1. Aim:- Android Environment Setup using Android Studio IDE

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

- Android Studio is the Official IDE for Google's Operating System that is Android.
- It is built on JetBrains' IntelliJ IDEA software, especially for Android Application
- Development.
- This can be used on Windows, macOS, and also Linux based operating systems.
- Google announced Android Studio as the primary IDE for native android apps on
- 16th May 2013 at the Google I/O conference.
- Java was the official language of Android Studio for Android app development,
- which was recently replaced by Kotlin on 7th May 2019.

Features of Android Studio

- It has Gradle-built support
- It lets debug the code easily and has quick fixes.
- There are lint tools that catch performance issues, usability, or version compatibility problems.
- It has a template-based wizard that helps in creating common Android designs and layouts.
- This has a rich editor that allows us to drag and drop the UI components and check it very easily.
- It has a rich color preview editor that lets us add the color and the color name in the resource directory.

Gradle is a build system, which is responsible for code compilation, testing, deployment and conversion of the code into .dex files and hence running the app on the device.

Note: https://www.studytonight.com/android/introduction-to-gradle

In development of an Android application, the first step is to configure a computer system to act as the development platform. This involves installing

(1) Java Development Kit (JDK) and

(2) Android Studio Integrated Development Environment (IDE) which also includes the Android Software Development Kit (SDK)

System Requirements

Android application development may be performed on any of the following system types:

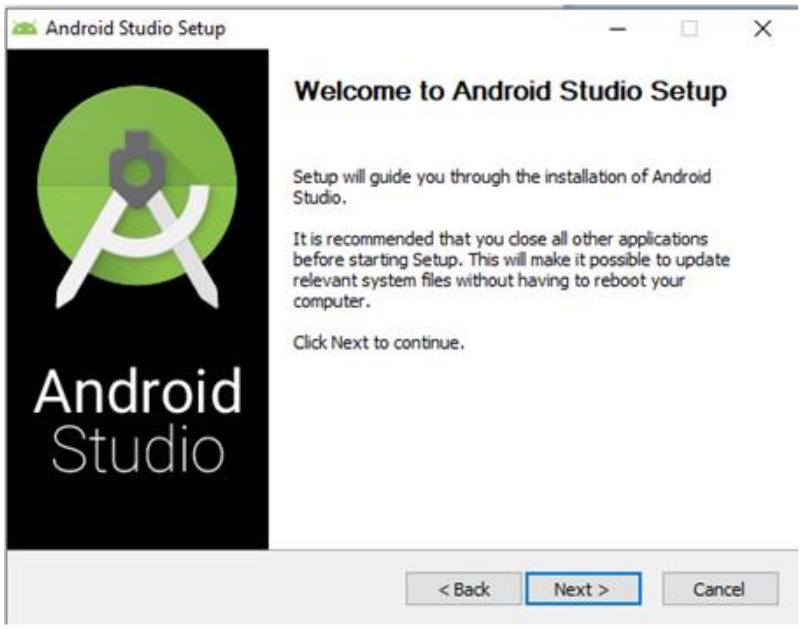
- Microsoft Windows 7/8/8.1/10/11 or later (32-bit or 64-bit)
- Mac OS X 10.10 or later (Intel based systems only)
- Linux systems GNOME or KDE desktop

- Minimum of 4GB of RAM (8/12/16 GB is preferred) (plus 1 GB for the Android Emulator)
- Approximately 4GB of available disk space ((500 MB for Android Studio and 1.5 GB for the Android SDK and emulator system image)

To set up the Android development environment using Android Studio, you will need to perform the following steps:

Install Java: Android development requires the Java Development Kit (JDK) to be installed on your system. You can download the JDK from the Oracle website and follow the instructions to install it on your system.
Install Android Studio: Android Studio is the official integrated development environment (IDE) for Android development. You can download Android Studio from the Android website and follow the instructions to install it on your system.
Install the Android SDK: The Android SDK is a set of tools and libraries that are required for Android development. Android Studio includes the Android SDK, but you may need to install additional components or update to the latest version. To do this, open Android Studio and go to the "SDK Manager" (found under the "Tools" menu). Select the desired components and click "Apply" to install them.
Create an Android Virtual Device (AVD): An AVD is a virtual device that allows you to test your Android apps on your computer. To create an AVD, open the AVD Manager (found in the "Tools" menu) and follow the prompts to create a new virtual device.
Set up the emulator: The emulator is a tool that allows you to run and test your Android apps on your computer. To set up the emulator, open the AVD Manager and select your desired AVD. Click the "Start" button to launch the emulator and begin testing your app.

Installing Android Studio
Installation on Windows
Locate and double click Android Studio installation executable file to start the installation process.


Android Studio Setup

Welcome to Android Studio Setup

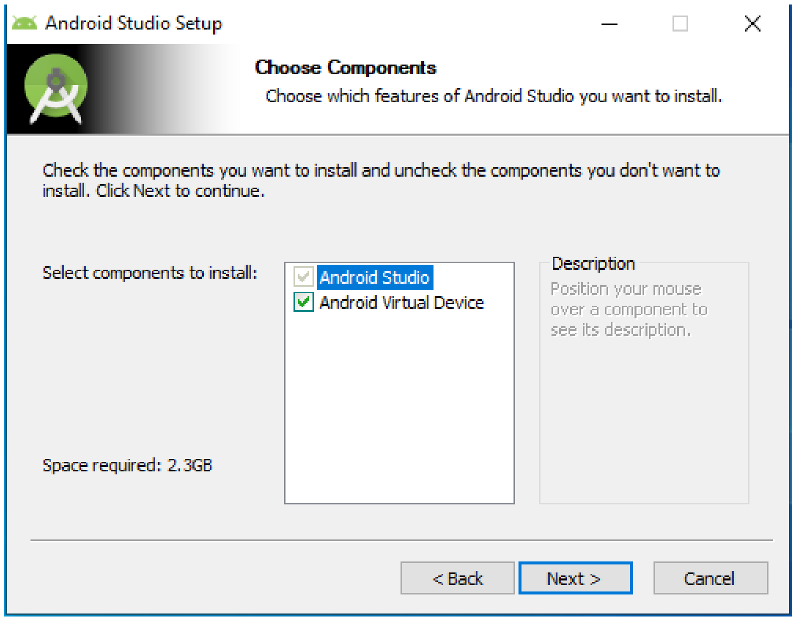
Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Once the Android Studio setup wizard appears, work through the various screens to configure the installation in terms of the file system location into which Android Studio should be installed and whether or not it should be made available to other users of the system.

When prompted to select the components to install, make sure that the Android Studio, Android Virtual Device options are all selected.


Android Studio Setup

Choose Components

Choose which features of Android Studio you want to install.

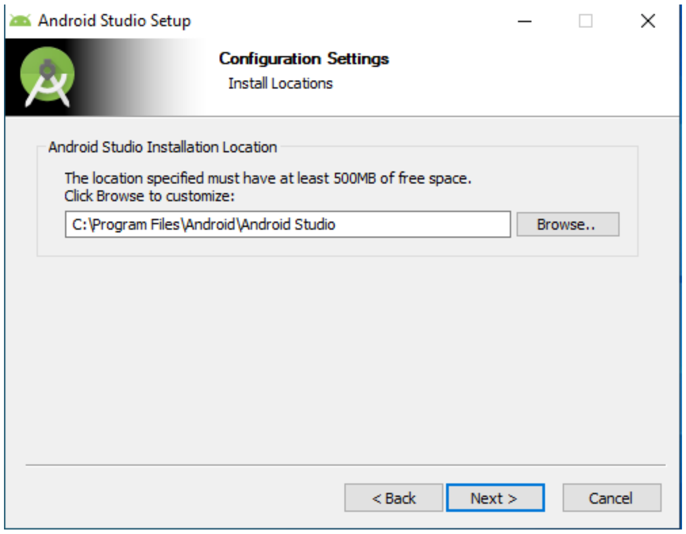
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- [✓] Android Studio
- [✓] Android Virtual Device

Description
Position your mouse over a component to see its description.

Space required: 2.3GB

Specify the location of local machine path for Android studio


Android Studio Setup

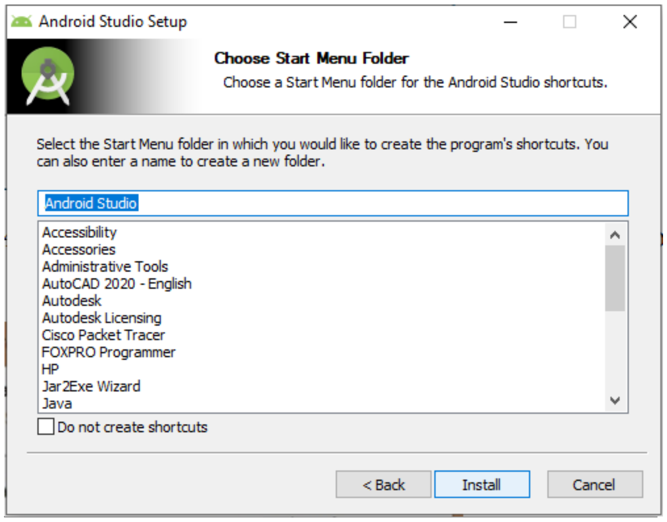
Configuration Settings
Install Locations

Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

C:\Program Files\Android\Android Studio

Choose Start Menu Folder and once the options have been configured, *click on the Install button* to begin the installation process.


Android Studio Setup

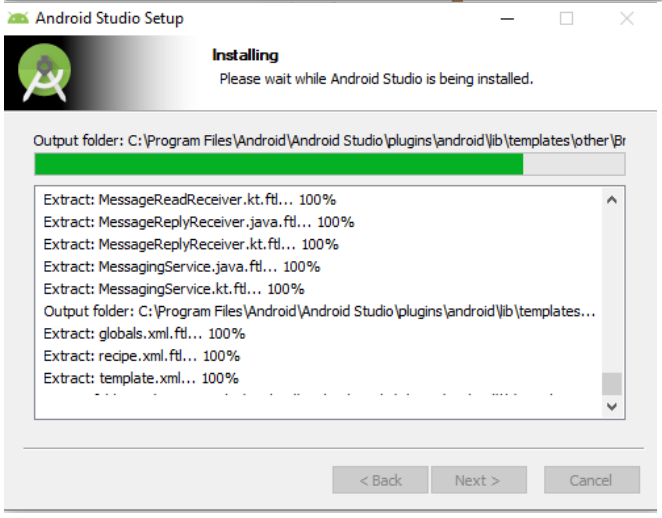
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Android Studio

Accessibility
Accessories
Administrative Tools
AutoCAD 2020 - English
Autodesk
Autodesk Licensing
Cisco Packet Tracer
FOXPRO Programmer
HP
Jar2Exe Wizard
Java

Do not create shortcuts

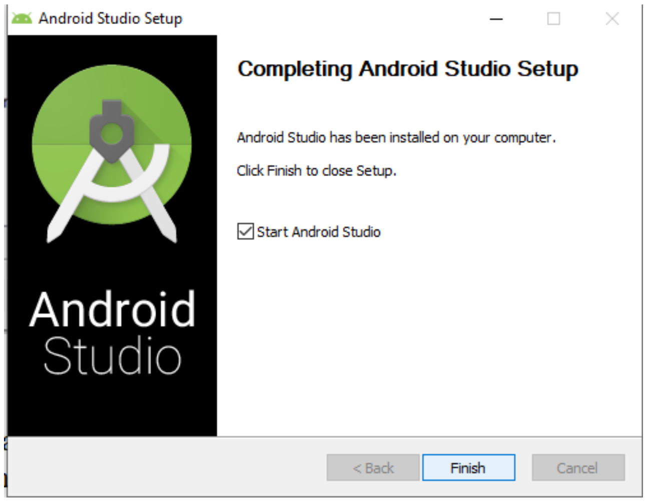

Android Studio Setup

Installing
Please wait while Android Studio is being installed.

Output folder: C:\Program Files\Android\Android Studio\plugins\android\lib\templates\other\Br

```
Extract: MessageReadReceiver.kt.ftl... 100%
Extract: MessageReplyReceiver.java.ftl... 100%
Extract: MessageReplyReceiver.kt.ftl... 100%
Extract: MessagingService.java.ftl... 100%
Extract: MessagingService.kt.ftl... 100%
Output folder: C:\Program Files\Android\Android Studio\plugins\android\lib\templates...
Extract: globals.xml.ftl... 100%
Extract: recipe.xml.ftl... 100%
Extract: template.xml... 100%
```

Once the installation completed click on Finish by selecting Start Android Studio check box


Android Studio Setup

Completing Android Studio Setup

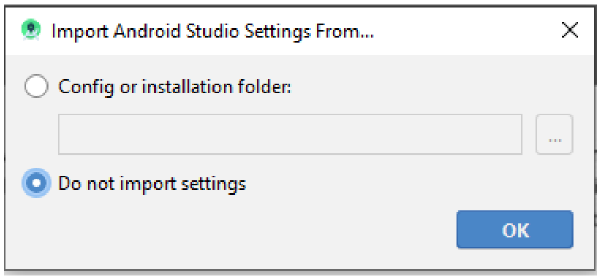
Android Studio has been installed on your computer.

Click Finish to close Setup.

☑ Start Android Studio

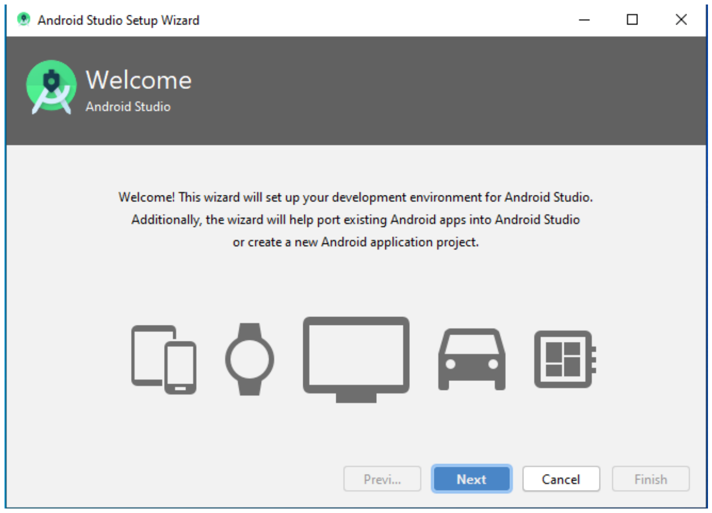
The Android Studio Setup Wizard

The first time that Android Studio is launched after being installed, a dialog will appear providing the option to import settings from a previous Android Studio version. If you have settings from a previous version and would like to import them into the latest installation, select the appropriate option and location. Alternatively, indicate that you do not need to import any previous settings and click on the OK button to proceed.


Import Android Studio Settings From...

Config or installation folder: [✓ Do not import settings]

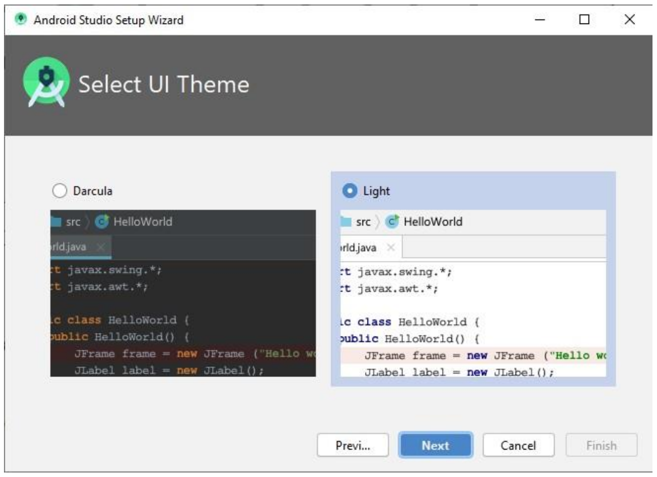
Next, the setup wizard may appear as shown below, but this dialog may not appear on allplatforms:


Android Studio Setup Wizard

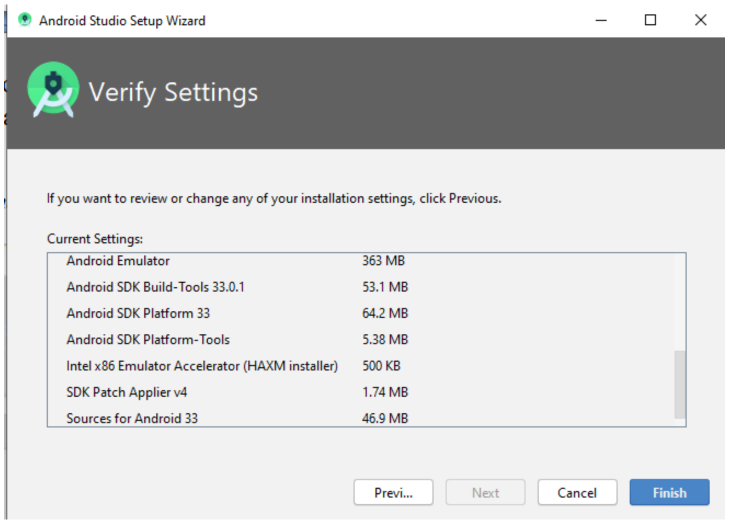
Welcome
Android Studio

Welcome! This wizard will set up your development environment for Android Studio.
Additionally, the wizard will help port existing Android apps into Android Studio
or create a new Android application project.

If the wizard appears, click on the Next button, choose the Standard installation option, Light theme and click on Next once again.



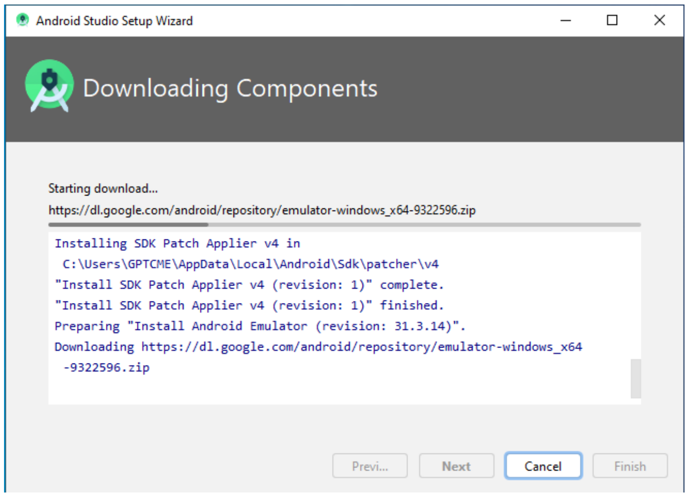
Android Studio will proceed to download and configure the latest Android SDK and some additional components and packages. Once this process has completed, click on the Finish button.



If you want to review or change any of your installation settings, click Previous.

Current Settings:

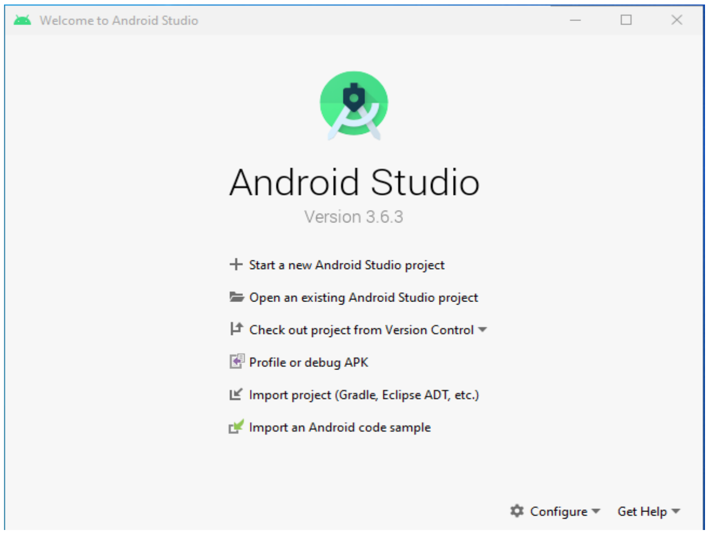
Android Emulator	363 MB
Android SDK Build-Tools 33.0.1	53.1 MB
Android SDK Platform 33	64.2 MB
Android SDK Platform-Tools	5.38 MB
Intel x86 Emulator Accelerator (HAXM installer)	500 KB
SDK Patch Applier v4	1.74 MB
Sources for Android 33	46.9 MB



Starting download...
https://dl.google.com/android/repository/emulator-windows_x64-9322596.zip

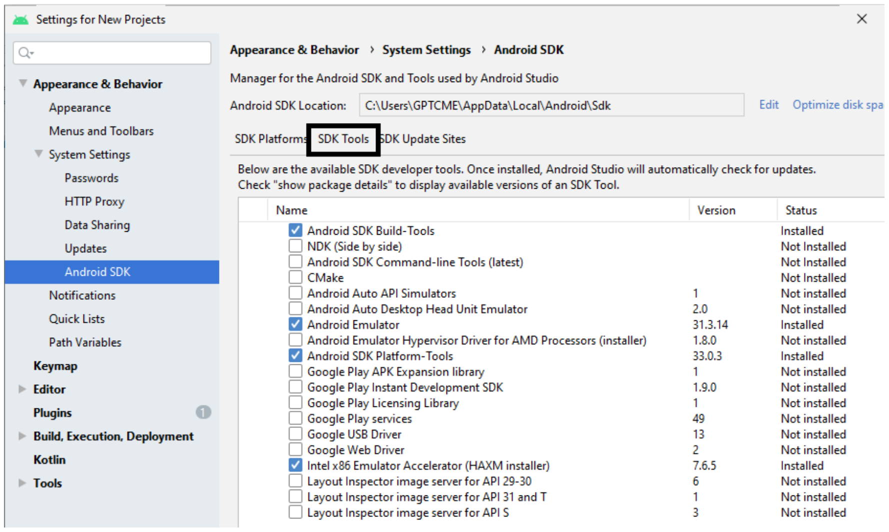
```
Installing SDK Patch Applier v4 in
 C:\Users\GPTCME\AppData\Local\Android\Sdk\patcher\v4
"Install SDK Patch Applier v4 (revision: 1)" complete.
"Install SDK Patch Applier v4 (revision: 1)" finished.
Preparing "Install Android Emulator (revision: 31.3.14)".
Downloading https://dl.google.com/android/repository/emulator-windows_x64
 -9322596.zip
```

Next, the Welcome to Android Studio screen will appear


Welcome to Android Studio
Android Studio
Version 3.6.3
+ Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

Installing Additional Android SDK Packages

To view the currently installed packages and check for updates, go to select Configure-> SDK Manager Option from the Android Studio welcome dialog and select the SDK Tools tab.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\GPTCME\AppData\Local\Android\Sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
✓ Android SDK Build-Tools		Installed
NDK (Side by side)		Not Installed
Android SDK Command-line Tools (latest)		Not Installed
CMake		Not Installed
Android Auto API Simulators	1	Not installed
Android Auto Desktop Head Unit Emulator	2.0	Not installed
✓ Android Emulator	31.3.14	Installed
Android Emulator Hypervisor Driver for AMD Processors (installer)	1.8.0	Not installed
✓ Android SDK Platform-Tools	33.0.3	Installed
Google Play APK Expansion library	1	Not installed
Google Play Instant Development SDK	1.9.0	Not installed
Google Play Licensing Library	1	Not installed
Google Play services	49	Not installed
Google USB Driver	13	Not installed
Google Web Driver	2	Not installed
✓ Intel x86 Emulator Accelerator (HAXM installer)	7.6.5	Installed
Layout Inspector image server for API 29-30	6	Not installed
Layout Inspector image server for API 31 and T	1	Not installed
Layout Inspector image server for API S	3	Not installed

Make sure that the following packages are listed as Installed in the Status column:

- Android SDK Build-tools
- Android SDK Platform-tools
- Android Emulator
- Intel x86 Emulator Accelerator (HAXM installer).

2. Aim:- Android Environment Setup using Eclipse IDE

Resources Required: A computer system with Android Studio IDE installed.

Procedure:
To set up the Android development environment using Eclipse IDE, you will need to perform the following steps:
Install the Java Development Kit (JDK): Android development requires the JDK to be installed on your system. You can download the JDK from the Oracle website and follow the instructions to install it on your system.
Install Eclipse: Eclipse is an open-source integrated development environment (IDE) that is commonly used for Android development. You can download Eclipse from the Eclipse website and follow the instructions to install it on your system.
Install the Android Development Tools (ADT) plugin: The ADT plugin is a set of tools that provides integration between Eclipse and the Android SDK. To install the ADT plugin, open Eclipse and go to the "Help" menu. Select "Install New Software" and enter the URL for the ADT plugin (https://dl-ssl.google.com/android/eclipse/). Follow the prompts to install the plugin.
Install the Android SDK: The Android SDK is a set of tools and libraries that are required for Android development. To install the Android SDK, download it from the Android website and follow the instructions to install it on your system.
Configure the Android SDK: Once the Android SDK is installed, you will need to configure it to use the correct versions of the Android platform and tools. To do this, open the Android SDK Manager (found in the Android SDK folder) and select the desired platform and tools.
Create an Android Virtual Device (AVD): An AVD is a virtual device that allows you to test your Android apps on your computer. To create an AVD, open the AVD Manager (found in the Android SDK folder) and follow the prompts to create a new virtual device.

Setting up the Android development environment using Eclipse IDE involves installing the required software and configuring the tools and libraries needed for Android development.

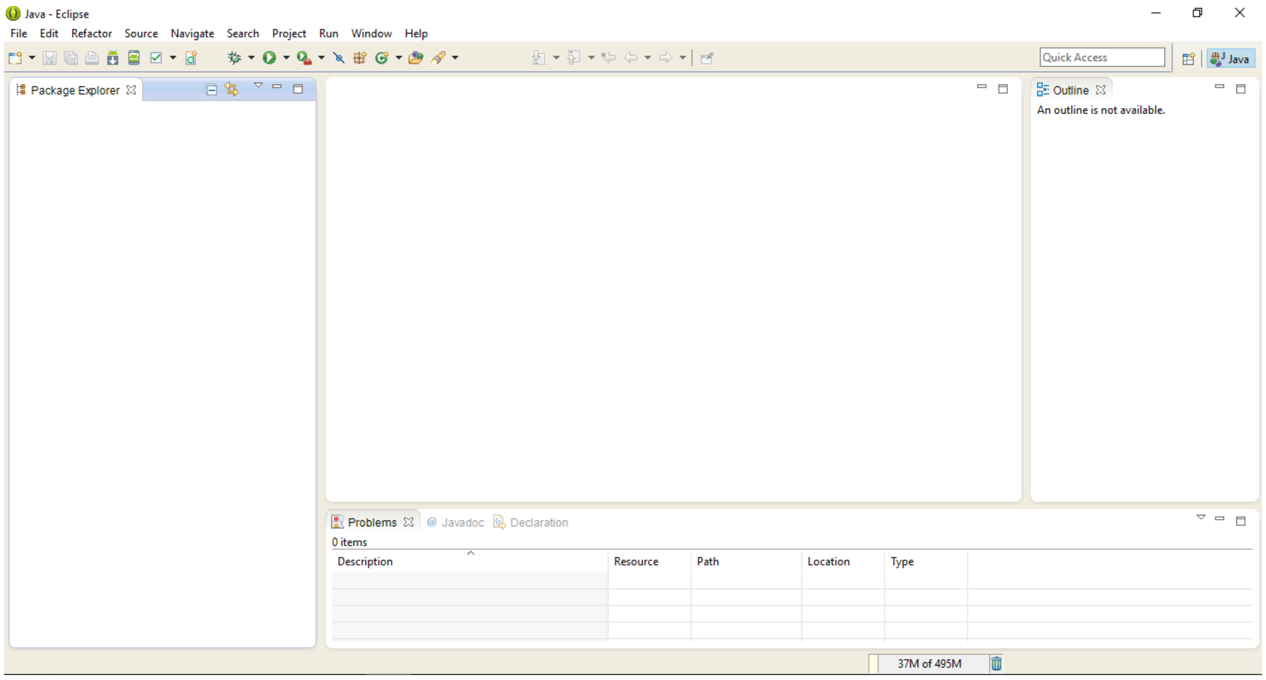
1. Install Java JDK latest version
2. Download and Install Eclipse IDE
3. Install Android Development Tools (ADT) Eclipse Plug-in
4. Install and configure Android SDK
5. Create Android Virtual Device (AVD)

Step 1: Setup Java Development Kit (JDK)

You can download the JDK and install it, which is pretty easy. After that, you just have to set PATH and JAVA_HOME variables to the folder where you have java and javac.

Step 2: Setup Eclipse IDE

Download Eclipse from https://www.eclipse.org/downloads/. Eclipse does not require installation we can run it directly from the folder where it is saved just by opening the Eclipse executable file



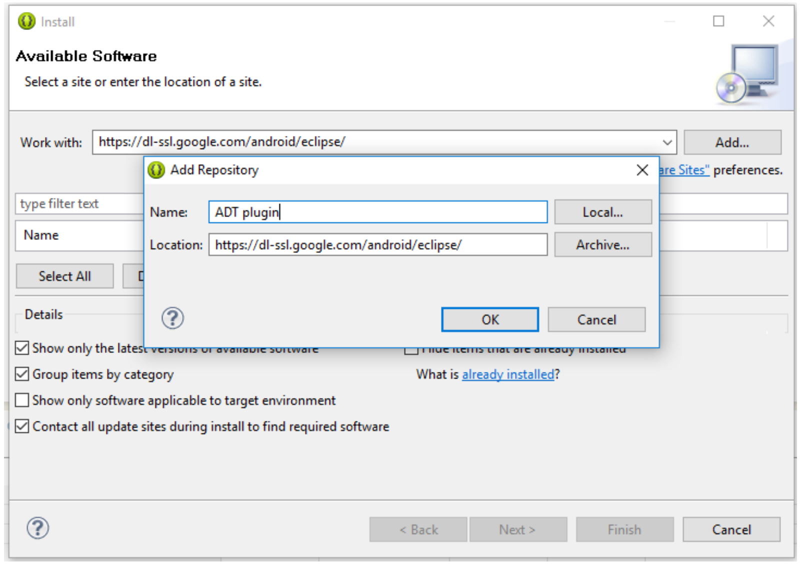
Step 3: Setup Android Development Tools (ADT) Plug-in

Android Development Tools (ADT) is the plug-in for Eclipse IDE that is designed to provide the integrated environment. ADT is required for developing the android application in the eclipse IDE. To install the Android Development Tool (ADT) plug-in for Eclipse, Start the Eclipse IDE and

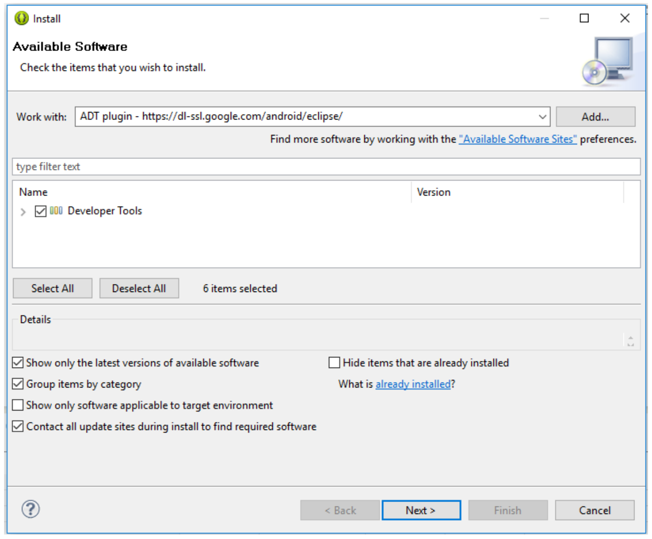
Click on **Help** > **Software Updates** > **Install New Software**. In the work with combo box, click on the Add button and then add the location

https://dl-ssl.google.com/android/eclipse/

Then press OK, now the Eclipse will start to search for the required plug-in and finally it will list the found plug-ins.


Install
Available Software
Select a site or enter the location of a site.
Work with: https://dl-ssl.google.com/android/eclipse/
Add...
Add Repository
Name: ADT plugin
Location: https://dl-ssl.google.com/android/eclipse/
Local...
Archive...
OK
Cancel
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
What is already installed?
< Back
Next >
Finish
Cancel

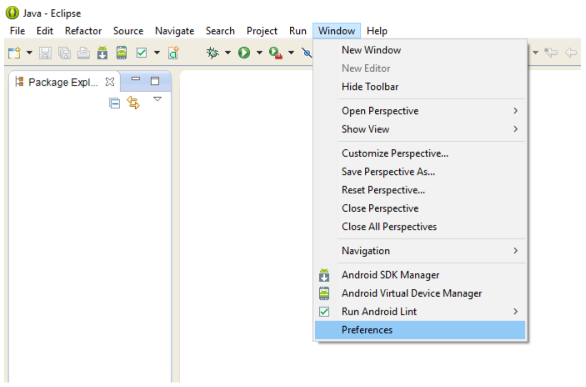
Then select the checkbox next to Developer Tools and click next then a list of tools to be downloaded here, click next then click finish After completing the installation,

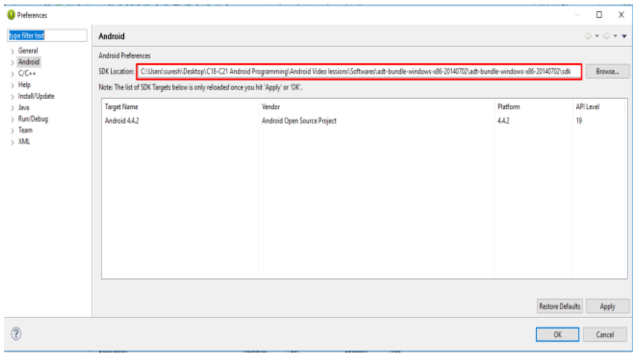

Install
Available Software
Check the items that you wish to install.
Work with: ADT plugin - https://dl-ssl.google.com/android/eclipse/
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
Developer Tools
Select All
Deselect All
6 items selected
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Show only software applicable to target environment
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel

restart the eclipse IDE

After the installing ADT plug-in, now tell the eclipse IDE for your android SDK location for this

1. Select the Window menu > preferences
2. Now select the android from the left panel.
3. Click on the browse button and locate your SDK directory e.g. my SDK location is C:\Users\suresh\Desktop\Android\adt-bundle-windows-x86-20140702\sdk
4. Click the apply button then OK.





Android Preferences

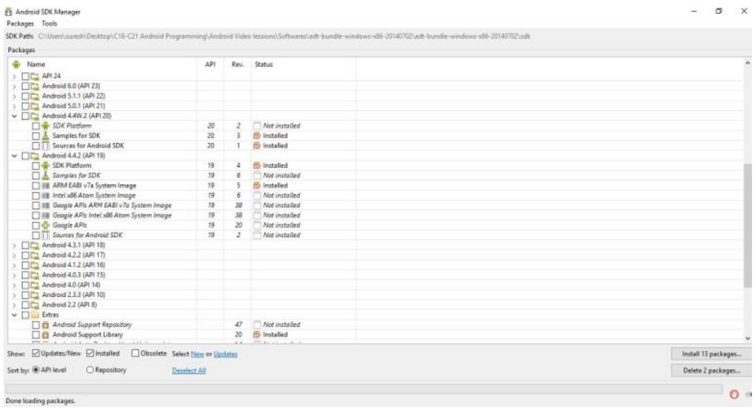
SDK Location: C:\Users\suresh\Desktop\C18-C21 Android Programming\Android Video lessions\Softwares\adt-bundle-windows-x86-20140702\adt-bundle-windows-x86-20140702\sdk

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	API Level
Android 4.4.2	Android Open Source Project	4.4.2	19

Step 4: Configure Android SDK

After successfully installed the Android SDK then configure it. After installing the Android SDK, you will get a window like this, (if not, open SDK manager.exe from the folder where you have installed Android SDK).



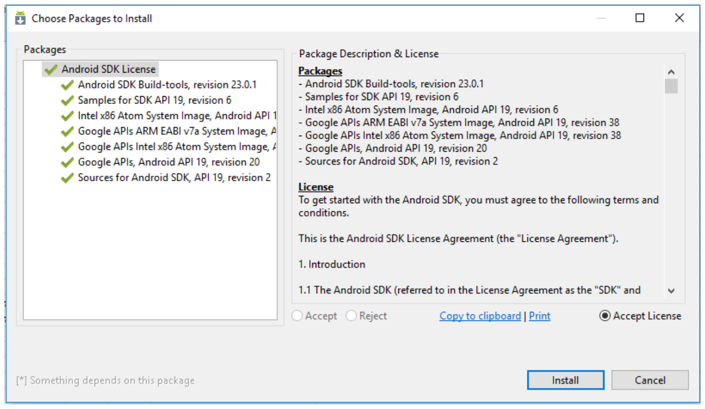
SDK Path: C:\Users\suresh\Desktop\C18-C21 Android Programming\Android Video lessions\Softwares\adt-bundle-windows-x86-20140702\adt-bundle-windows-x86-20140702\sdk

Name	API	Rev.	Status
API 24			
Android 6.0 (API 23)			
Android 5.1.1 (API 22)			
Android 5.0.1 (API 21)			
Android 4.4W.2 (API 20)			
SDK Platform	20	2	Not installed
Samples for SDK	20	3	Installed
Sources for Android SDK	20	1	Installed
Android 4.4.2 (API 19)			
SDK Platform	19	4	Installed
Samples for SDK	19	6	Not installed
ARM EABI v7a System Image	19	5	Installed
Intel x86 Atom System Image	19	6	Not installed
Google APIs ARM EABI v7a System Image	19	38	Not installed
Google APIs Intel x86 Atom System Image	19	38	Not installed
Google APIs	19	20	Not installed
Sources for Android SDK	19	2	Not installed
Android 4.3.1 (API 18)			
Android 4.2.2 (API 17)			
Android 4.1.2 (API 16)			
Android 4.0.3 (API 15)			
Android 4.0 (API 14)			
Android 2.3.3 (API 10)			
Android 2.2 (API 8)			
Extras			
Android Support Repository		47	Not installed
Android Support Library		20	Installed

Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates
Sort by: [✓ API level] Repository Deselect All

Done loading packages.

Click on Install packages (number of packages depends on the Android version) to continue with the installation. You will get a dialogue box like this:


Choose Packages to Install

Packages
- Android SDK License
  - Android SDK Build-tools, revision 23.0.1
  - Samples for SDK API 19, revision 6
  - Intel x86 Atom System Image, Android API 1
  - Google APIs ARM EABI v7a System Image, A
  - Google APIs Intel x86 Atom System Image, A
  - Google APIs, Android API 19, revision 20
  - Sources for Android SDK, API 19, revision 2

Package Description & License

Packages
- Android SDK Build-tools, revision 23.0.1
- Samples for SDK API 19, revision 6
- Intel x86 Atom System Image, Android API 19, revision 6
- Google APIs ARM EABI v7a System Image, Android API 19, revision 38
- Google APIs Intel x86 Atom System Image, Android API 19, revision 38
- Google APIs, Android API 19, revision 20
- Sources for Android SDK, API 19, revision 2

License
To get started with the Android SDK, you must agree to the following terms and conditions.

This is the Android SDK License Agreement (the "License Agreement").

1. Introduction

1.1 The Android SDK (referred to in the License Agreement as the "SDK" and

Accept Reject Copy to clipboard | Print [✓ Accept License]

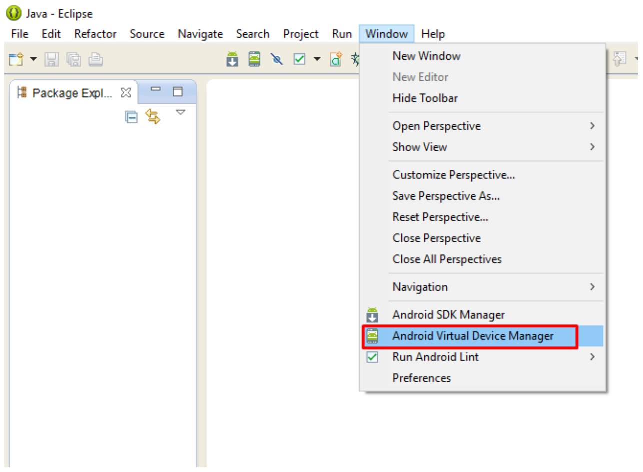
[*] Something depends on this package

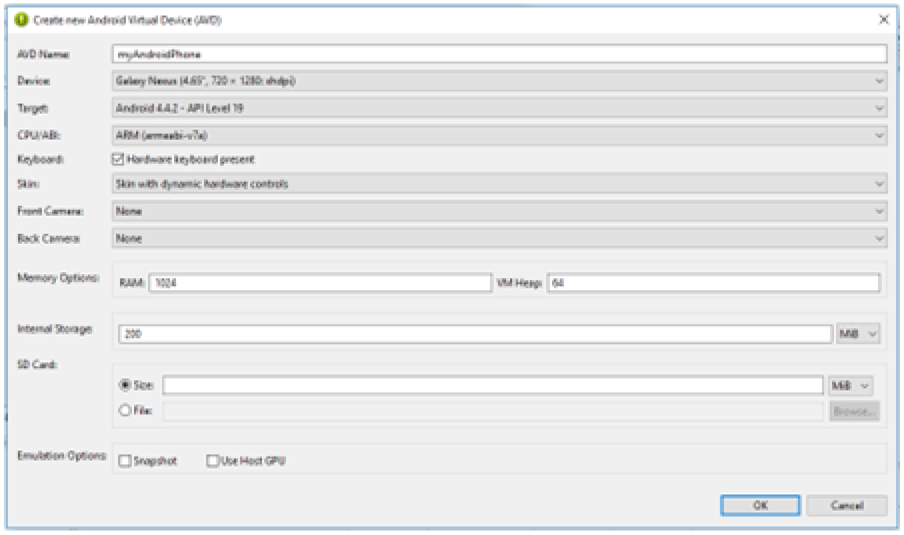
Install Cancel

Step 5: Create Android Virtual Device

The last step is to create Android Virtual Device (AVD), which we will use to test our Android applications.

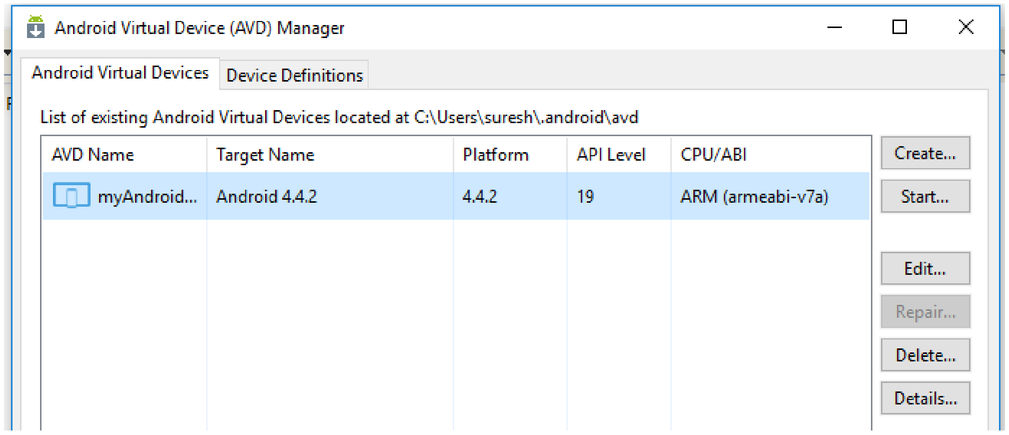
To do this, open Eclipse and Launch Android AVD Manager from options **Window > AVD Manager** and click on New which will create a successful Android Virtual Device and provide the appropriate values to AVD.





Create new Android Virtual Device (AVD)

AVD Name:	myAndroidPhone
Device:	Galaxy Nexus (4.65", 720 × 1280: xhdpi)
Target:	Android 4.4.2 - API Level 19
CPU/ABI:	ARM (armeabi-v7a)
Keyboard:	[✓] Hardware keyboard present
Skin:	Skin with dynamic hardware controls
Front Camera:	None
Back Camera:	None
Memory Options:	RAM: 1024 VM Heap: 64
Internal Storage:	200 MiB
SD Card:	● Size: MiB ○ File:
Emulation Options:	☐ Snapshot ☐ Use Host GPU



Android Virtual Device (AVD) Manager

List of existing Android Virtual Devices located at C:\Users\suresh\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
myAndroid...	Android 4.4.2	4.4.2	19	ARM (armeabi-v7a)

We have successfully created Android Application Development environment using Eclipse IDE and it is ready to create Android App.

3. Aim:- Create Android Virtual Devices (AVDs)

Resources Required: A computer system with Android Studio IDE installed.

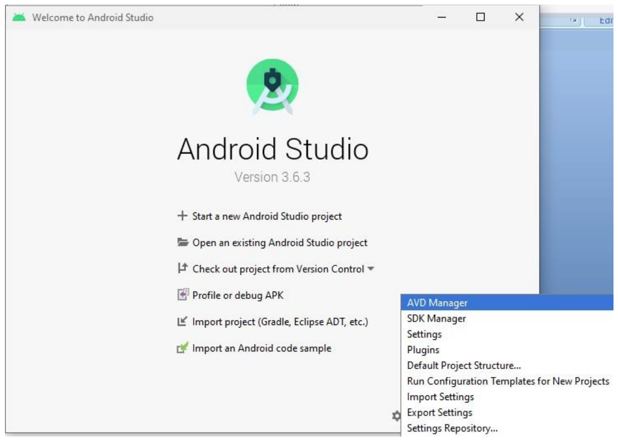
Procedure:

An **Android Virtual Device (AVD)** is a device configuration that runs on the Android Emulator. It provides virtual device-specific Android Environment in which we can install & test our Android Application. AVD Manager is a part of SDK Manager to create and manage the virtual devices created.

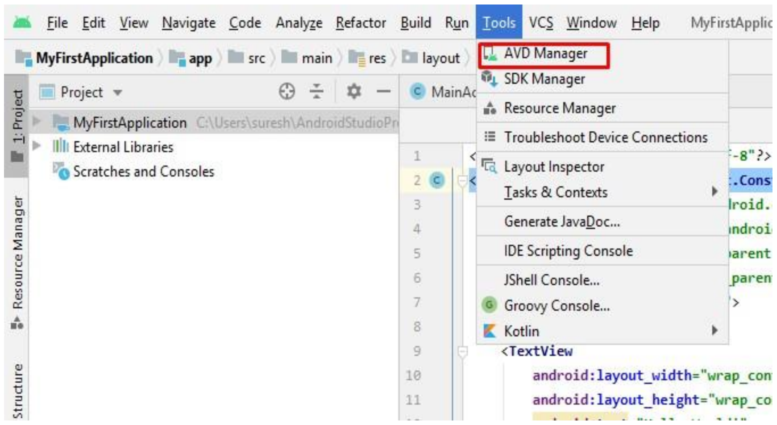
To create an Android Virtual Device (AVD) in Android Studio, follow these steps:

1. Open Android Studio and go to the "Welcome to Android Studio" window.
2. Click on "Configure" and then select "AVD Manager" from the drop-down menu.
3. In the AVD Manager, click the "Create Virtual Device" button.
4. In the "Select Hardware" screen, select a phone device, and then click "Next".
5. In the "System Image" screen, select the version of Android you want to use for the AVD, and then click "Next".
6. In the "AVD Name" screen, enter the name for the AVD. On the "Verify Configuration" screen, you can customize the AVD's hardware and settings. When you're ready, click "Finish".
7. Once the AVD has been created, you should now see the AVD in the AVD Manager window. You can start it by selecting it from the AVD Manager and clicking on the "Start" button. You can also use the AVD Manager to edit the configuration of an AVD or delete it.

To create AVD from the Android Studio Welcome screen, select click on Configure AVD Manager

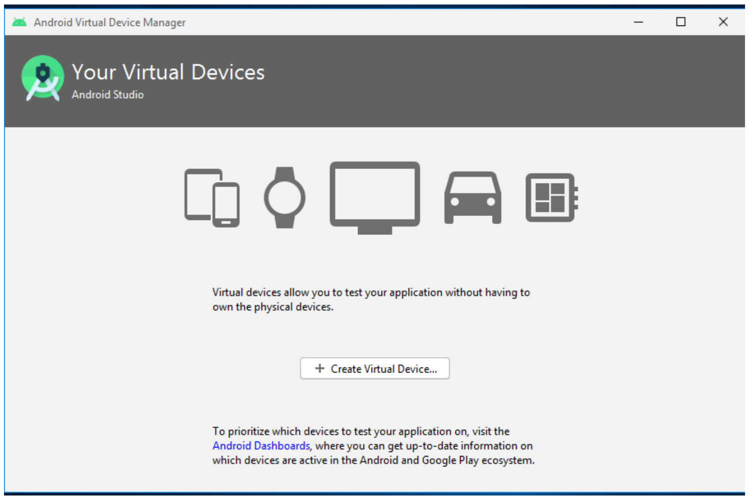

Welcome to Android Studio
Android Studio
Version 3.6.3
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
AVD Manager
SDK Manager
Settings
Plugins
Default Project Structure...
Run Configuration Templates for New Projects
Import Settings
Export Settings
Settings Repository...

To open AVD manager from Android IDE, go to Tools → AVD Manager as shown in below image.



```
<TextView
    android:layout_width="wrap_con
    android:layout_height="wrap_co
```

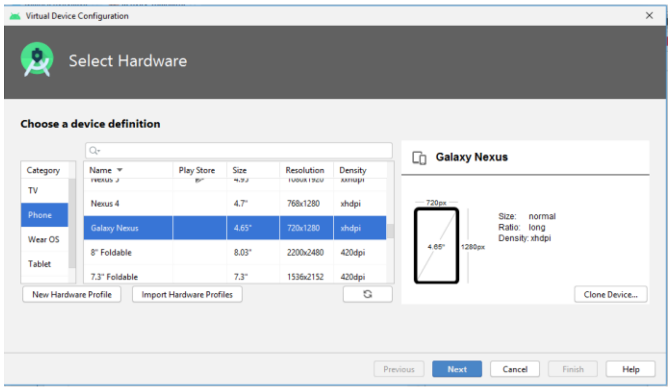
It will open AVD Manager with a list of created virtual devices as shown in below image. It may be empty for you now as you haven't created any device as of now. To create a new device, click on Create Virtual Device button.



Virtual devices allow you to test your application without having to own the physical devices.

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.

It will open a window to Select Hardware type for your virtual device. This list contains almost all the Android devices with their respective settings. Select any one out of all the devices listed, with your required configuration (like Size of the screen, Resolution and Density) and click on Next.



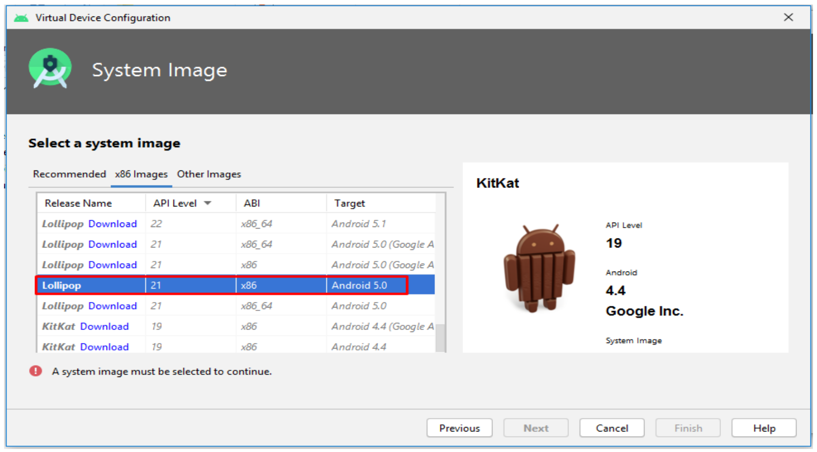
Choose a device definition

Name	Play Store	Size	Resolution	Density
Nexus 5		4.95	1080x1920	xxhdpi
Nexus 4		4.7"	768x1280	xhdpi
Galaxy Nexus		4.65"	720x1280	xhdpi
8" Foldable		8.03"	2200x2480	420dpi
7.3" Foldable		7.3"	1536x2152	420dpi

Galaxy Nexus

Size: normal
Ratio: long
Density: xhdpi

Next you will be asked to select System Image that will be the running Android Version for your newly created virtual device. You can choose any Android system images that are already available in your Android Studio, or Download the one you want, by clicking on the Download option available with the names. Recommended section will list the best choices available as per the latest updates available. X86 Images contain images that are mostly used and Other Images section contains system images with Google Play Services. Choose as per your required configuration (We've selected API level 21). Click on Next once you are done.


Virtual Device Configuration

System Image

Select a system image

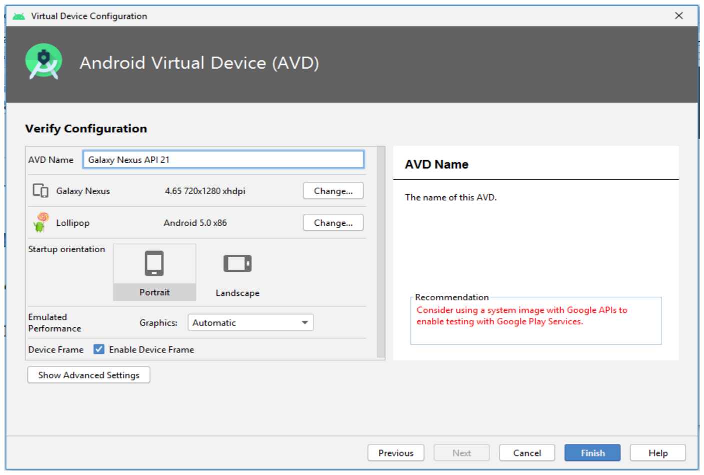
Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
Lollipop Download	22	x86_64	Android 5.1
Lollipop Download	21	x86_64	Android 5.0 (Google A
Lollipop Download	21	x86	Android 5.0 (Google A
Lollipop	21	x86	Android 5.0
Lollipop Download	21	x86_64	Android 5.0
KitKat Download	19	x86	Android 4.4 (Google A
KitKat Download	19	x86	Android 4.4

KitKat
API Level
19
Android
4.4
Google Inc.
System Image

A system image must be selected to continue.

Next window will list down all the configured settings for final verification. Here, you can give your AVD a name for identification, can change device type and API configuration and can also setup orientation as well as Graphics for your AVD.


Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

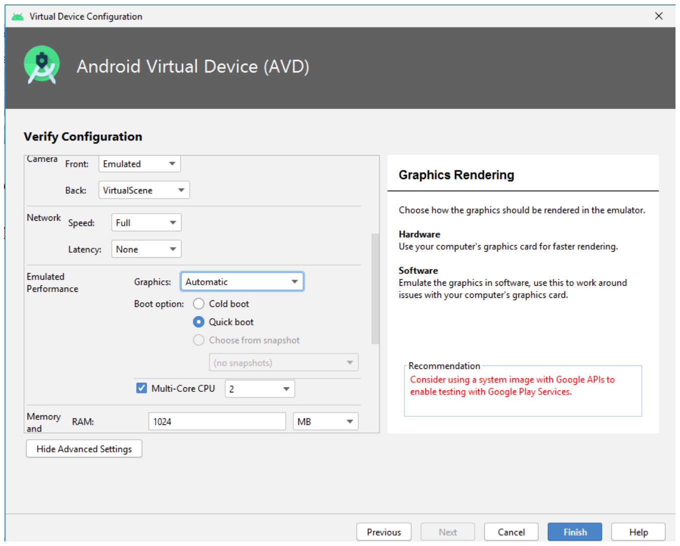
AVD Name: Galaxy Nexus API 21
Galaxy Nexus 4.65 720x1280 xhdpi Change...
Lollipop Android 5.0 x86 Change...
Startup orientation: Portrait | Landscape
Emulated Performance Graphics: Automatic
Device Frame ✓ Enable Device Frame

Show Advanced Settings

AVD Name
The name of this AVD.

Recommendation
Consider using a system image with Google APIs to enable testing with Google Play Services.

Click on Show Advanced Settings and you will see more advanced settings for your virtual device as shown in image below. Here you have settings for Camera, Network, Memory (RAM & Heap) and Storage (Internal & External) and Virtual Device Frame.



Verify Configuration

Camera — Front: Emulated; Back: VirtualScene

Network — Speed: Full; Latency: None

Emulated Performance — Graphics: Automatic; Boot option: Cold boot [✓ Quick boot] Choose from snapshot; (no snapshots); [✓] Multi-Core CPU: 2

Memory and — RAM: 1024 MB

Graphics Rendering

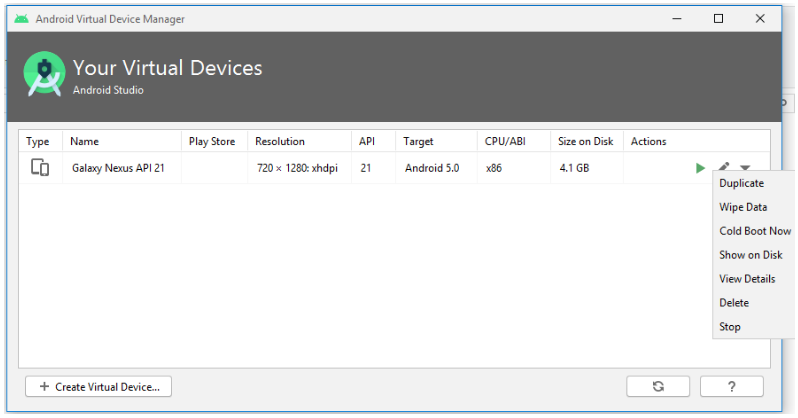
Choose how the graphics should be rendered in the emulator.

Hardware
Use your computer's graphics card for faster rendering.

Software
Emulate the graphics in software, use this to work around issues with your computer's graphics card.

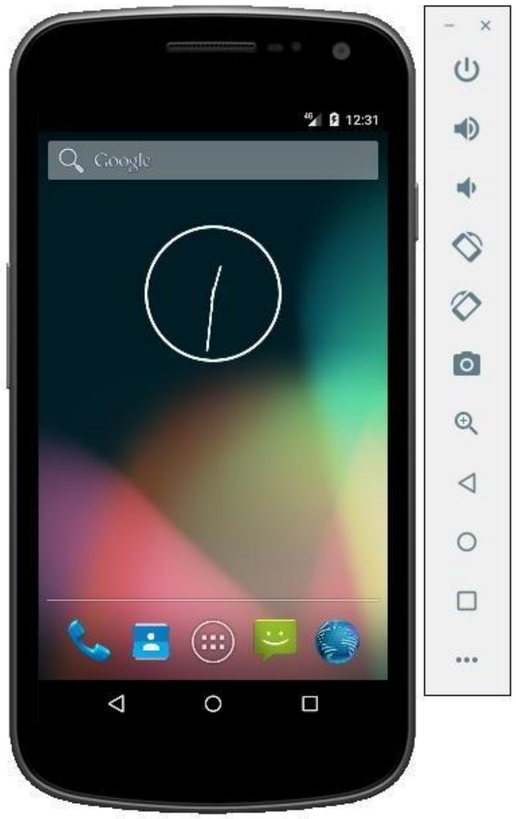
Recommendation
Consider using a system image with Google APIs to enable testing with Google Play Services.

You can configure your device as per your requirements and click on Finish. Android Studio will immediately start building AVD with the selected configurations & might take some time. When it completes, AVD Manager will list out your virtual device in the available devices list as shown in below image.



Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Galaxy Nexus API 21		720 × 1280: xhdpi	21	Android 5.0	x86	4.1 GB	

From the Action column (last column of the table), you can perform several actions like Launch AVD and Edit AVD configurations etc. Launch your first AVD by clicking Start icon (green play icon). It will start a Virtual Device just like an Android Device as shown in below image. Side toolbar contains buttons to perform actions like volume up-down, change orientation, go back, go to home or recent & more. You can also turn the power off for the virtual devices using the power button and to close the virtual device select close button.



Now you have your own Android Virtual Device running on your system where you can test various Android Application Projects. You can have more than one virtual devices in your AVD manager as per your project requirements. Similarly you can also create AVD for devices like Android TV and Android Wear for testing.

4. Aim:- Write about Android project structure

Resources Required: A computer system with Android Studio IDE installed.

Procedure:
An Android project consists of several directories and files that contain the code, resources, and manifest file for your app. If you're working on an Android project, it's important to familiarize yourself with this structure so that you know where to find the files you need and how to properly organize your own files.
The project structure for an Android application is as follows:

1. **app:** This directory contains the code, resources, and manifest file for your app.
2. **manifests:** This directory contains the AndroidManifest.xml file, which is the central configuration file for your app. It specifies the app's package name, the minimum required Android version, the components that are included in the app, and any permissions that the app requires.
3. **src:** This directory contains the source code files for your project, including the main activity class and any other Java classes you've created.
4. **res:** This directory contains all the resources for your project, including layouts, drawable, and strings. It is organized into subdirectories based on the type of resource. For example, the drawable directory contains image files, and the layout directory contains XML files that define the layout of your app's user interface.
   The res directory in an Android project contains resources that are used by the app, such as layouts, drawables, and strings. This directory is organized into several subdirectories, each of which contains a specific type of resource. Here is a list of the common subdirectories found in the res directory:
   - **drawable:** This directory contains image files and other drawable resources.
   - **layout:** This directory contains XML files that define the layout of the user interface for your app.
   - **values:** This directory contains XML files that define various types of values, such as strings, dimensions, and colors.
   - **menu:** This directory contains XML files that define the menus used in your app.
   - **mipmap:** This directory contains the app launcher icons for different densities.
   - **anim:** This directory contains XML files that define property animations.

raw: This directory contains raw files that can be accessed by the app, such as audio or video files.

xml: This directory can contain any XML files that are used by the app.

5. **assets:** This directory can be used to store any files that your app needs to access, but that are not compiled into the APK file.
6. **libs:** This directory contains any third-party libraries that your app depends on.
7. **build:** This directory contains the files that are generated by the Android build system, such as the APK file that is used to install your app on a device.
   build.gradle: This file is used to configure the build for your app. It specifies the dependencies for your app, as well as any custom build options that you need.
8. **proguard-rules.pro:** This file is used to configure ProGuard, which is a tool that is used to shrink and optimize your app's code.

5. Aim:- Develop android application To Display Hello World Message

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as HelloWorld

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app’s components such as activities, services, broadcast receivers, and content providers.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HelloWorld"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>

</manifest>
```

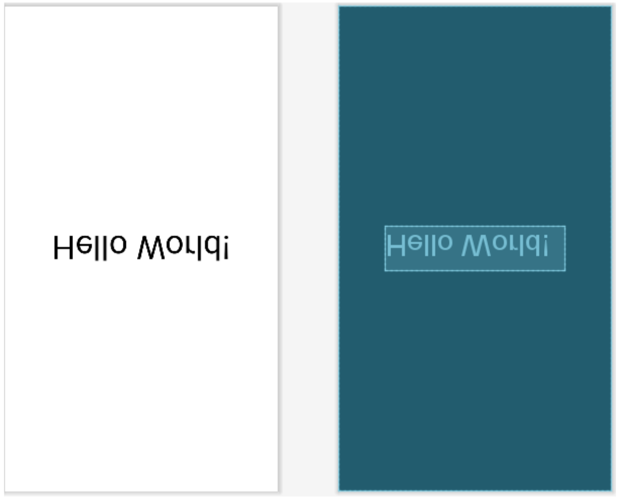
Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.

Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textColor="@color/black"
        android:textSize="50dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Layout Design Diagram:-

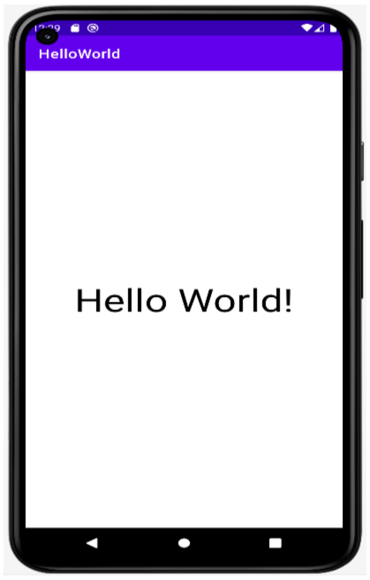


Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.helloworld;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



6. Aim: -Develop android application that will get the Text Entered in Edit Text and display that Text using toast message on clicking a button

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as Toastmessage

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app's components such as activities, services, broadcast receivers, and content providers.

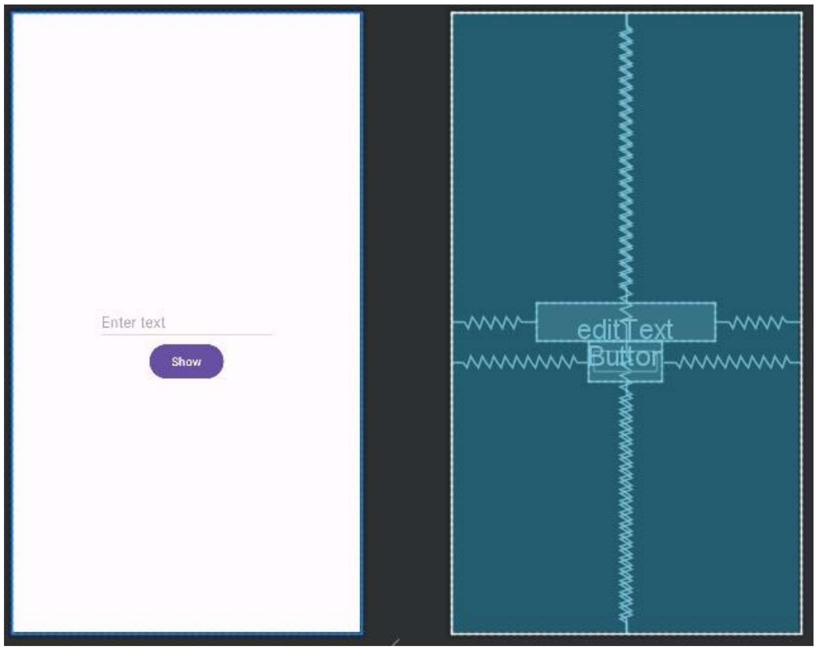
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Shashi"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.
Below is the code for the activity_main.xml file.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:id="@+id/rlVar1"
    >

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Enter text"
        android:ems="10"
        android:inputType="text"
        android:layout_centerInParent="true"/>
    <Button
        android:id="@+id/showBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText"
        android:layout_centerInParent="true"
        android:text="Show"/>

</RelativeLayout>
```

Layout Design Diagram:-

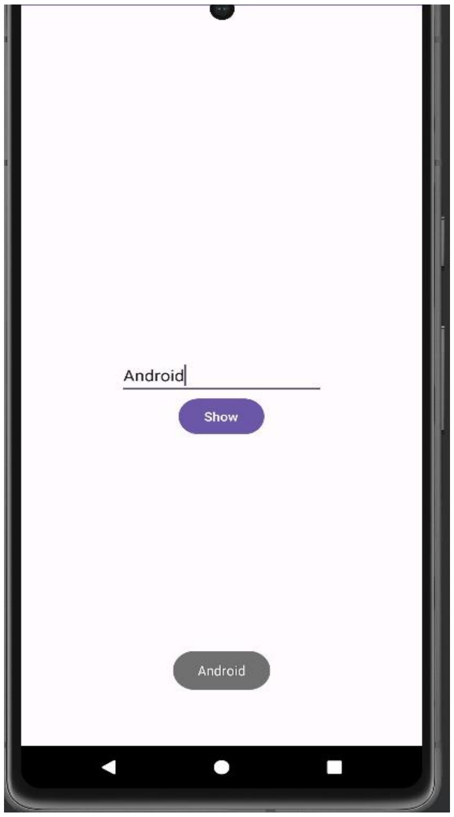


Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.toastmessage;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState)
{super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button showBtn = findViewById(R.id.showBtn);
        EditText editText = findViewById(R.id.editText);

        showBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),
editText.getText().toString(), Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



7. Aim: - Create an Android app to accept two numbers in two Edit Text (text fields) and display the sum of them in a Toast message on clicking a button

Resources Required: A computer system with Android Studio IDE installed.

Procedure:
Step 1: Create a New Project in Android Studio and name it as sum
Step 2: Add Permission to AndroidManifest.xml File
Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app's components such as activities, services, broadcast receivers, and content providers.

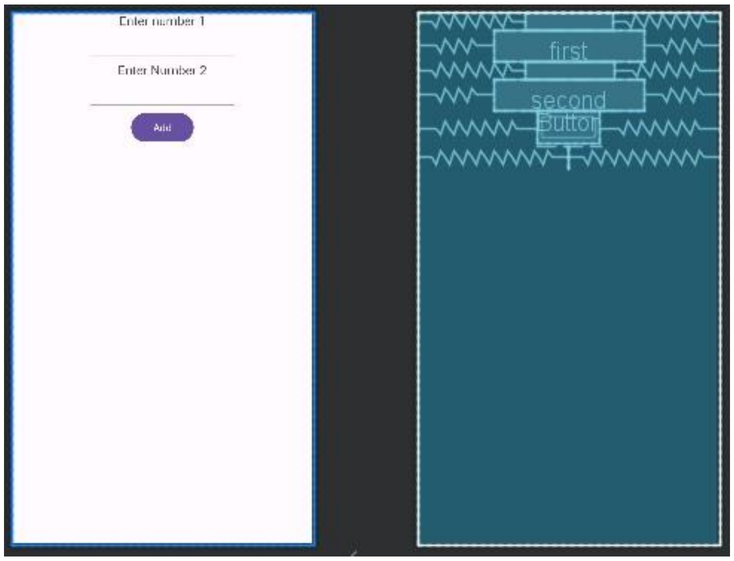
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Linearlayout"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Step 3: Working with the XML Files
Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.
Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerHorizontal="true"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textview1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Enter number 1"
        android:textSize="18sp" />

    <EditText
        android:id="@+id/first"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textview1"
        android:layout_alignParentTop="false"
        android:layout_alignParentRight="false"
        android:layout_centerHorizontal="true"
        android:ems="10"
        android:inputType="number" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/first"
        android:layout_centerHorizontal="true"
        android:text="Enter Number 2"
        android:textSize="18sp" />

    <EditText
        android:id="@+id/second"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textView2"
        android:layout_centerHorizontal="true"
        android:ems="10"
        android:inputType="number" />

    <Button
        android:id="@+id/buttonadd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/second"
        android:layout_centerHorizontal="true"
        android:text="Add" />

    <TextView
```

```
        android:id="@+id/result"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/buttonadd"
        android:layout_centerHorizontal="true"
        android:textSize="24sp" />

</RelativeLayout>
```

Layout Design Diagram:-

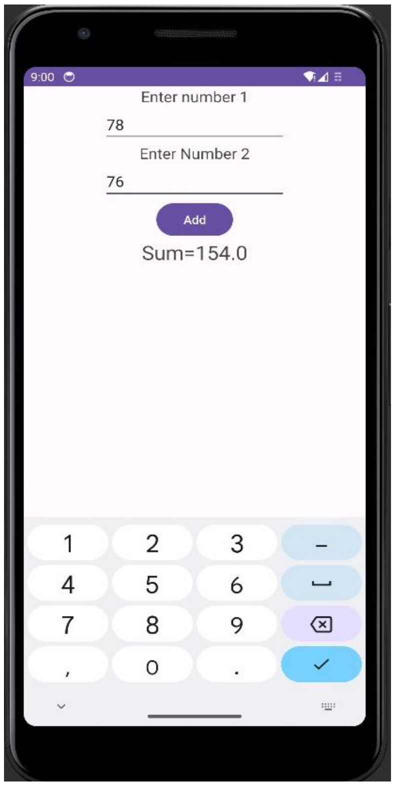


Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.sum;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    EditText firstnum,secondnum;
    TextView r;
    Button bt;
    double a,b,c;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        firstnum=(EditText) findViewById(R.id.first);
        secondnum=(EditText) findViewById(R.id.second);
        bt=(Button) findViewById(R.id.buttonadd);
        r=(TextView) findViewById(R.id.result);
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                a=Double.parseDouble(firstnum.getText().toString());
                b=Double.parseDouble(secondnum.getText().toString());
                c=a+b;
                r.setText("Sum="+c);
            }
        });
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



8. Aim:- Create an Android app to accept a number in EditText and display the factorial of it in a Toast message on clicking a button.

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as factorial

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app's components such as activities, services, broadcast receivers, and content providers.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Linearlayout"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>
</manifest>
```

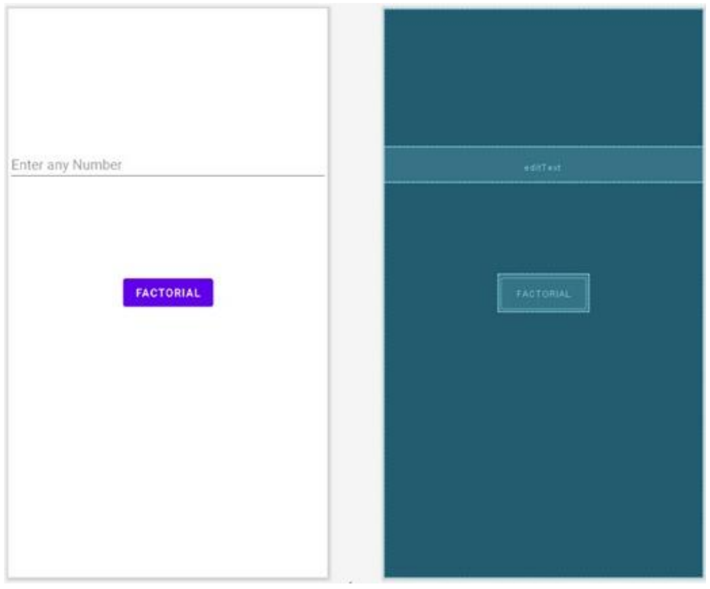
Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.

Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:hint="Enter any Number"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.259" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Factorial"
        android:id="@+id/btnFact"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText"
        app:layout_constraintVertical_bias="0.256"></Button>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Layout Design Diagram:-

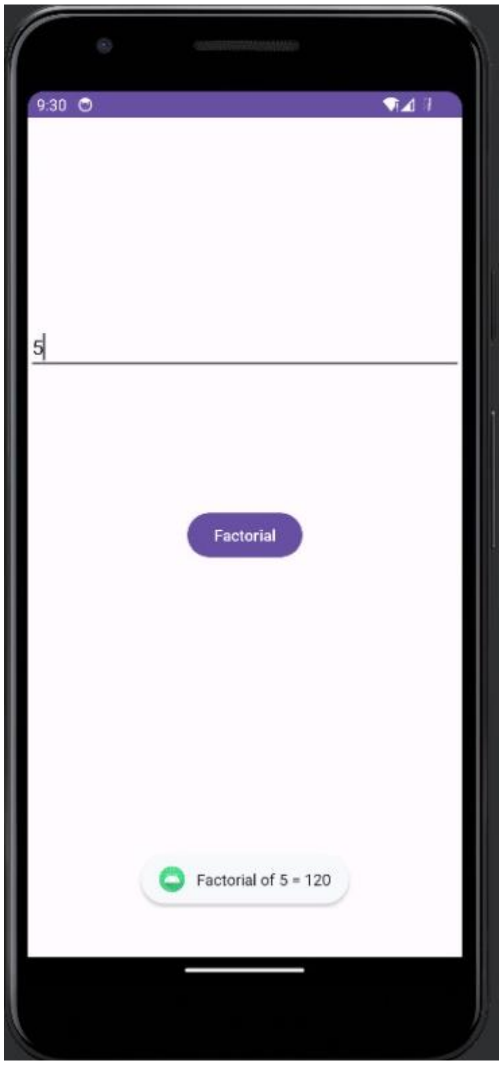


Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.factorial;
import androidx.appcompat.app.AppCompatActivity;
import android.annotation.SuppressLint;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    private EditText editText;
    private Button btnfact;
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText=findViewById(R.id.editText);
        btnfact=findViewById(R.id.btnFact);
        btnfact.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(TextUtils.isEmpty(editText.getText().toString()))
                {
                    Toast.makeText(MainActivity.this, "Empty Credential",
Toast.LENGTH_SHORT).show();
                }
                else
                {
                    int a=Integer.parseInt(editText.getText().toString());
                    int fact=1;

                    for(int i=1;i<=a;i++)
                    {
                        fact=fact*i;
                    }
                    Toast.makeText(MainActivity.this, "Factorial of "+a+" = "+fact,
Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



Factorial of 5 = 120

9. Aim:- Design a simple calculator application to perform addition, subtraction, multiplication and division using different buttons.

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as Calculator

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app’s components such as activities, services, broadcast receivers, and content providers.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication1"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>
</manifest>
```

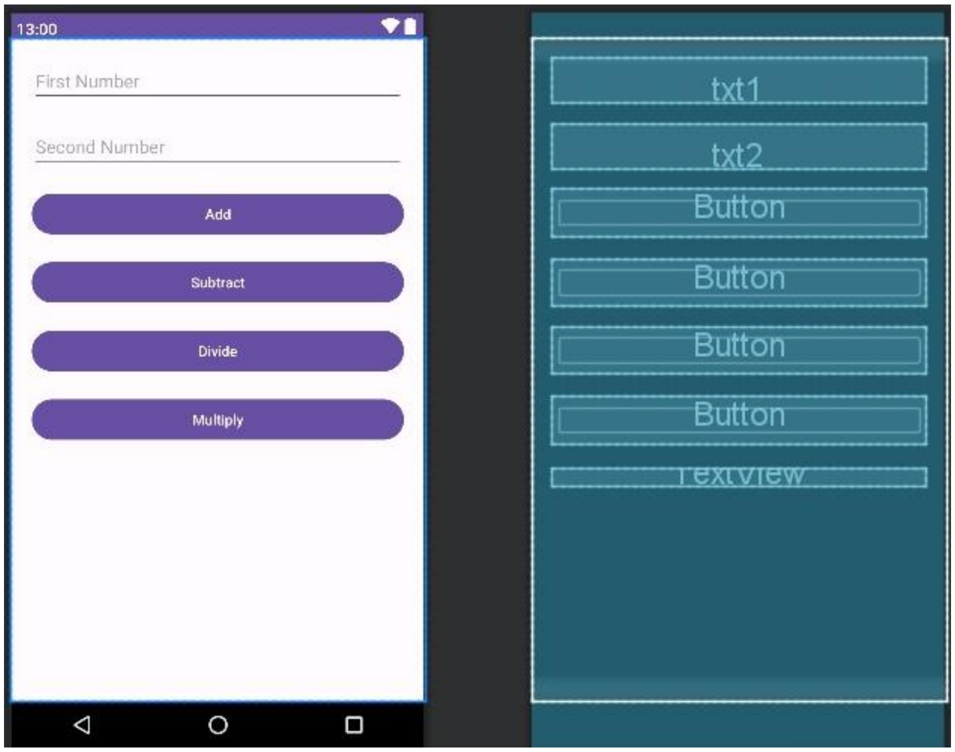
Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.

Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/txt1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="20dp"
        android:ems="10"
        android:hint="First Number"
        android:inputType="numberDecimal"
        tools:ignore="TouchTargetSizeCheck,SpeakableTextPresentCheck" />
    <EditText
        android:id="@+id/txt2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="20dp"
        android:ems="10"
        android:hint="Second Number"
        android:inputType="numberDecimal"
        tools:ignore="SpeakableTextPresentCheck,TouchTargetSizeCheck" />
    <Button
        android:id="@+id/btnadd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="20dp"
        android:text="Add" /
    <Button
        android:id="@+id/btnsubs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="20dp"
        android:text="Subtract" />
    <Button
        android:id="@+id/btndiv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="20dp"
        android:text="Divide" />
    <Button
        android:id="@+id/btnmult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="20dp"
        android:text="Multiply" />
    <TextView
        android:id="@+id/result"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="25dp"
        android:layout_marginRight="20dp"
        />
</LinearLayout>
```

Layout Design Diagram:-

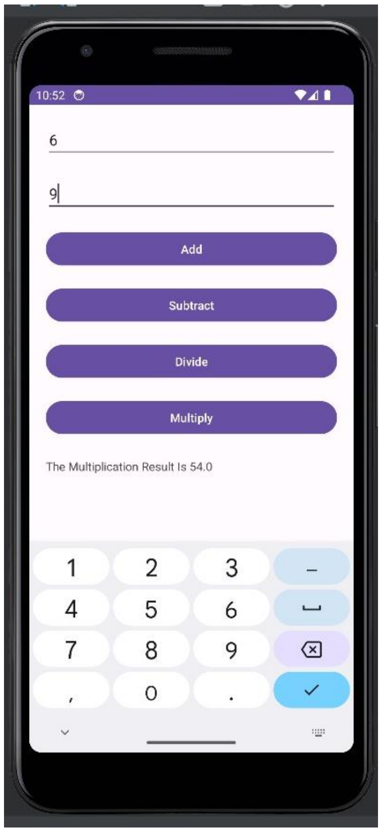


Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.calculator;
import androidx.appcompat.app.AppCompatActivity;
import android.annotation.SuppressLint;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
public class MainActivity extends AppCompatActivity {
    Button btnadd,btnsubs,btnmult,btndiv;
    EditText txt1,txt2;
    TextView result;
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnadd=findViewById(R.id.btnadd);
        btnsubs=findViewById(R.id.btnsubs);
        btndiv=findViewById(R.id.btndiv);
        btnmult=findViewById(R.id.btnmult);
        txt1=findViewById(R.id.txt1);
        txt2=findViewById(R.id.txt2);
        result=findViewById(R.id.result);
        btnadd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Checking Input First Is Blank Or Not
                if (txt1.getText().toString().equals("")) {
                    // Showing Toast (Message)
                    Toast.makeText(MainActivity.this,   "Please   Enter   Number",
Toast.LENGTH_SHORT).show();
                } else if (txt2.getText().toString().equals("")) {
                    Toast.makeText(MainActivity.this,   "Please   Enter   Number",
Toast.LENGTH_SHORT).show();
                }
                // Both Inputs Are Not Blank , Starting Calculation
                else {
                    float a, b, c;
                    a = Float.parseFloat(txt1.getText().toString());
                    b = Float.parseFloat(txt2.getText().toString());
                    c = a + b; // Using Third Variable To Store Output Value
                    result.setText("The Addition Result Is " + c);
                }
            }
        });
        btnsubs.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Checking Input First Is Blank Or Not
                if (txt1.getText().toString().equals("")) {
                    // Showing Toast (Message)
                    Toast.makeText(MainActivity.this,   "Please   Enter   Number",
Toast.LENGTH_SHORT).show();
                } else if (txt2.getText().toString().equals("")) {
                    Toast.makeText(MainActivity.this,   "Please   Enter   Number",
Toast.LENGTH_SHORT).show();
                }
                // Both Inputs Are Not Blank , Starting Calculation
                else {
                    float a, b, c;
                    a = Float.parseFloat(txt1.getText().toString());
                    b = Float.parseFloat(txt2.getText().toString());
                    c = a - b; // Using Third Variable To Store Output Value
                    result.setText("The Subtraction Result Is " + c);
                }
            }
        });
        btnmult.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Checking Input First Is Blank Or Not
                if (txt1.getText().toString().equals("")) {
                    // Showing Toast (Message)
                    Toast.makeText(MainActivity.this,   "Please   Enter   Number",
Toast.LENGTH_SHORT).show();
                } else if (txt2.getText().toString().equals("")) {
                    Toast.makeText(MainActivity.this,   "Please   Enter   Number",
Toast.LENGTH_SHORT).show();
                }
                // Both Inputs Are Not Blank , Starting Calculation
                else {
                    float a, b, c;
                    a = Float.parseFloat(txt1.getText().toString());
                    b = Float.parseFloat(txt2.getText().toString());
                    c = a*b; // Using Third Variable To Store Output Value
                    result.setText("The Multiplication Result Is " + c);
                }
            }
        });
        btndiv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Checking Input First Is Blank Or Not
                if (txt1.getText().toString().equals("")) {
                    // Showing Toast (Message)
                    Toast.makeText(MainActivity.this,    "Please    Enter    Number",
Toast.LENGTH_SHORT).show();
                } else if (txt2.getText().toString().equals("")) {
                    Toast.makeText(MainActivity.this,    "Please    Enter    Number",
Toast.LENGTH_SHORT).show();
                }
                // Both Inputs Are Not Blank , Starting Calculation
                else {
                    float a, b, c;
                    a = Float.parseFloat(txt1.getText().toString());
                    b = Float.parseFloat(txt2.getText().toString());
                    c = a/b; // Using Third Variable To Store Output Value
                    result.setText("The Division Result Is " + c);
                }
            }
        });
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



The Multiplication Result Is 54.0

10. Aim:- Develop android Application using LinearLayout

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as LinearLayout Example

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app's components such as activities, services, broadcast receivers, and content providers.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.LinearLayoutExample"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>

</manifest>
```

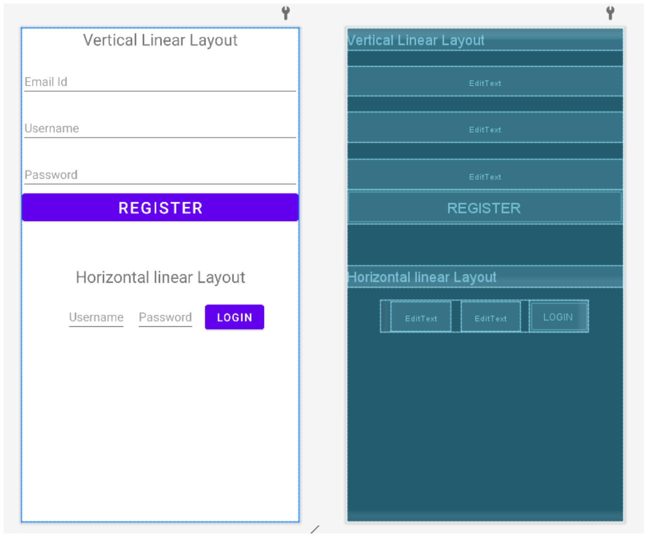
Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.

Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="Vertical Linear Layout"
        android:textSize="24sp"
        />
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:hint="Email Id"/>
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:hint="Username"/>
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:hint="Password"/>
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:text="register"
            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:text="Horizontal linear Layout"
            android:layout_marginTop="60dp"
            android:textSize="24sp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="20dp"
        android:layout_gravity="center_horizontal">

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:hint="Username"/>
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
           android:layout_marginStart="15dp"
            android:hint="Password"/>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:text="Login"
            android:textSize="16sp" />
    </LinearLayout>
</LinearLayout>
```

Layout Design Diagram:-

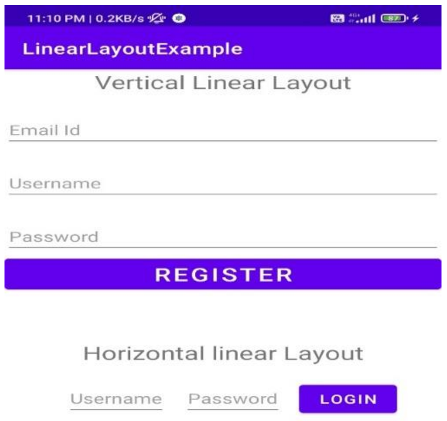


Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.linearlayoutexample;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



11. Aim:- Develop android application using Relative Layout

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as Relativelayout

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app's components such as activities, services, broadcast receivers, and content providers.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Linearlayout"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>
</manifest>
```

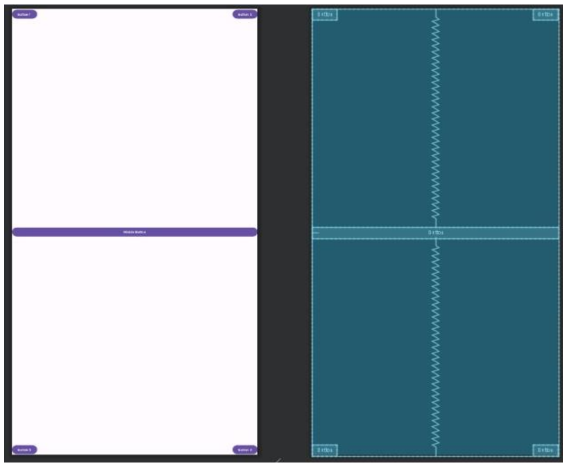
Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.

Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="button 1"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"/>
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="button 2"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"/>
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="button 3"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"/>
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="button 4"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"/>
    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Middle Button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

Layout Design Diagram:-

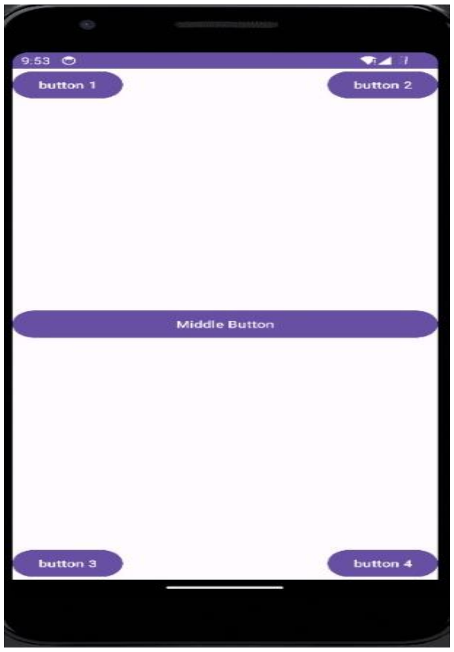


Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.realativelayout;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



12. Aim:- Develop android application using Table Layout

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as Tablelayout

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app's components such as activities, services, broadcast receivers, and content providers.

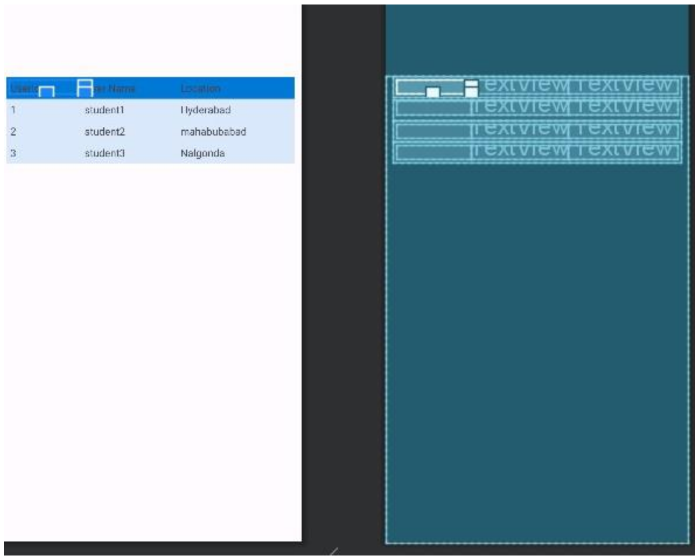
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication1"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.
Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="100dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp" >
    <TableRow android:background="#0079D6" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="UserId" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="User Name" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Location" />
    </TableRow>
    <TableRow android:background="#DAE8FC" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="1" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="student1" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Hyderabad" />
    </TableRow>
    <TableRow android:background="#DAE8FC" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="2" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="student2" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="mahabubabad" />
    </TableRow>
    <TableRow android:background="#DAE8FC" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="3" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="student3" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Nalgonda" />
    </TableRow>
</TableLayout>
```

Layout Design Diagram:-



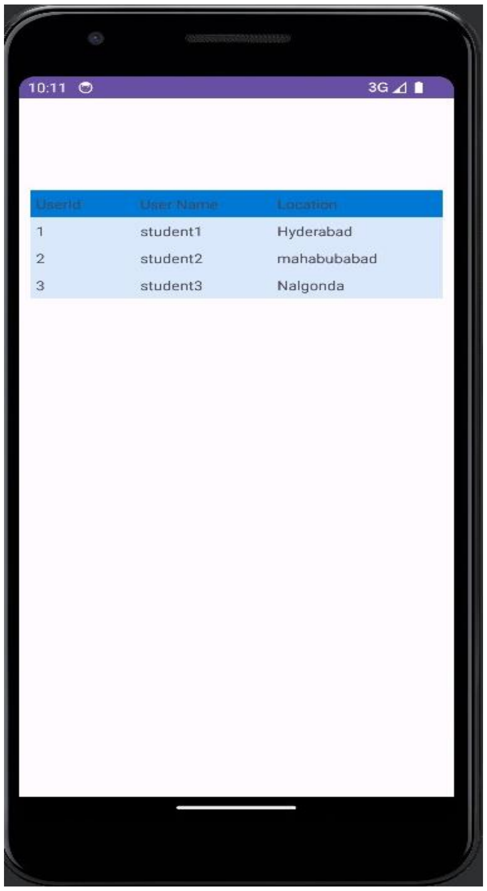
[illegible]	[illegible]	Location
1	student1	Hyderabad
2	student2	mahabubabad
3	student3	Nalgonda

Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.tablelayout;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



UserId	User Name	Location
1	student1	Hyderabad
2	student2	mahabubabad
3	student3	Nalgonda

13. Aim:- Develop android application using Constraint Layout

Resources Required: A computer system with Android Studio IDE installed.

Procedure:

Step 1: Create a New Project in Android Studio and name it as Constraintlayout

Step 2: Add Permission to AndroidManifest.xml File

Go to the app - > manifests -> AndroidManifest.xml file, which represents the complete information about the app s package name, version, and required permissions, as well as the app’s components such as activities, services, broadcast receivers, and content providers.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication1"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>
</manifest>
```

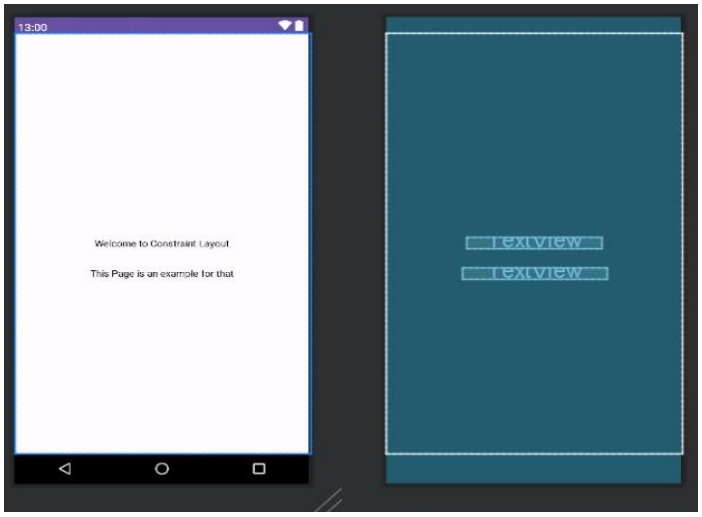
Step 3: Working with the XML Files

Next, go to the app -> layout -> activity_main.xml file, which represents the UI of the project.

Below is the code for the activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to Constraint Layout"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:textColor="@android:color/black"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This Page is an example for that"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintVertical_bias="0.091"
        android:textColor="@android:color/black"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Layout Design Diagram:-

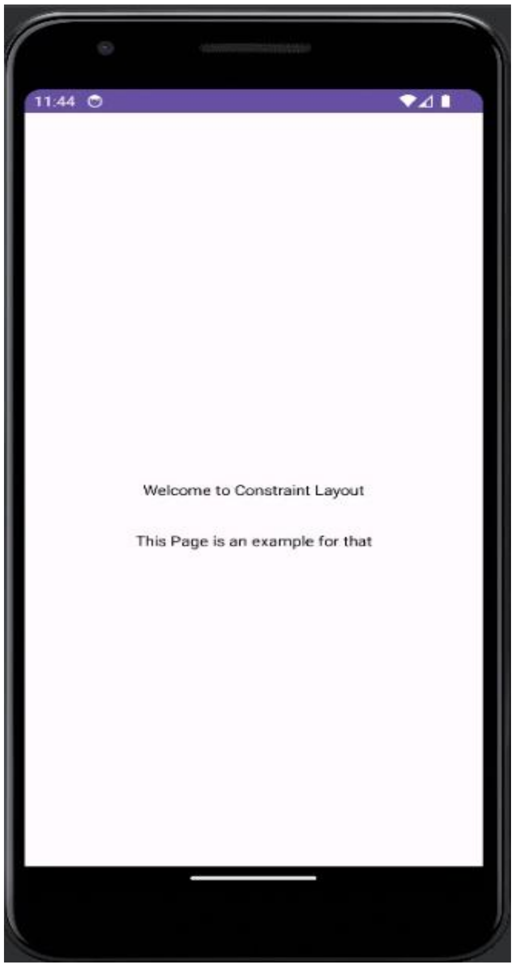

13:00
Welcome to Constraint Layout
This Page is an example for that
TextView
TextView

Step 4: Working with the MainActivity File

Go to the MainActivity File and refer to the following code. Below is the code for the MainActivity File.

```
package com.example.constraintlayput;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Step 5: Running the application to show the Output in Emulator/MobilePhone



Welcome to Constraint Layout

This Page is an example for that