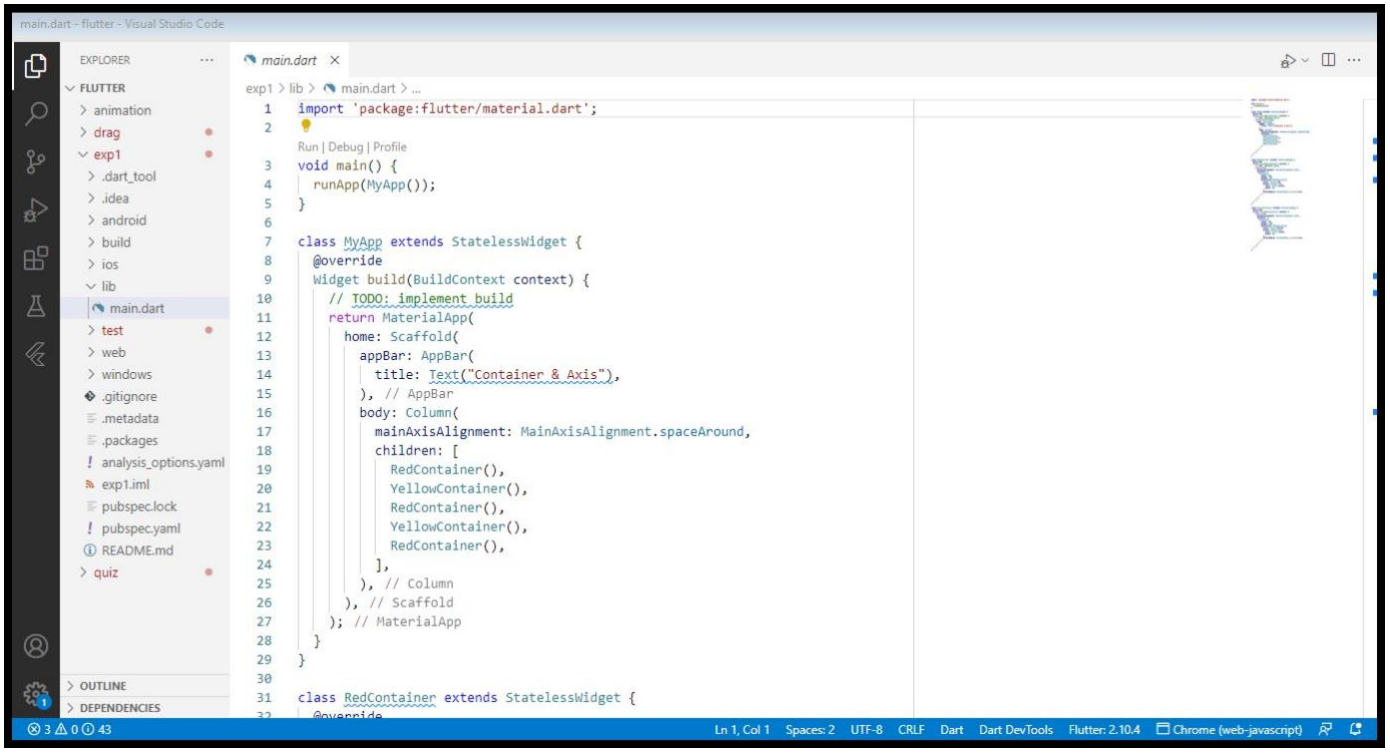
Flutter is basically Google's portable user interface (UI) toolkit, used to build and develop eye-catching, natively-built applications for mobile, desktop, and web, from a single codebase. Flutter is free, open-sourced, and compatible with existing code. It is utilized by companies and developers around the world, due to its user-friendly interface and fairly simple, yet to-the-point commands.

Steps:

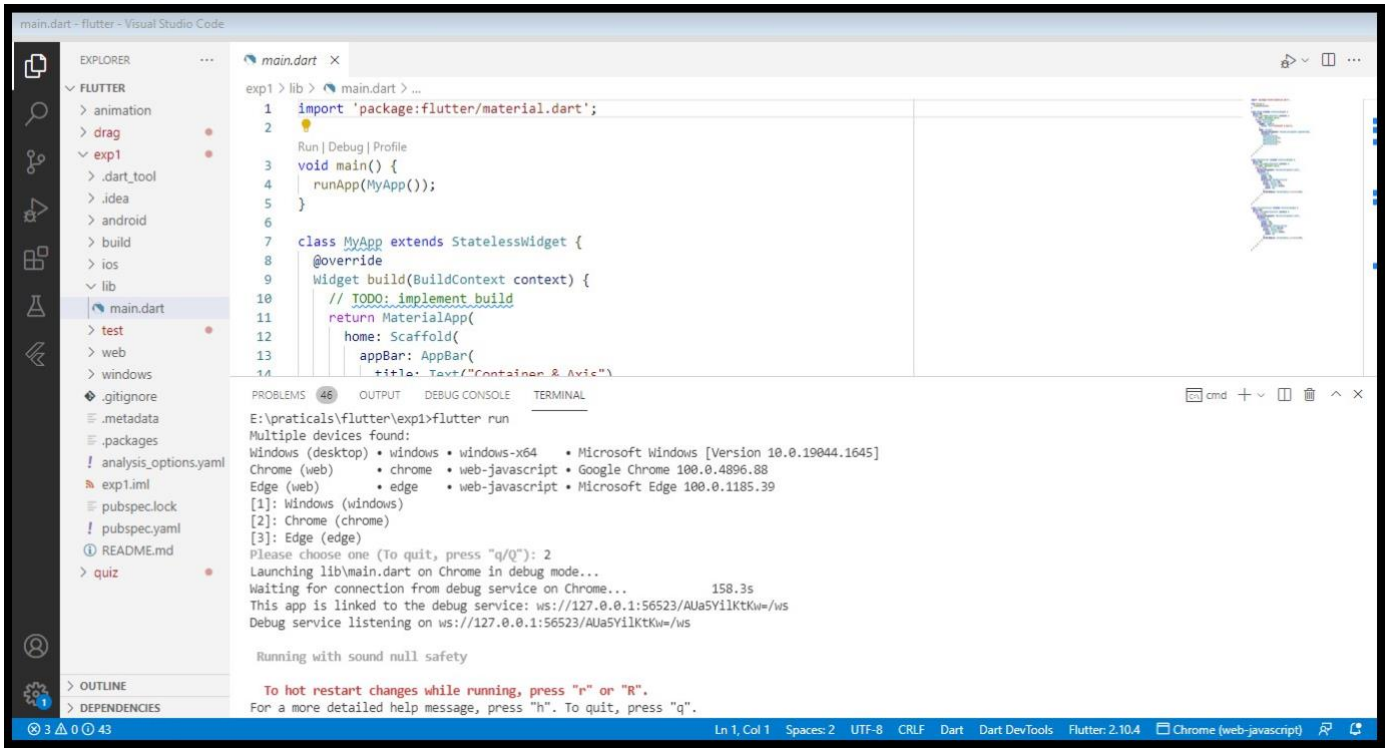
Step1: create app using Flutter create command



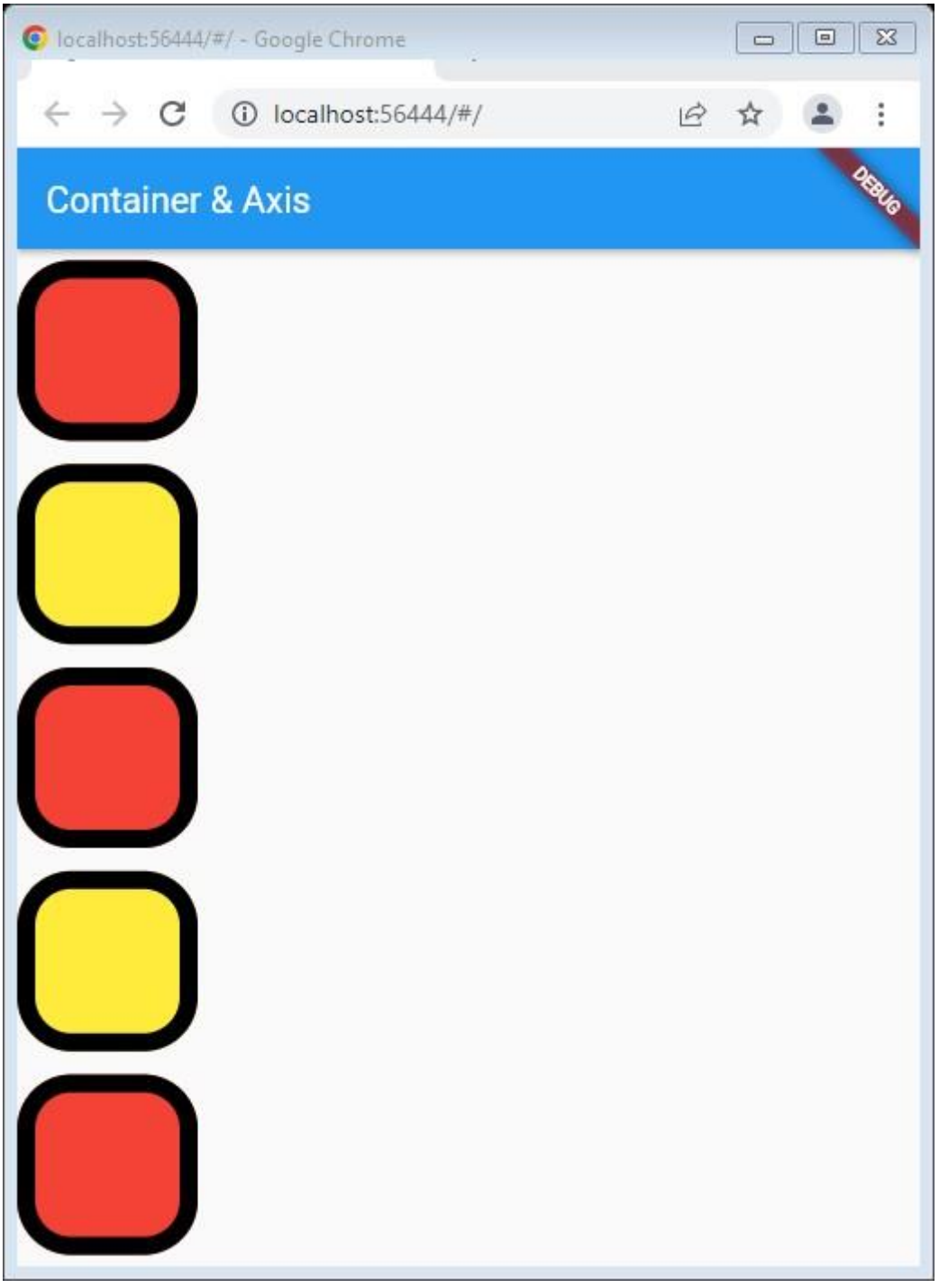
Step2: Write code in main.dart file



Step3: Run app Using Flutter Run Command



Step4: Output



Conclusion:

We understand how to design a layout of Flutter App using layout widgets.

EXPERIMENT NO. 3

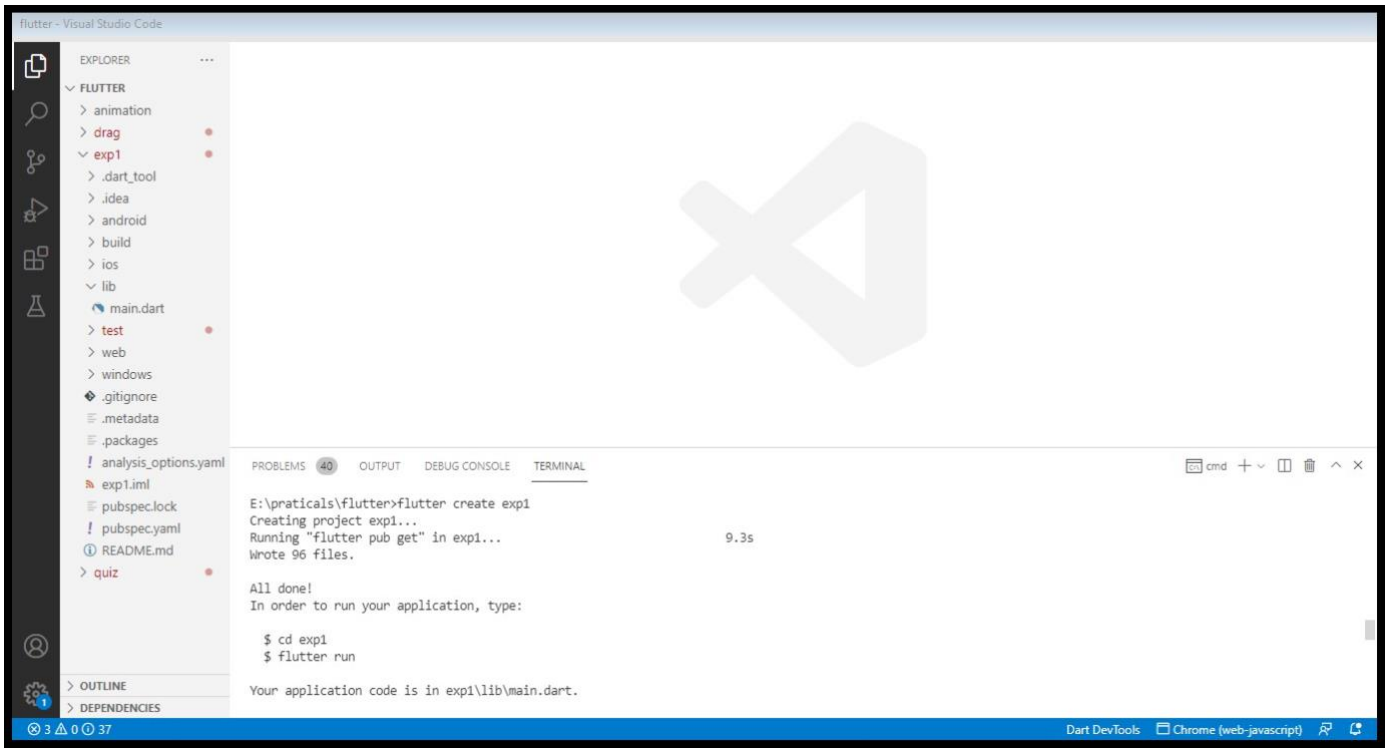
Aim: To apply animation to image in Flutter app

Theory:

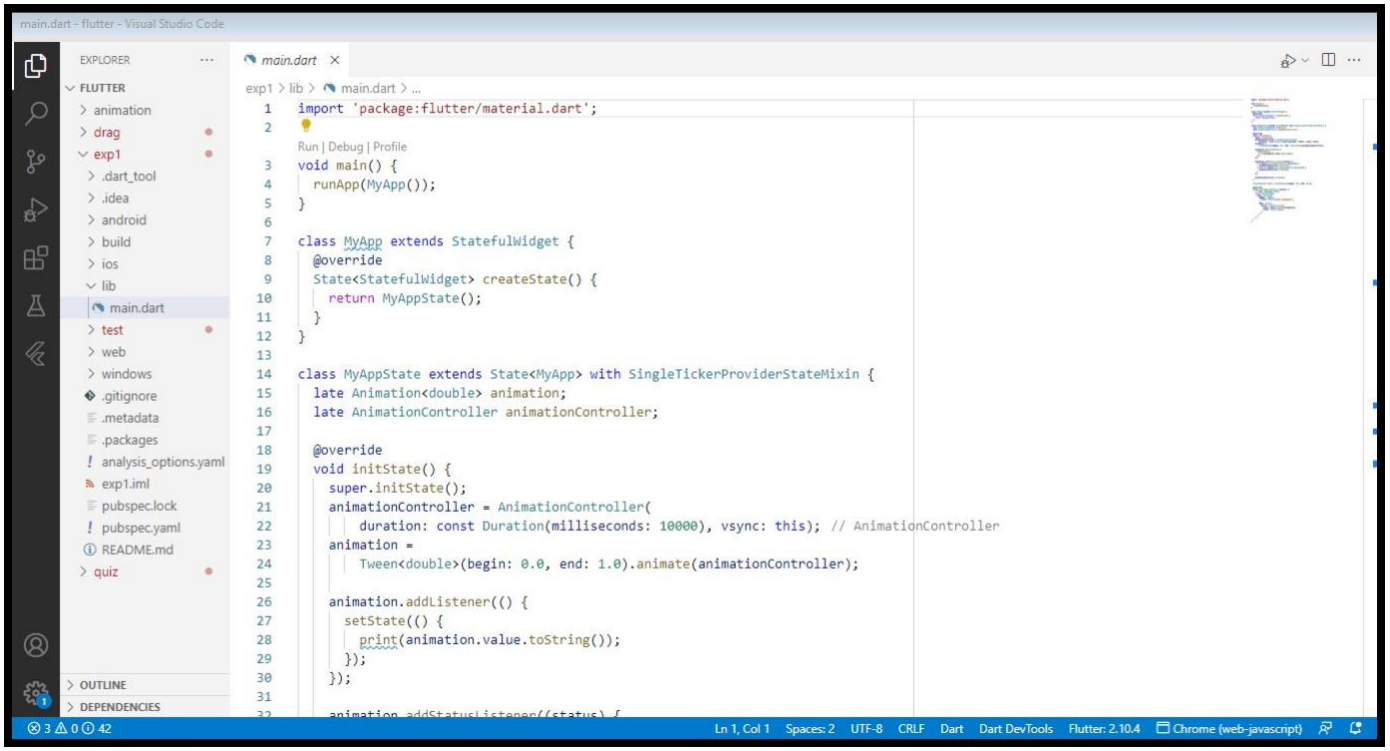
Flutter is basically Google's portable user interface (UI) toolkit, used to build and develop eye-catching, natively-built applications for mobile, desktop, and web, from a single codebase. Flutter is free, open-sourced, and compatible with existing code. It is utilized by companies and developers around the world, due to its user-friendly interface and fairly simple, yet to-the-point commands.

Steps:

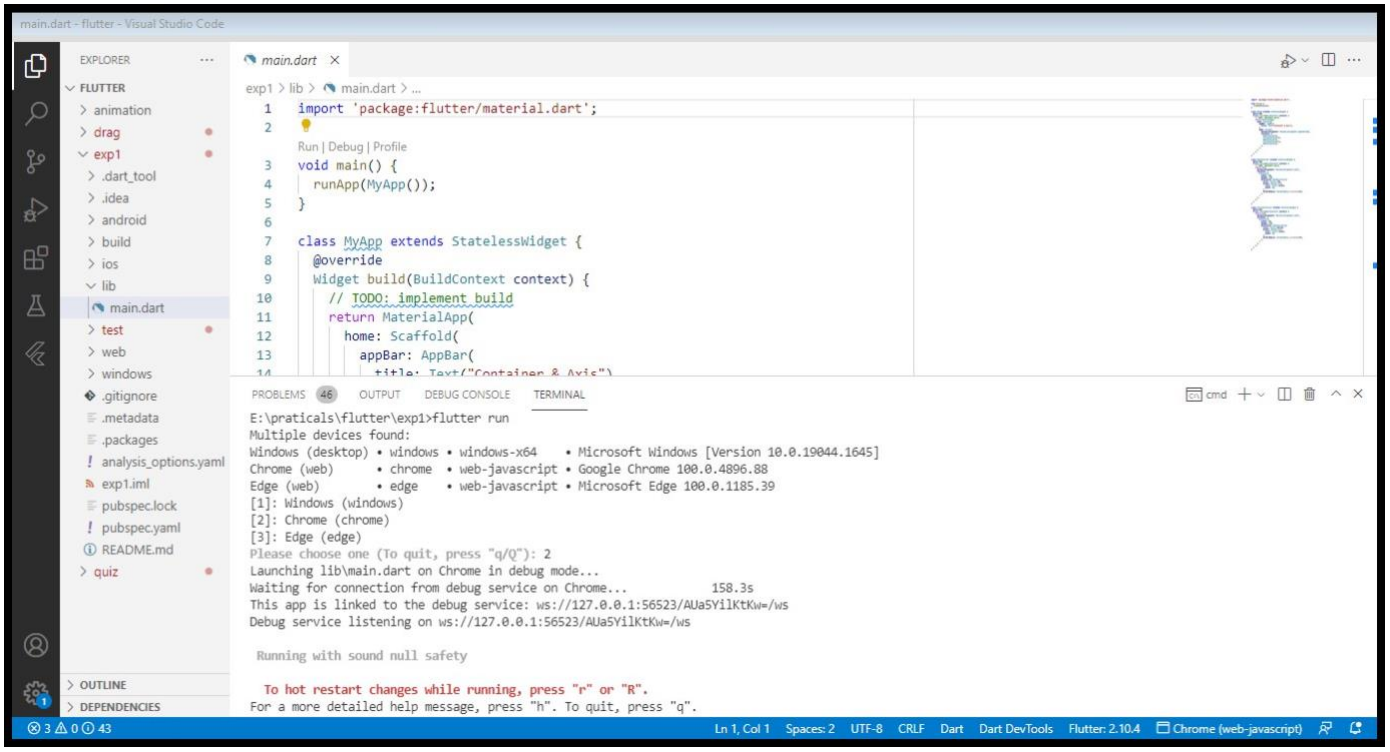
Step1: create app using Flutter create command



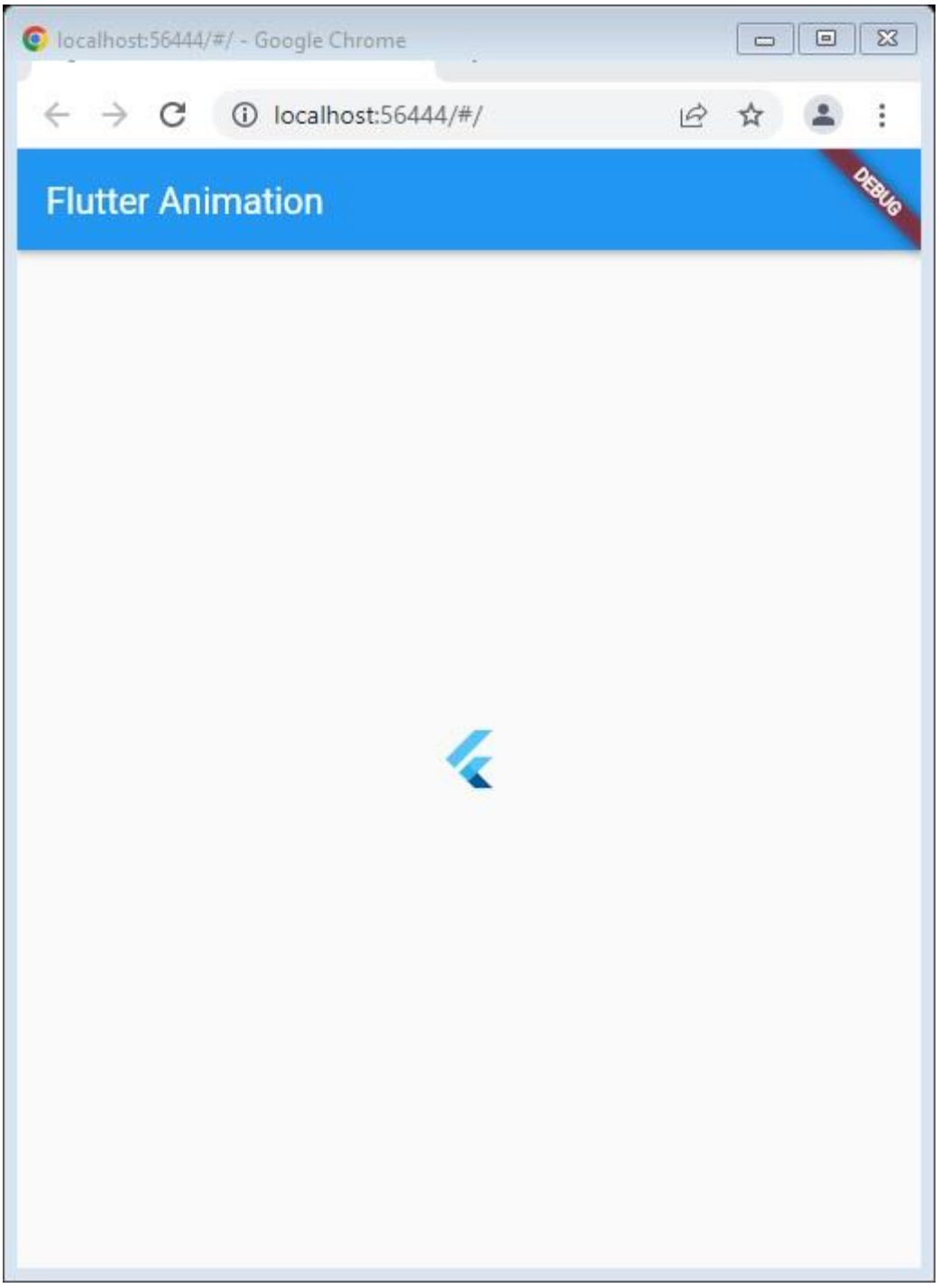
Step2: Write code in main.dart file

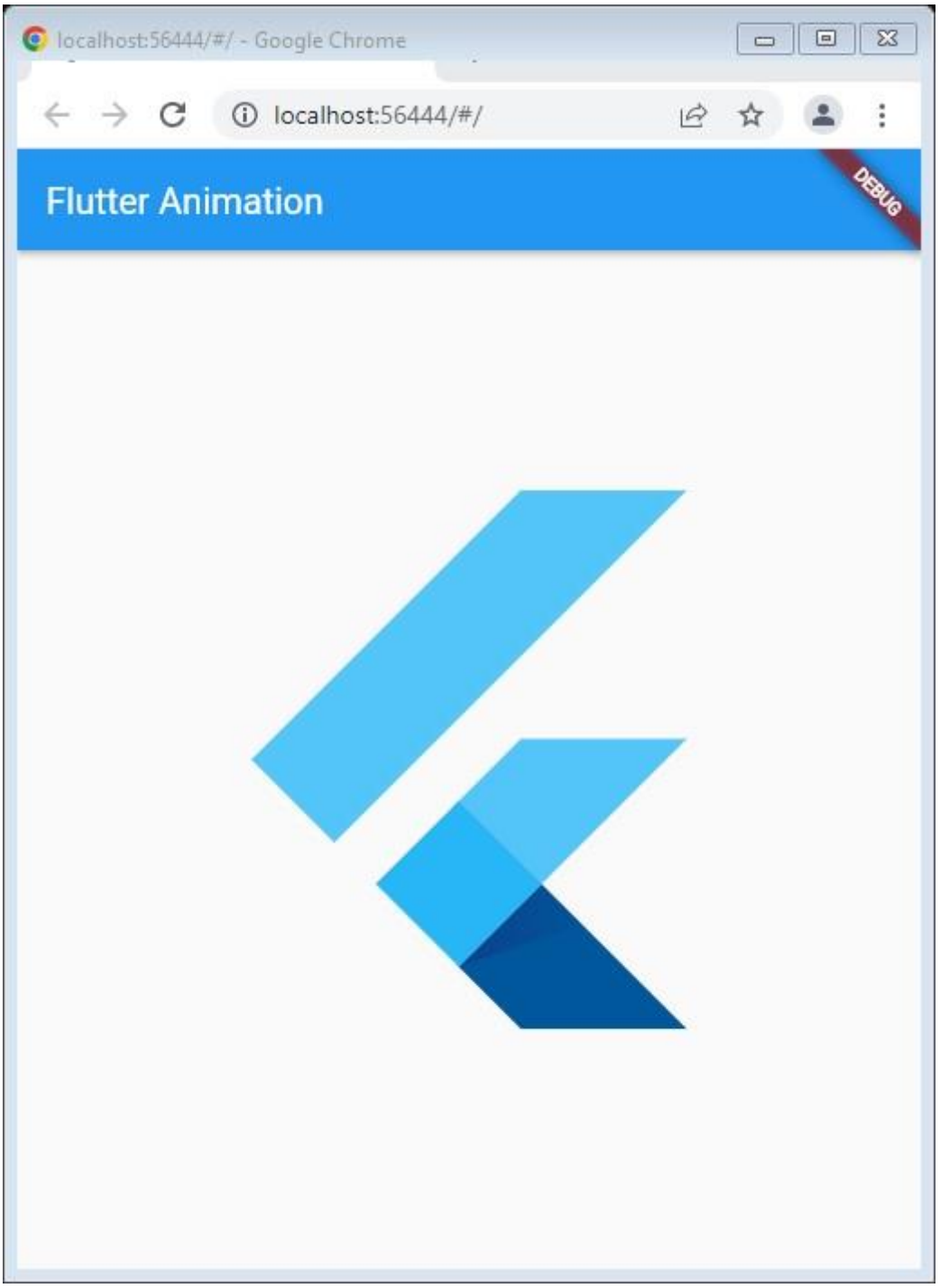


Step3: Run app Using Flutter Run Command



Step4: Output





Conclusion:

We understand how to apply animation to image in Flutter app.

EXPERIMENT NO. 4

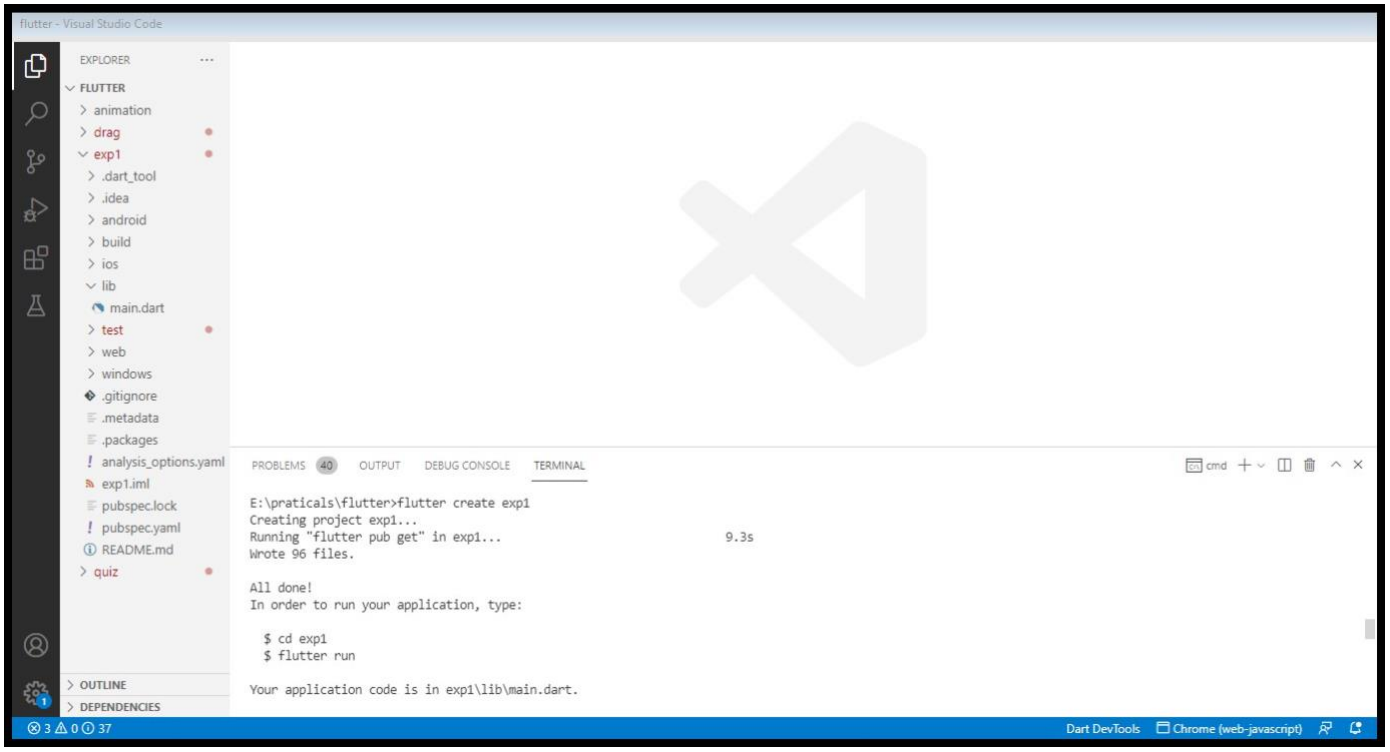
Aim: To apply navigation in Flutter App.

Theory:

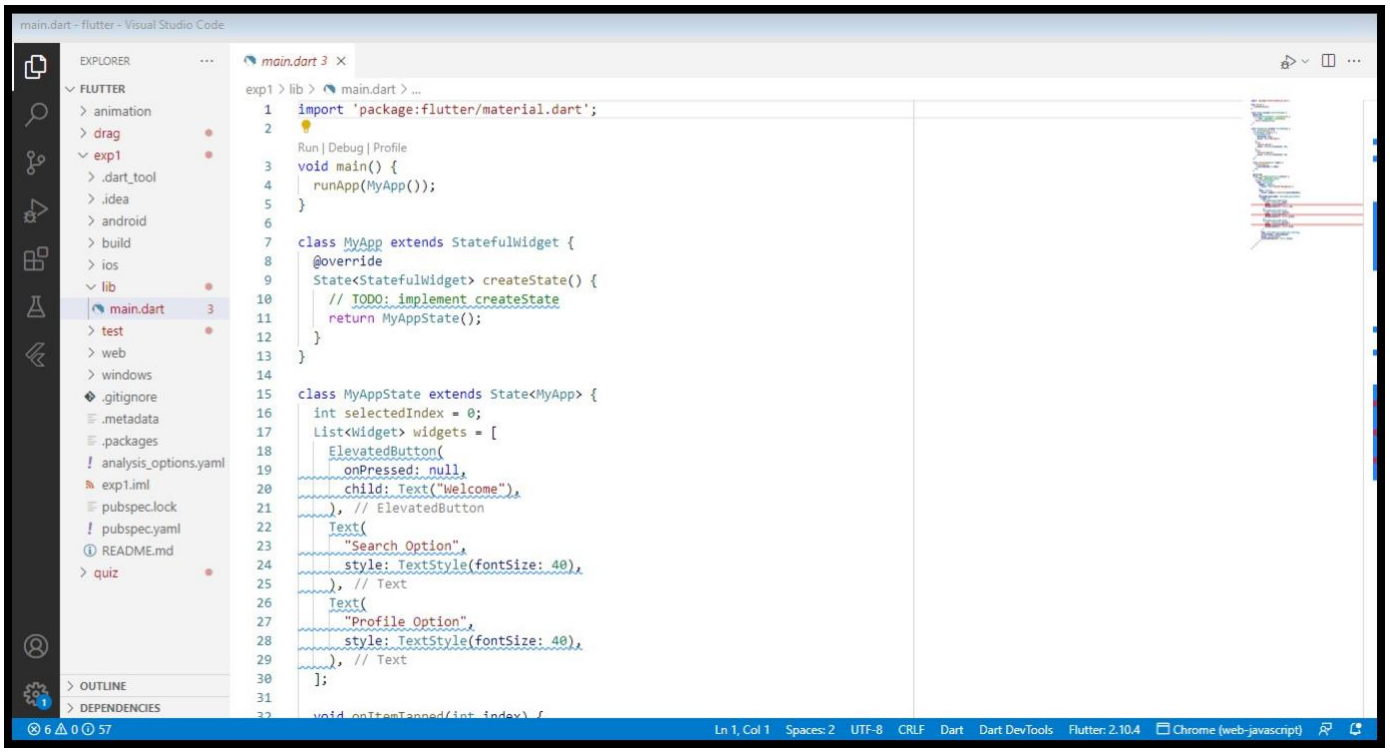
Flutter is basically Google's portable user interface (UI) toolkit, used to build and develop eye-catching, natively-built applications for mobile, desktop, and web, from a single codebase. Flutter is free, open-sourced, and compatible with existing code. It is utilized by companies and developers around the world, due to its user-friendly interface and fairly simple, yet to-the-point commands.

Steps:

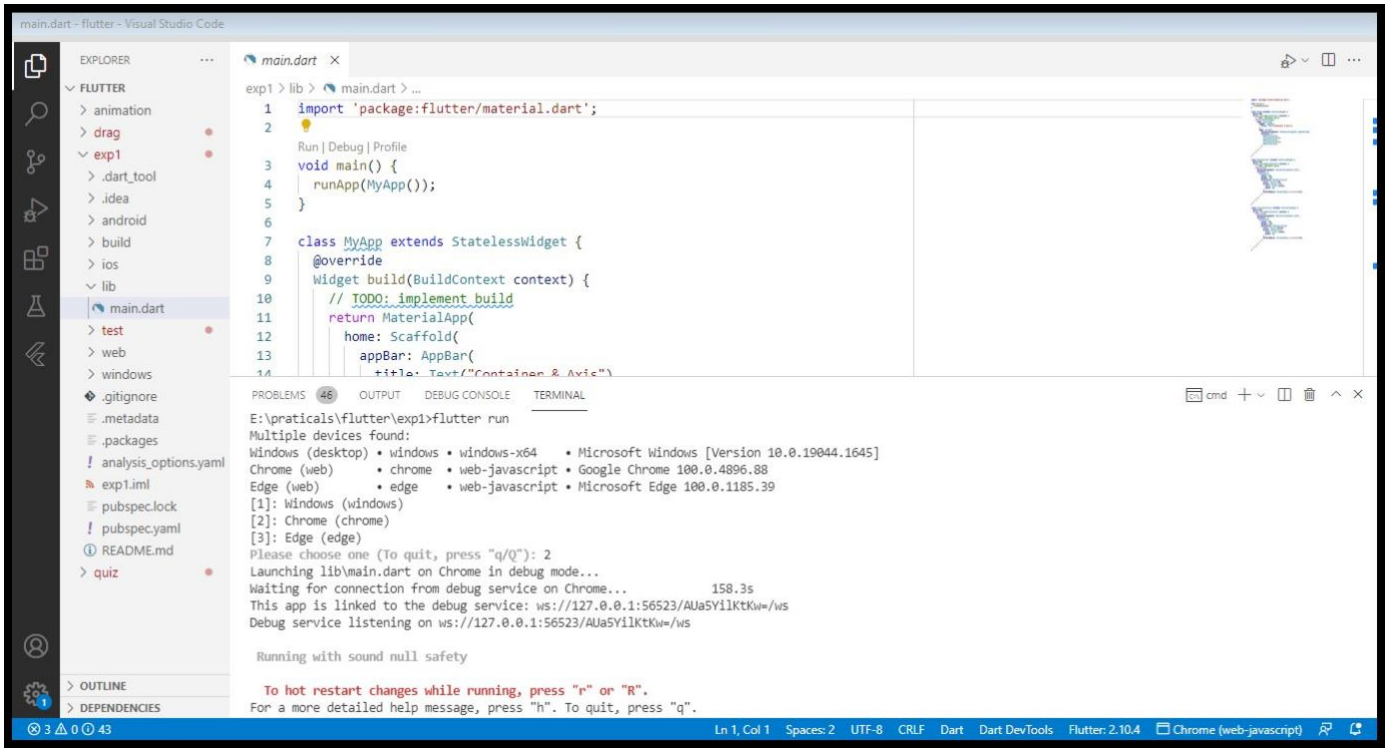
Step1: create app using Flutter create command



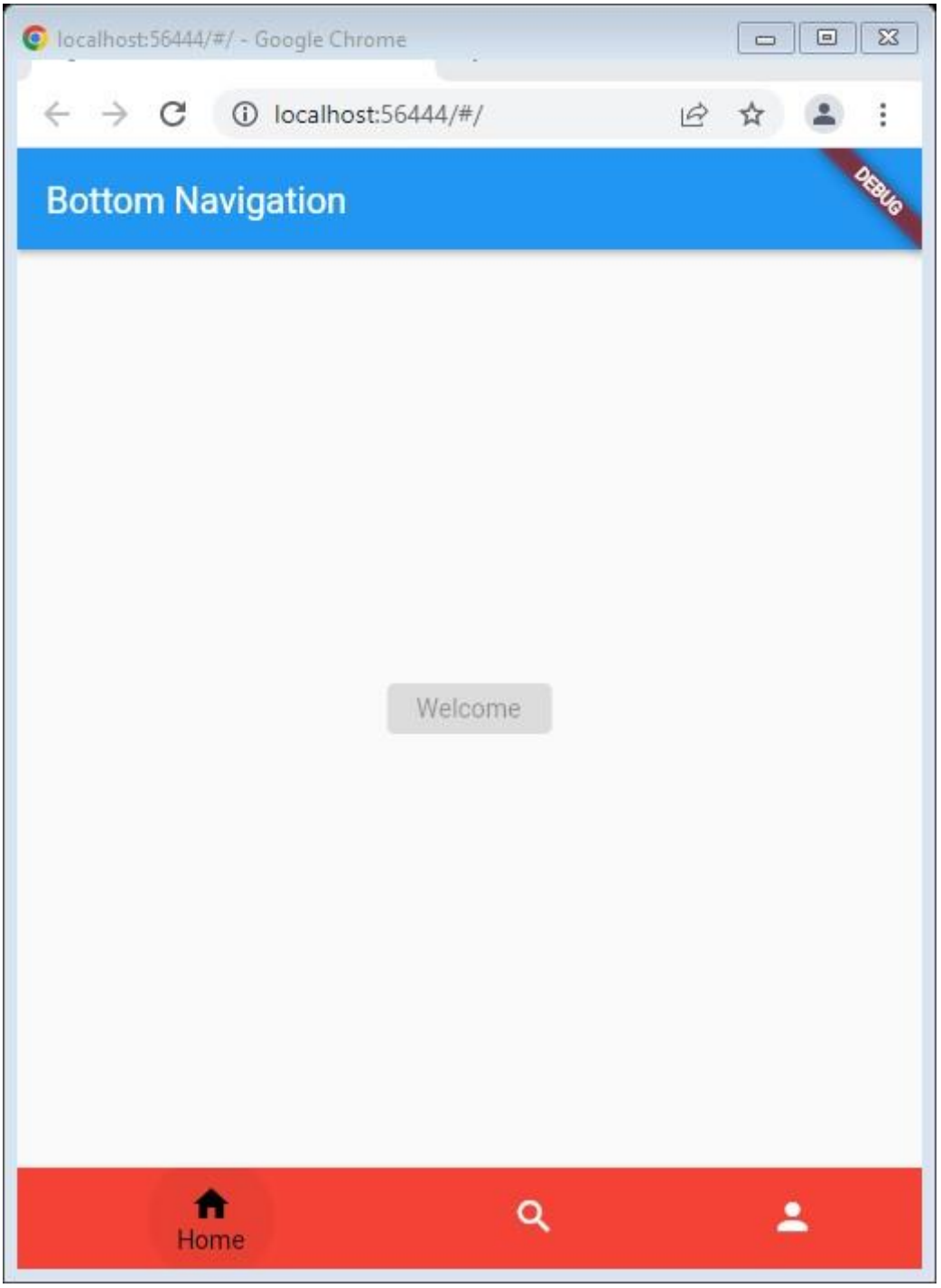
Step2: Write code in main.dart file

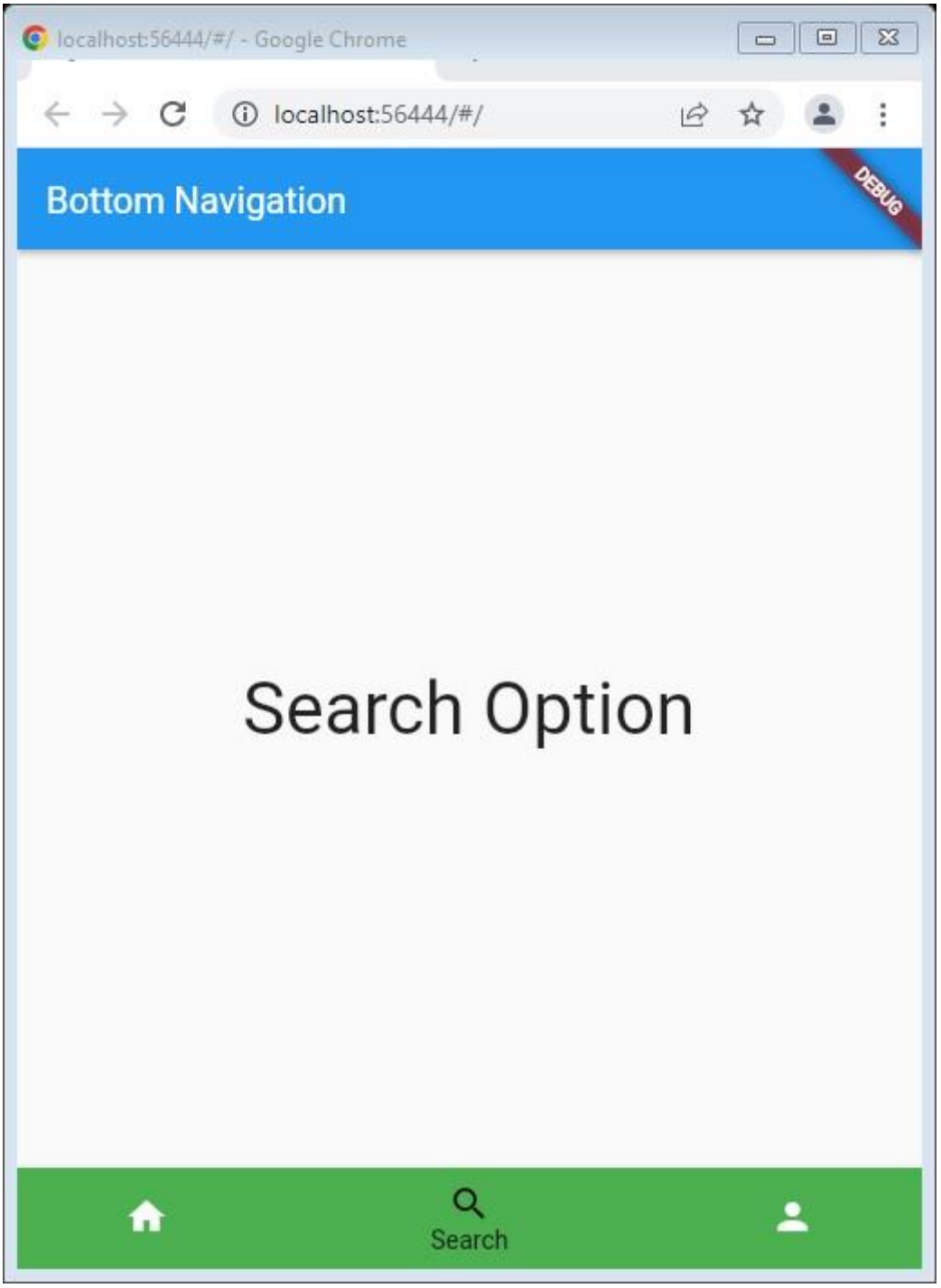


Step3: Run app Using Flutter Run Command



Step4: Output


localhost:56444/#/ - Google Chrome
localhost:56444/#/
Bottom Navigation
DEBUG
Welcome
Home



Conclusion:

We understand how to apply navigation in Flutter App.

EXPERIMENT NO. 5

Aim: To create a responsive User Interface using jQuery Mobile/ Material UI/ Angular UI/ React UI for Ecommerce application.

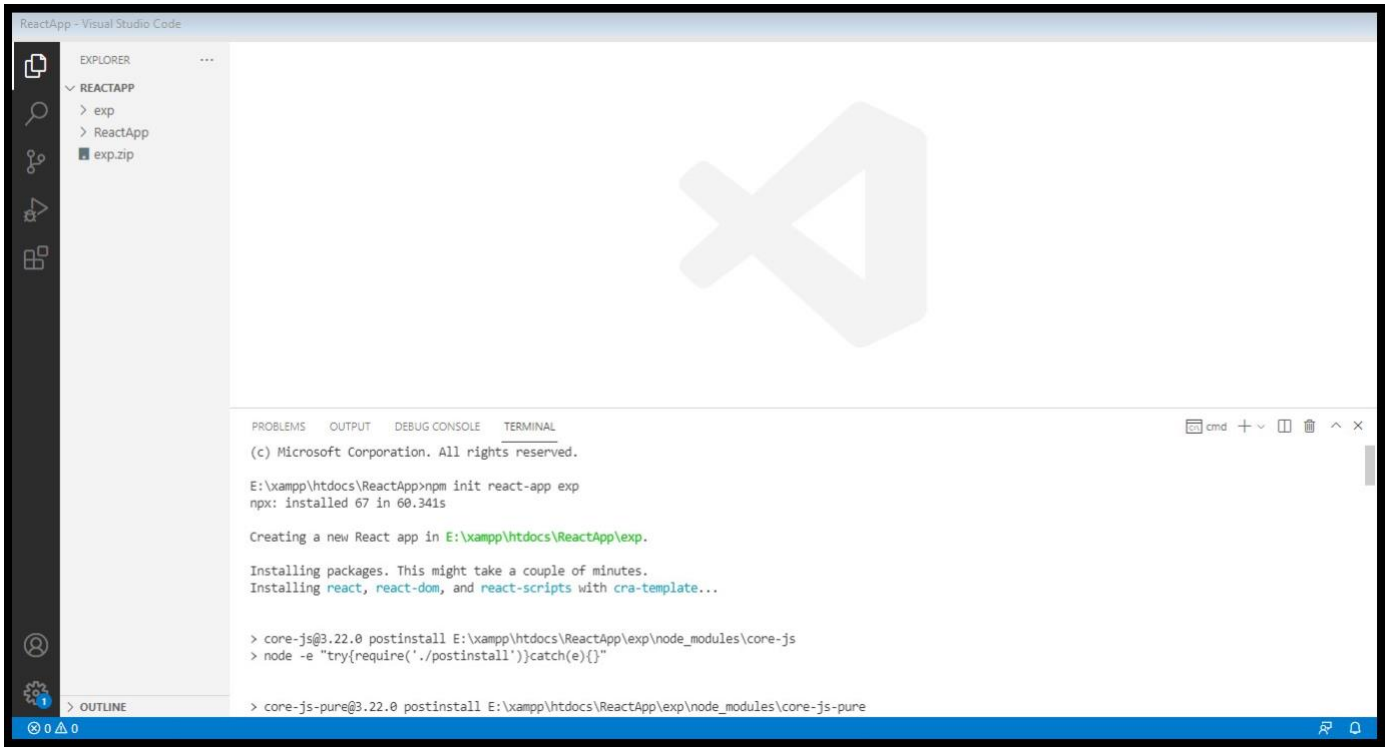
Theory:

Responsive user interface

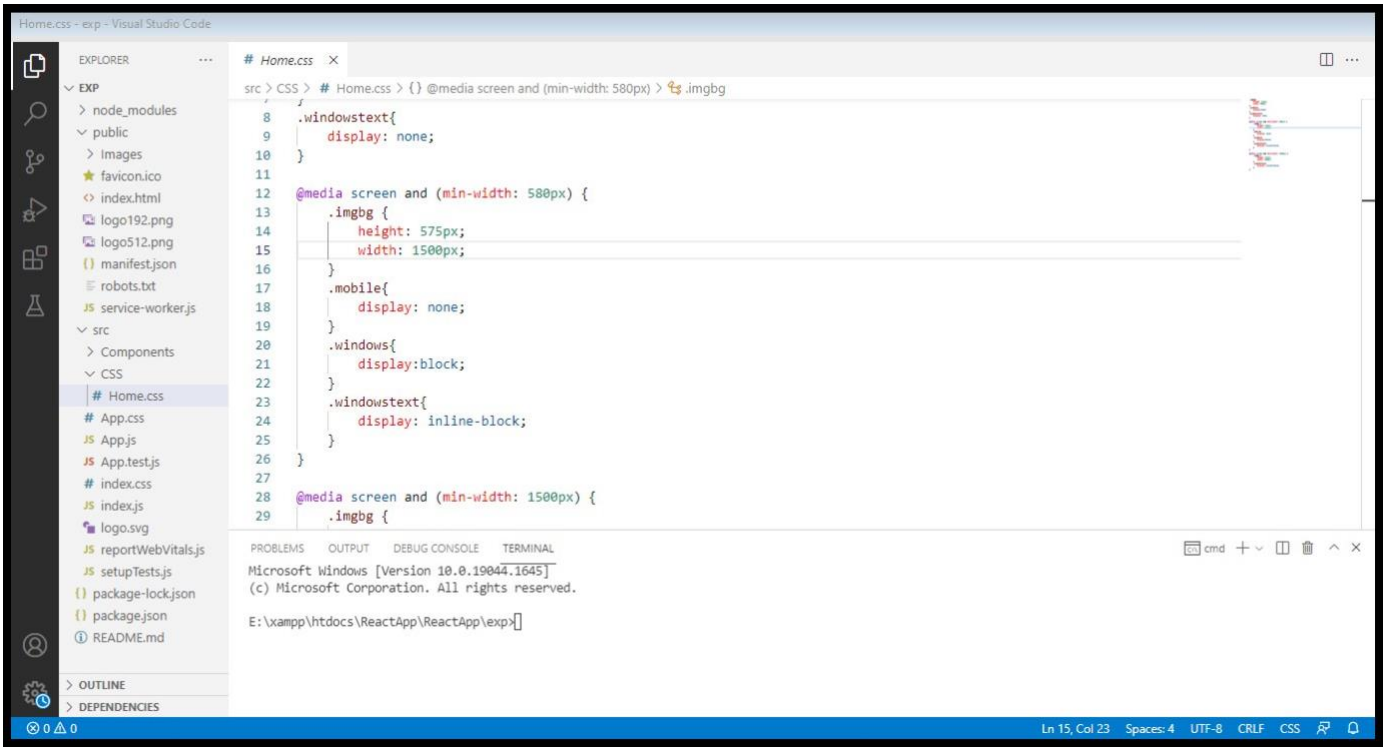
Responsive design is **a graphic user interface (GUI) design approach used to create content that adjusts smoothly to various screen sizes**. Designers size elements in relative units (%) and apply media queries, so their designs can automatically adapt to the browser space to ensure content consistency across devices.

Steps:

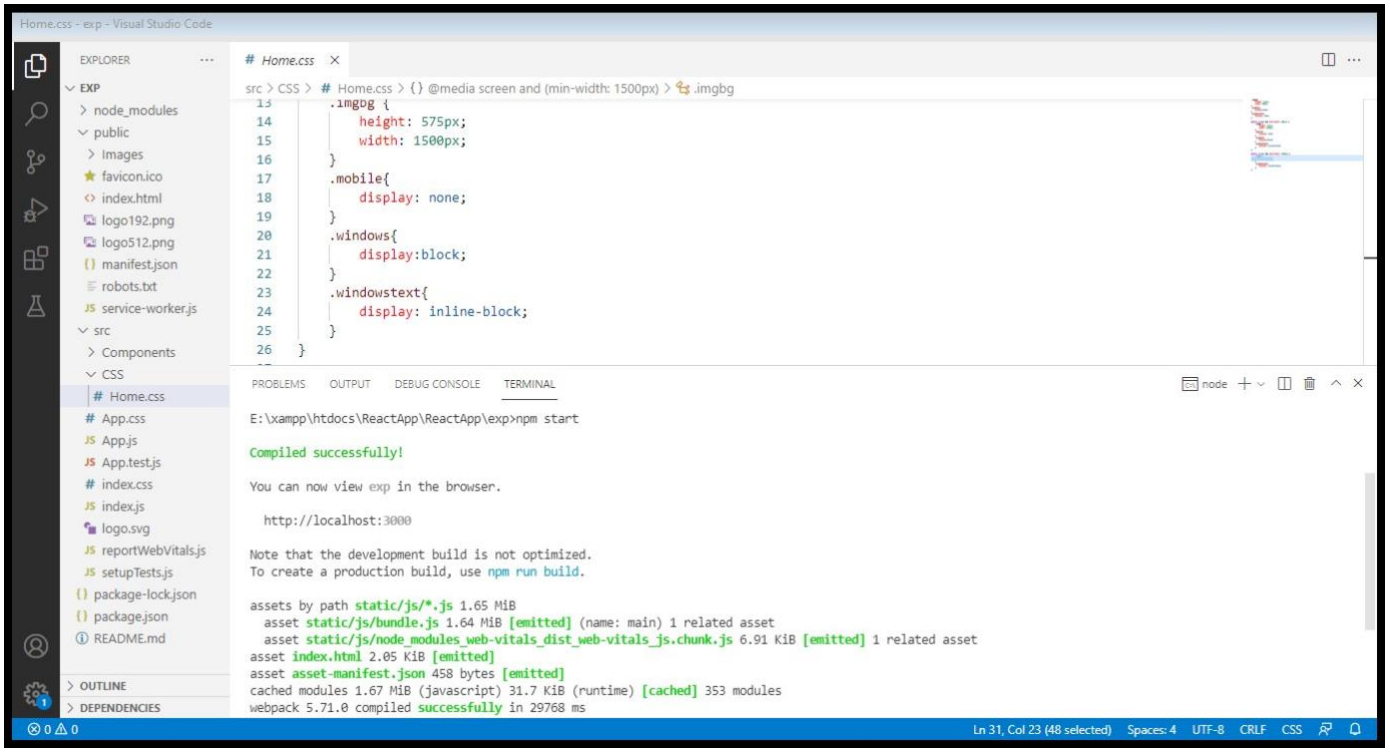
Step1: create app using npm init react-app <react-app-name> command



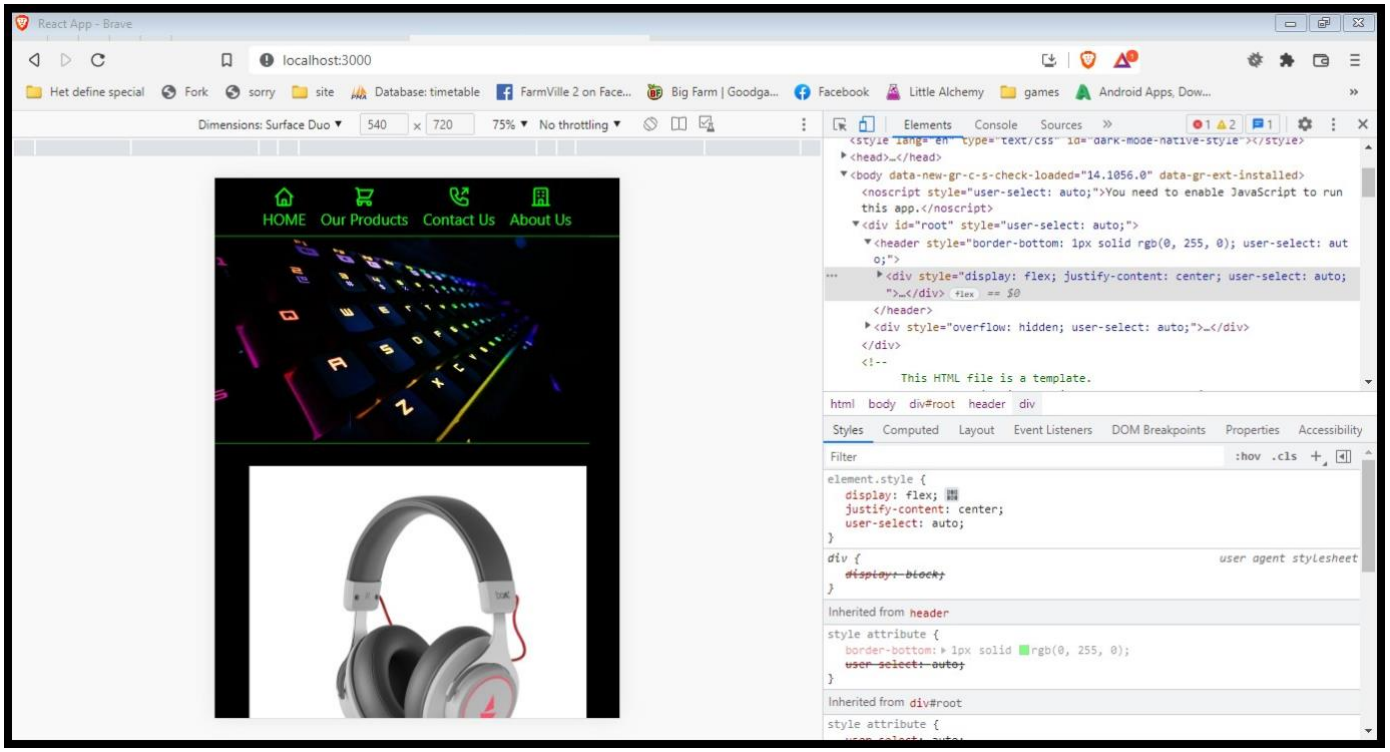
Step2: Write @media rule code in Home.css file

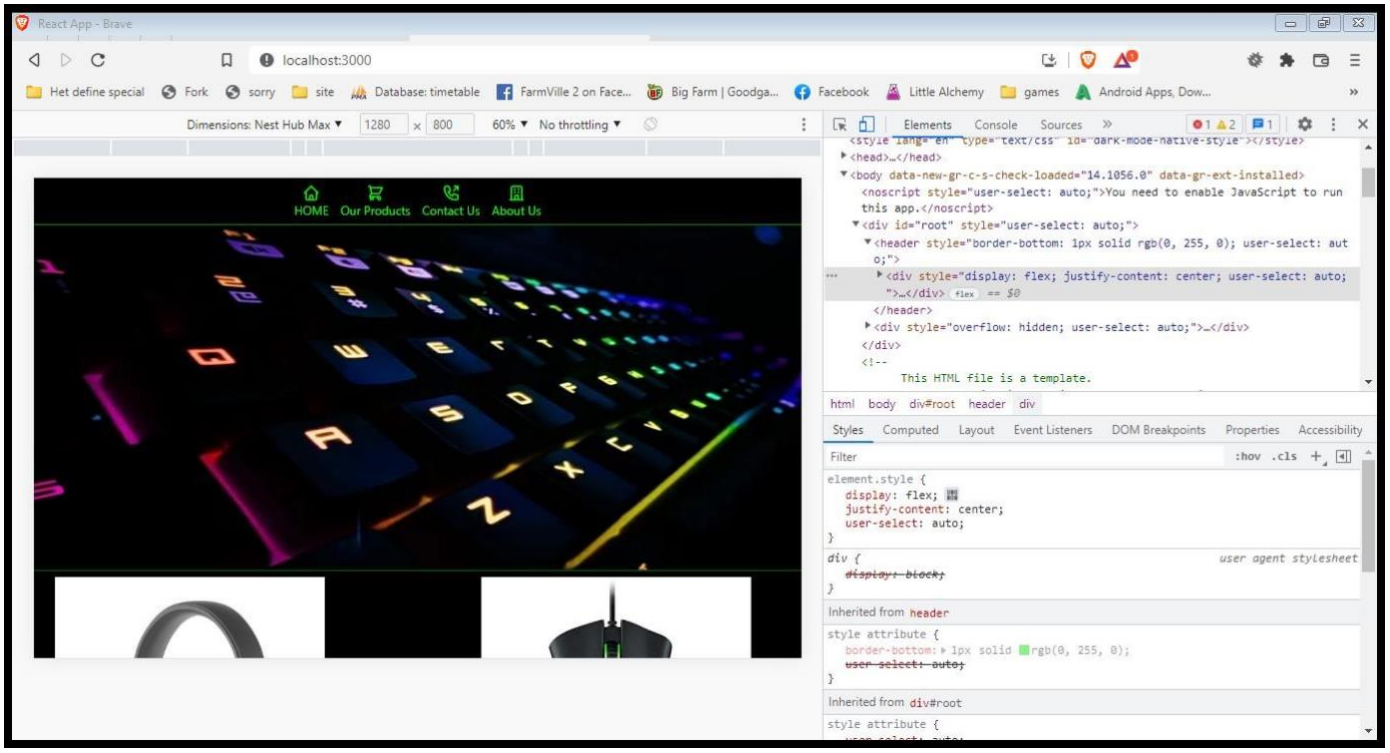


Step3: Run app Using npm start Command



Step4: Output





Conclusion:

We understand how to create a responsive User Interface using jQuery Mobile/ Material UI/ Angular UI/ React UI for Ecommerce application.

EXPERIMENT NO. 6

Aim: To write meta data of your Ecommerce PWA in a Web app manifest file to enable "add to homescreen feature".

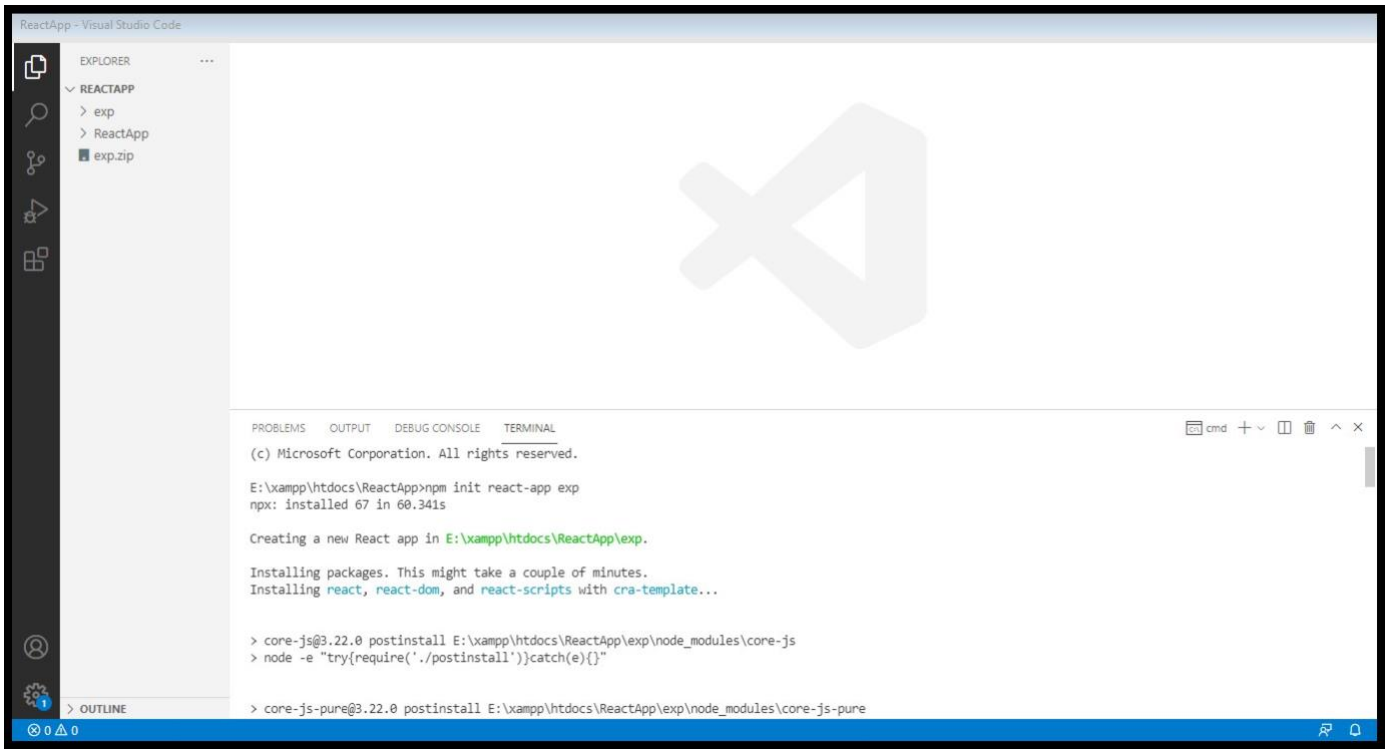
Theory:

What is a PWA manifest?

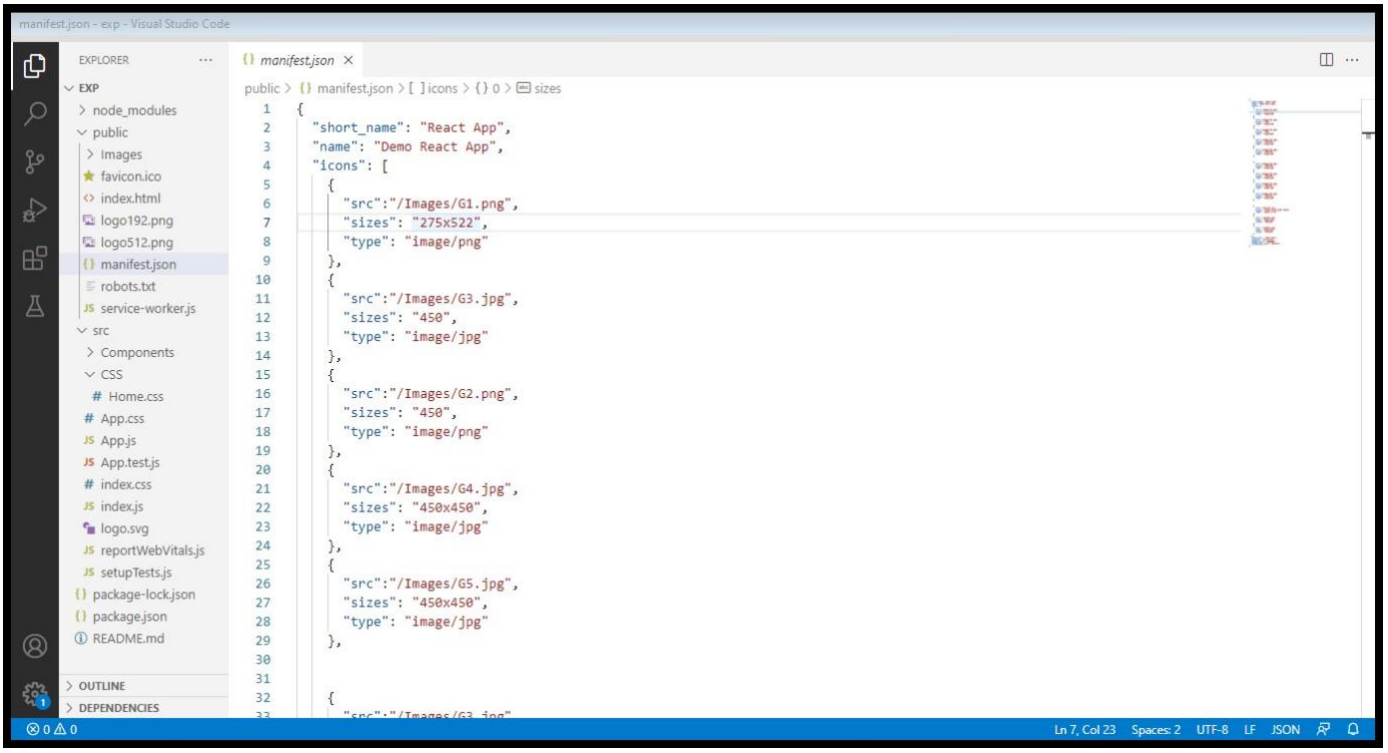
The web app manifest is a JSON file that tells the browser about your Progressive Web App and how it should behave when installed on the user's desktop or mobile device. A typical manifest file includes the app name, the icons the app should use, and the URL that should be opened when the app is launched

Steps:

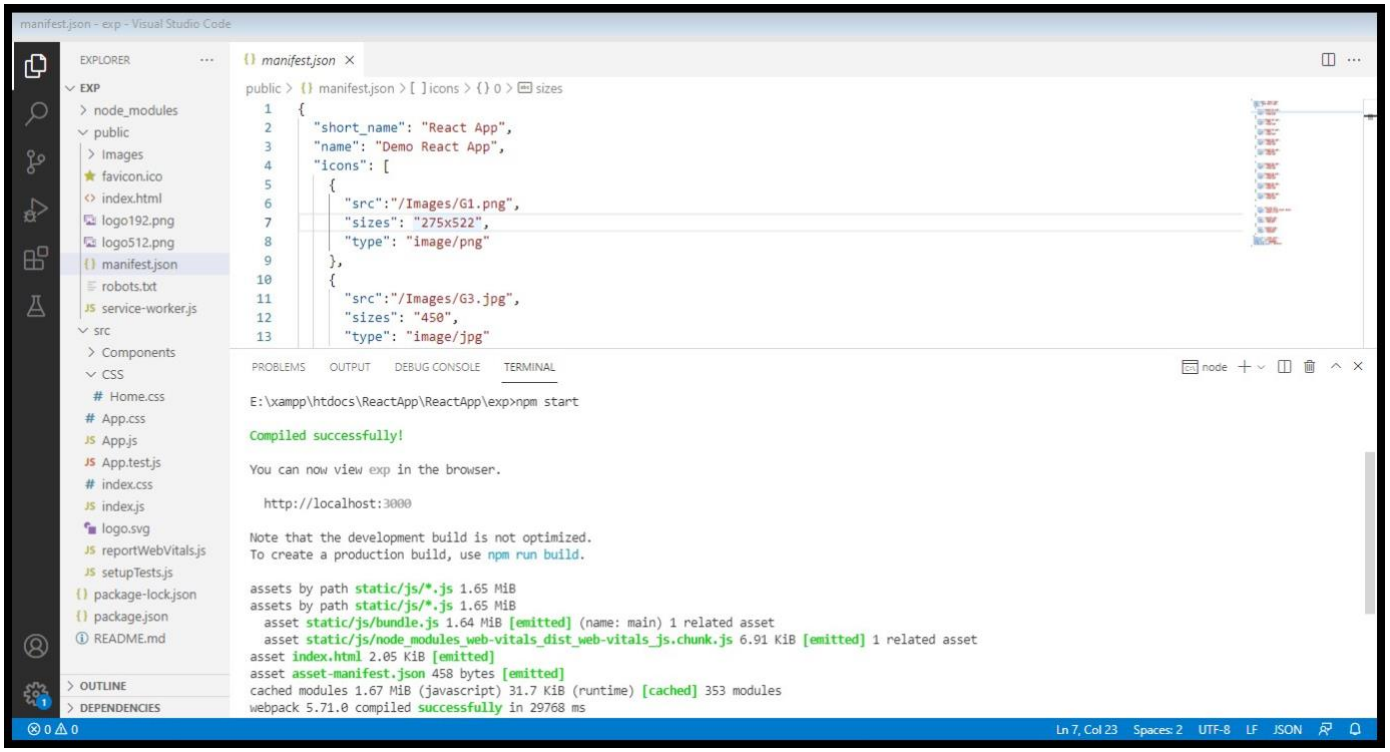
Step1: create app using npm init react-app <react-app-name> command



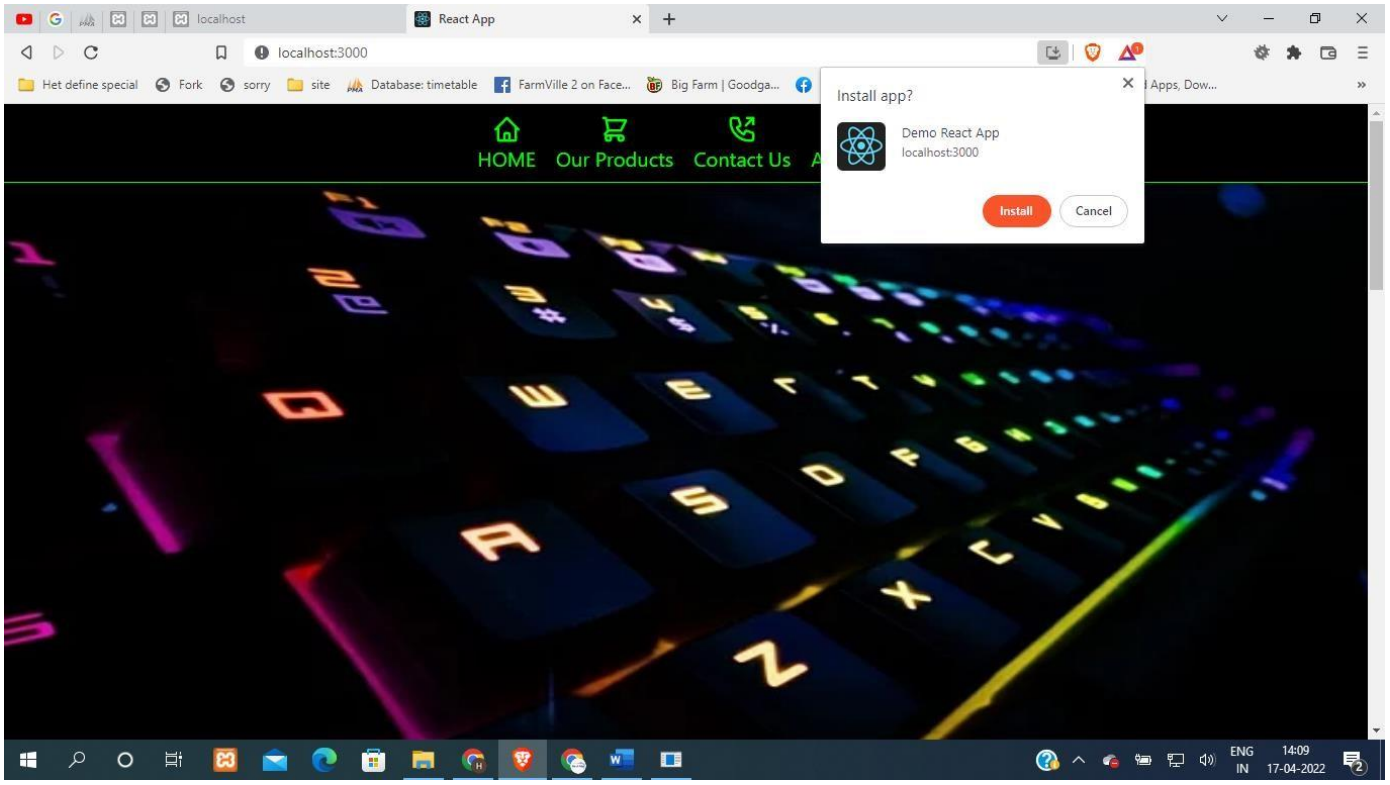
Step2: Write code in manifest.json file



Step3: Run app Using npm start Command



Step4: Output



Conclusion:

We understand how to write meta data of your Ecommerce PWA in a Web app manifest file to enable "add to homescreen feature".

EXPERIMENT NO. 7

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

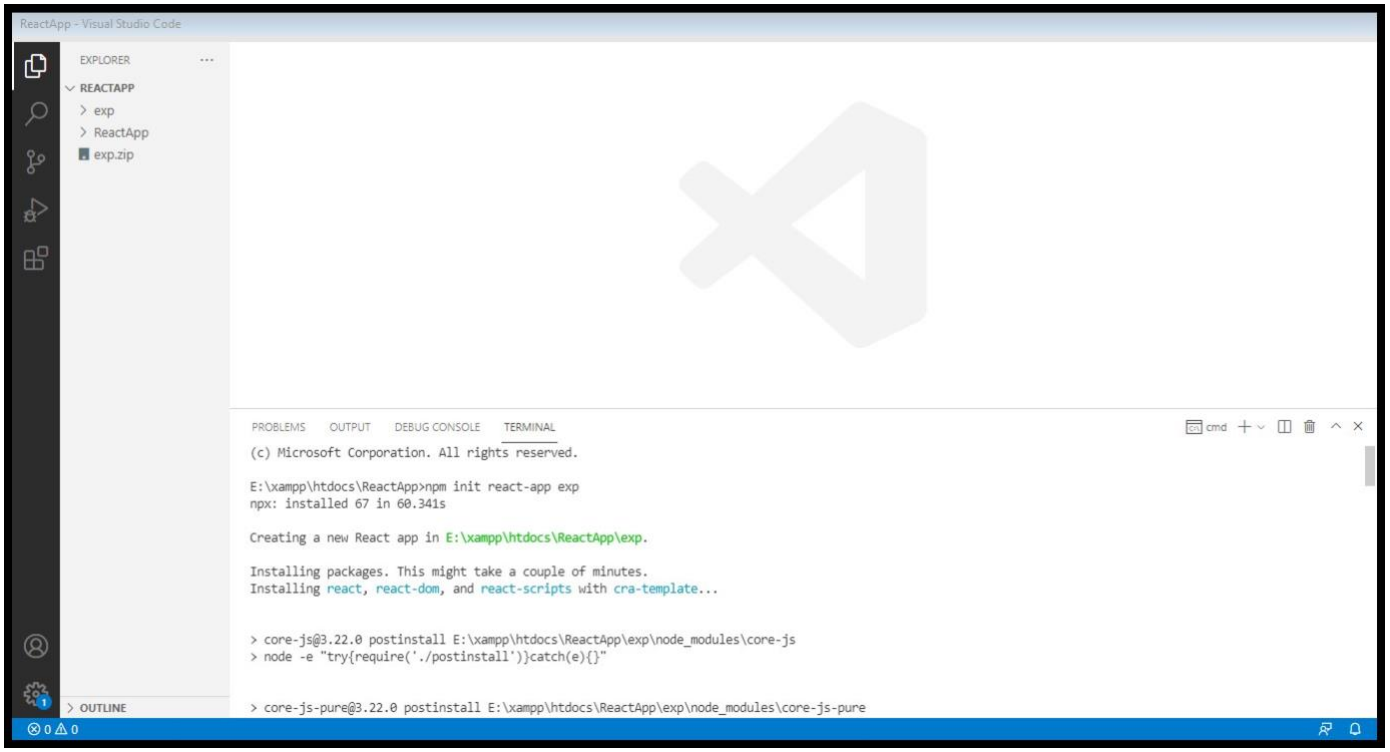
Theory:

What is a service worker PWA?

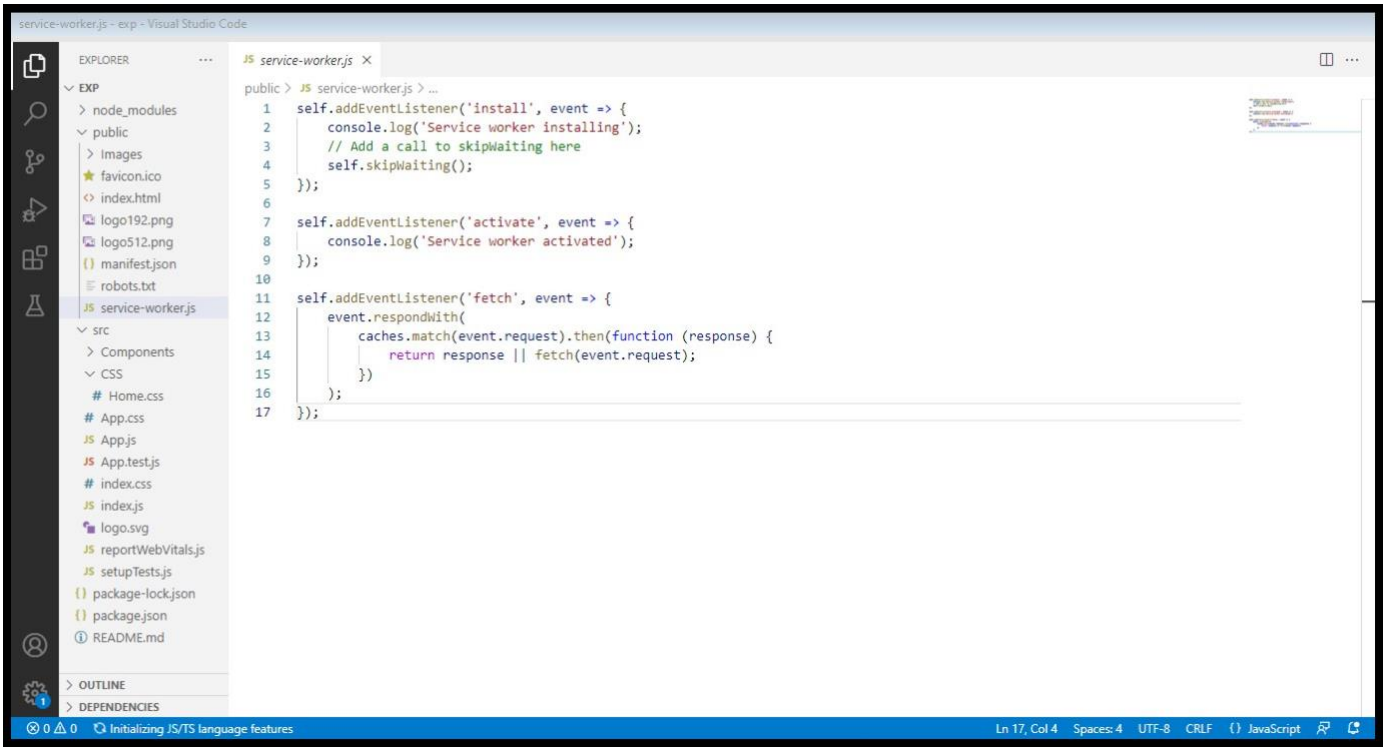
Service workers are a fundamental part of a PWA. They enable fast loading (regardless of the network), offline access, push notifications, and other capabilities.

Steps:

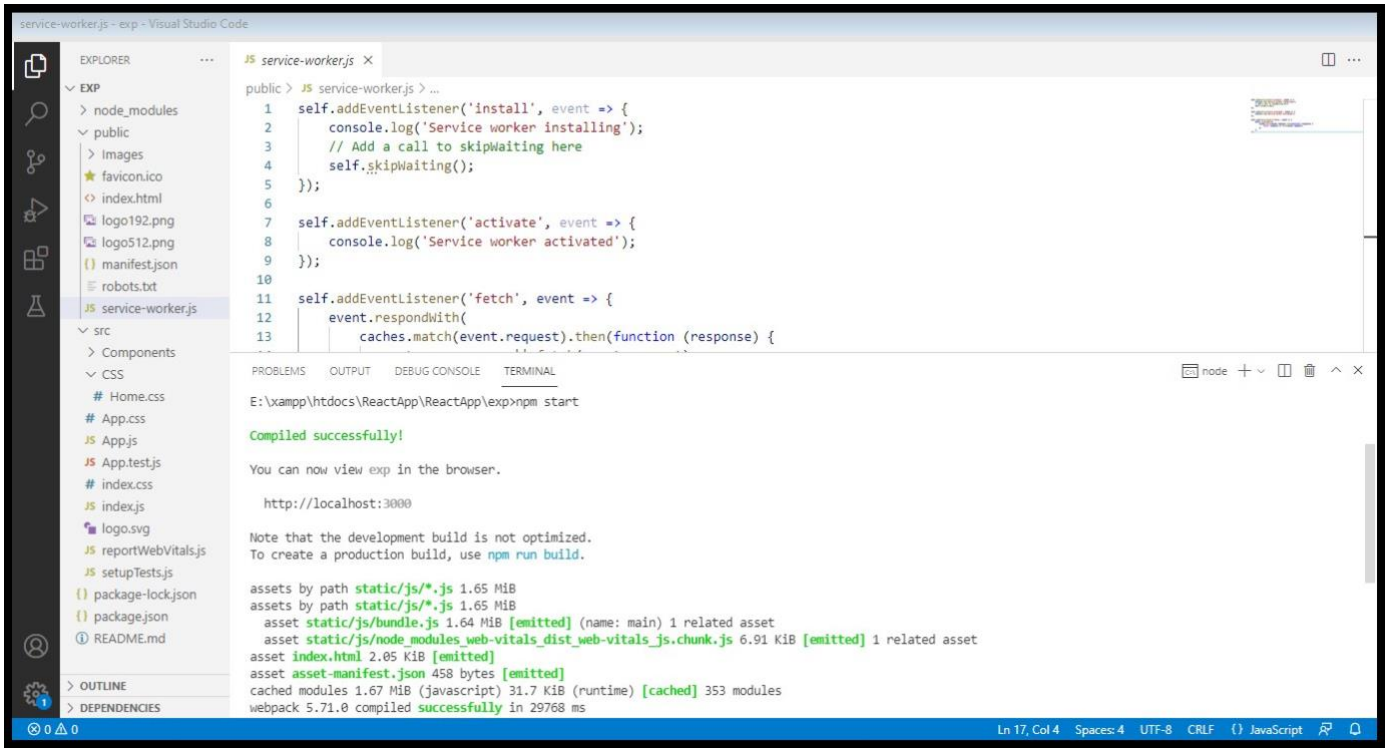
Step1: create app using npm init react-app <react-app-name> command



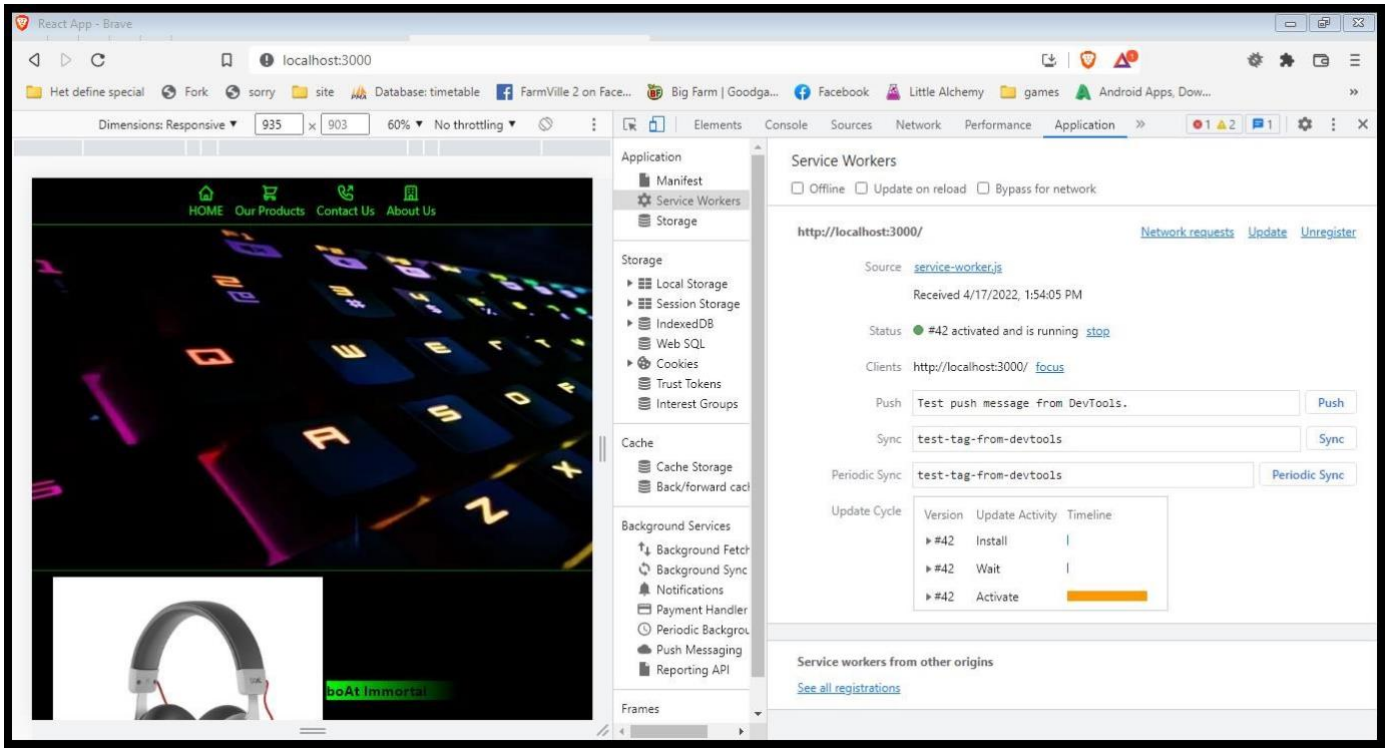
Step2: Write service-worker code in service-worker.js file



Step3: Run app Using npm start Command



Step4: Output



Conclusion:

We understand how to code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

EXPERIMENT NO. 8

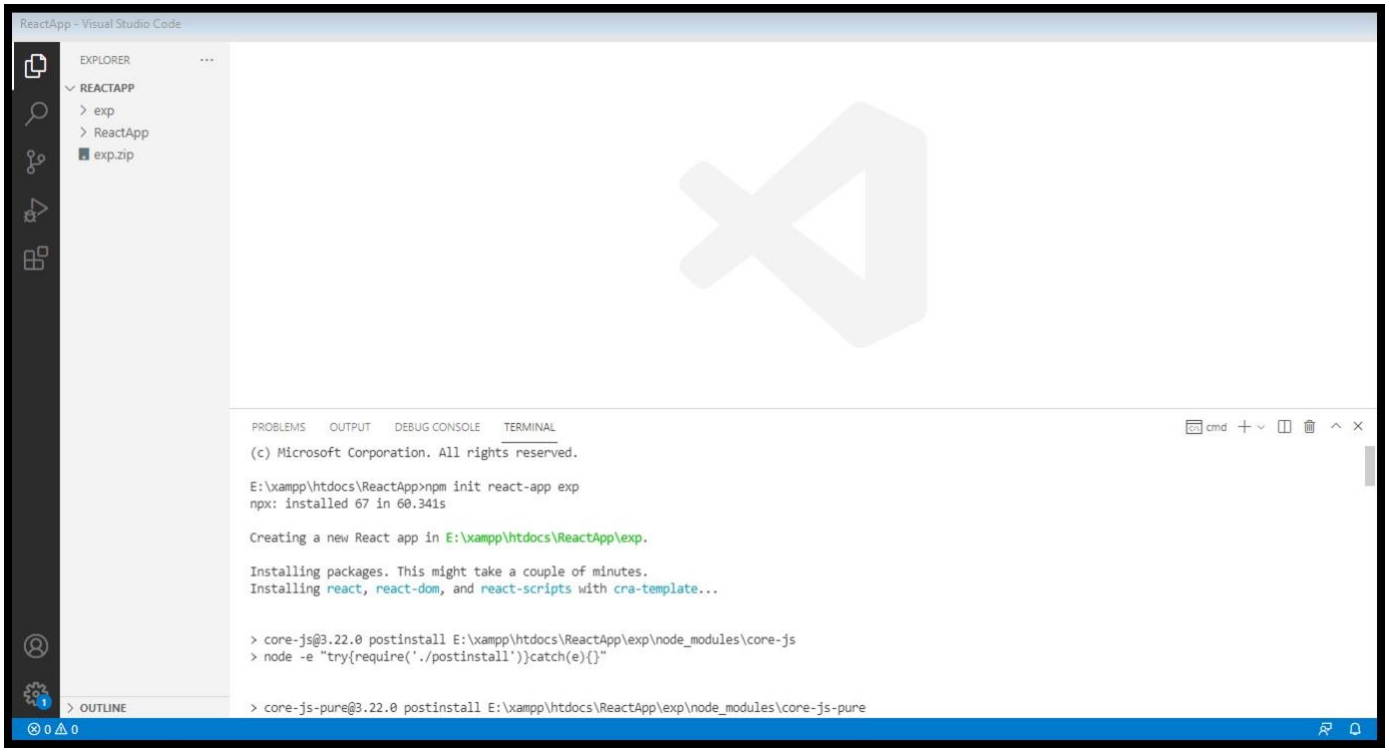
Aim: To use google Lighthouse PWA Analysis Tool to test the PWA functioning.

Theory:

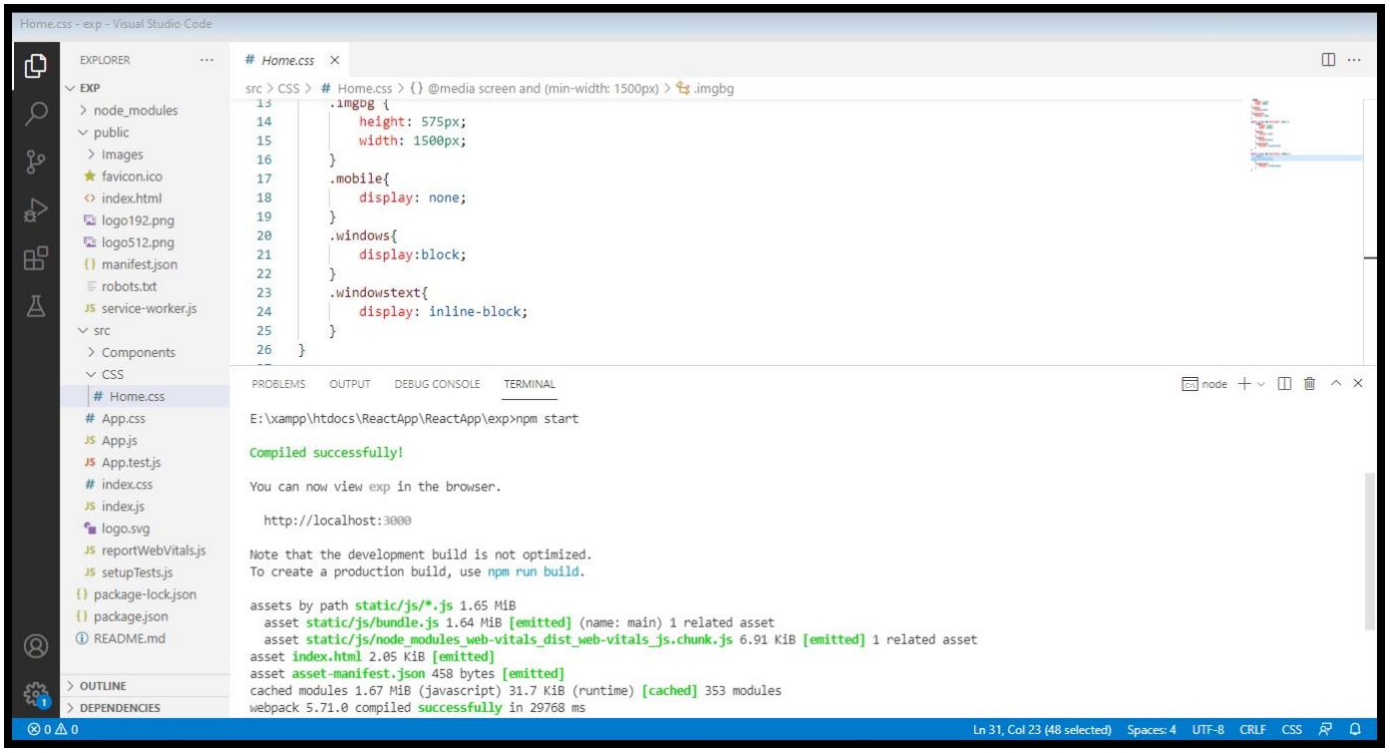
Google Lighthouse is an open-source, automated tool for measuring the quality of web pages. It can be run against any web page, public or requiring authentication. Google Lighthouse audits performance, accessibility and search engine optimization of web pages.

Steps:

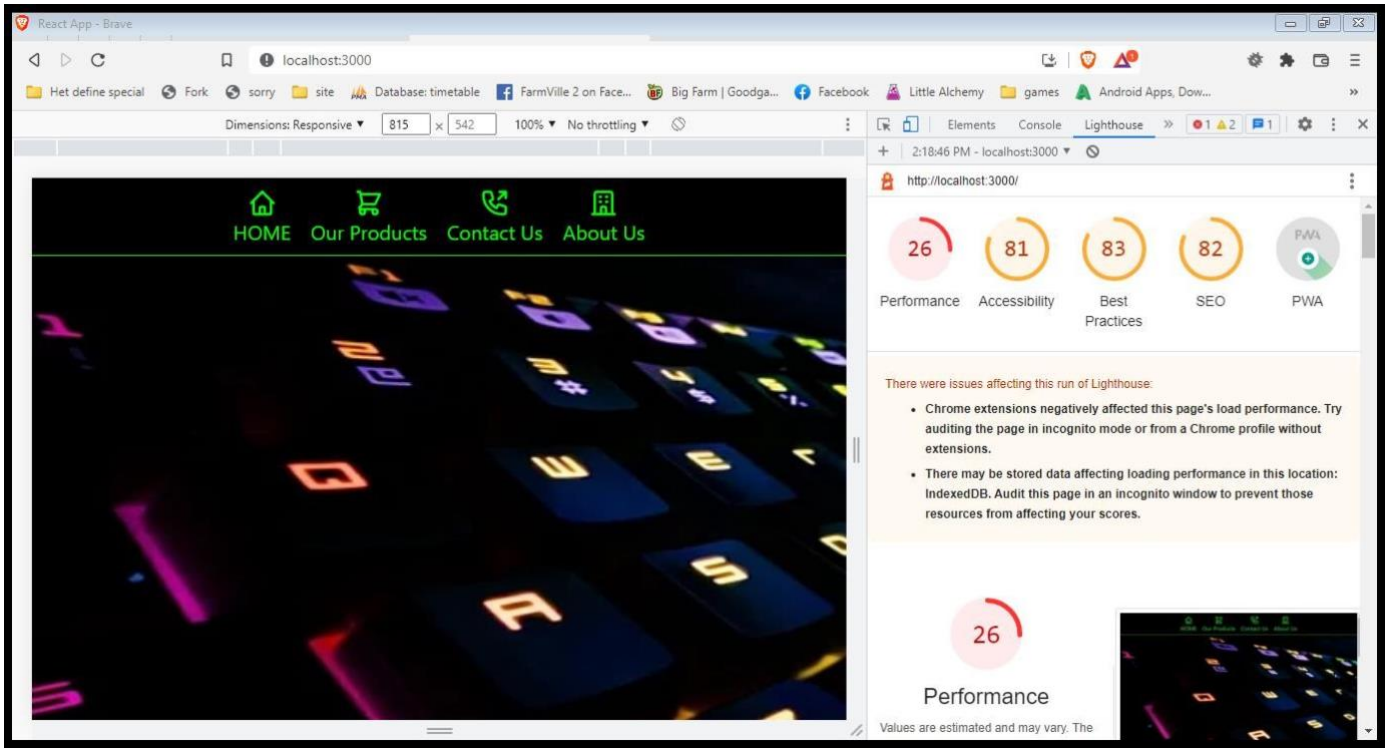
Step1: create app using npm init react-app <react-app-name> command



Step2: Run app Using npm start Command



Step3: Output



Conclusion:

We understand how to use google Lighthouse PWA Analysis Tool to test the PWA functioning.