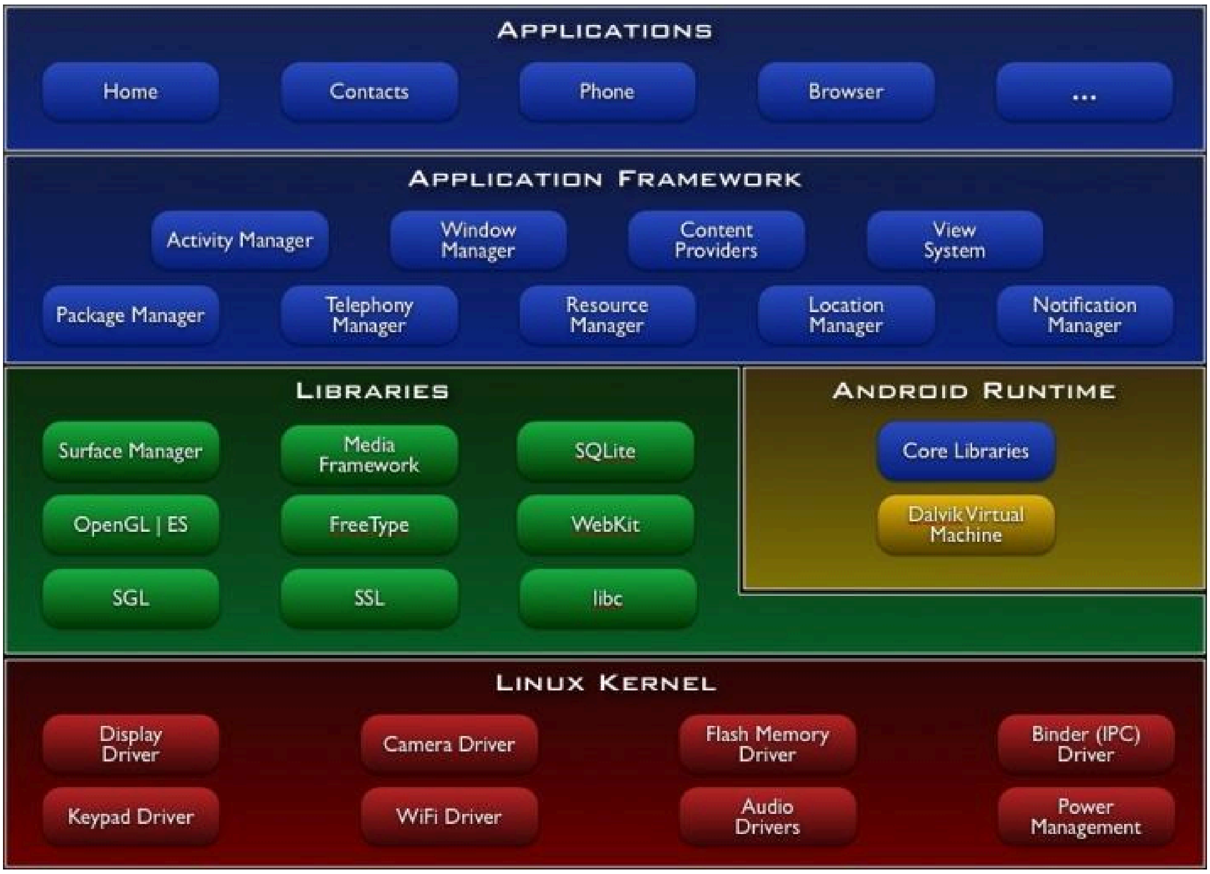
Android:

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touch screen mobile devices such as smart phones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google. Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The current stable version is Android 10, released on September 3, 2019.

Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



Linux kernel

At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

Libraries

On top of Linux kernel there is a set of libraries including open -source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.

Android UI

An Android application user interface is everything that the user can see and interact with.

Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Project Structure

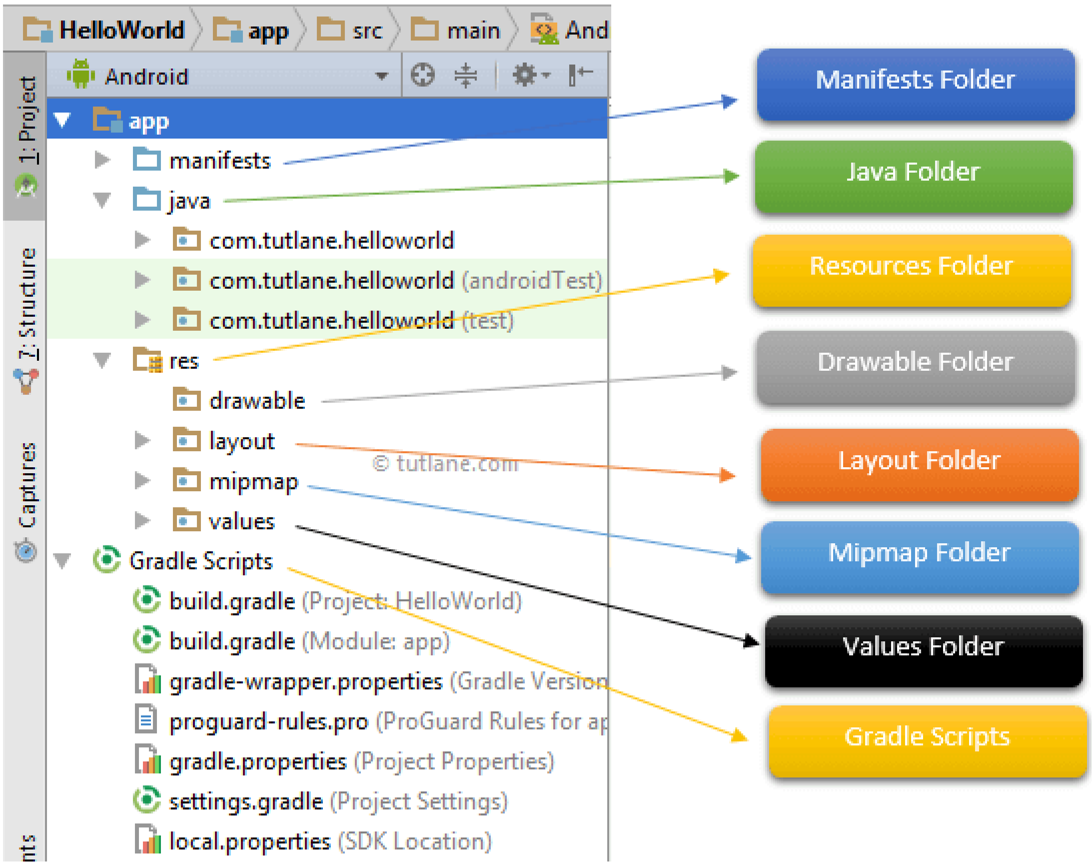


Figure 1. The project files in Android view

Each project in Android Studio contains one or more modules with source code files and resource files.

Types of modules include:

- **Android app modules**
- **Library modules**
- **Google App Engine modules**

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- **manifests:** Contains the AndroidManifest.xml file.
- **java:** Contains the Java source code files, including JUnit test code.
- **res:** Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

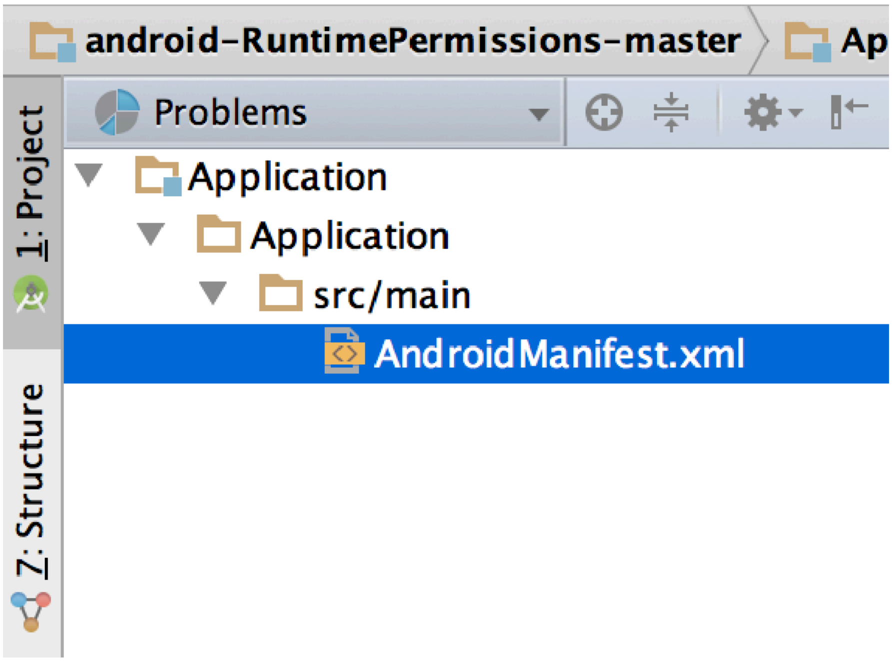


Figure 2. The project files in Problems view, showing a layout file with a problem.

The user interface

The Android Studio main window is made up of several logical areas identified in figure 3.

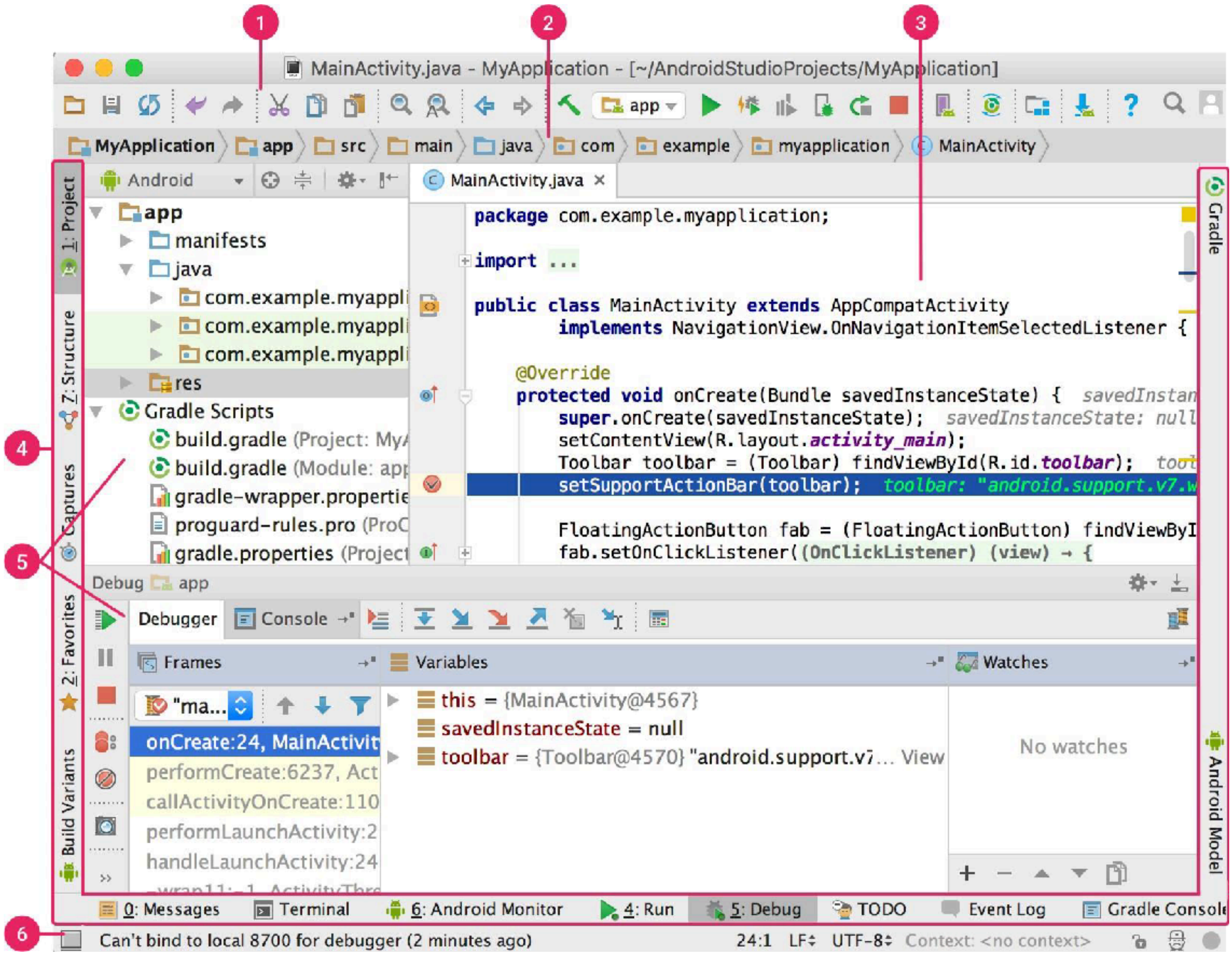


Figure 3. The Android Studio main window.

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Tool windows

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

• To expand or collapse a tool window, click the tool's name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.

• To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.

• To show or hide the entire tool window bar, click the window icon in the bottom left-hand corner of the Android Studio window.

• To locate a specific tool window, hover over the window icon and select the tool window from the menu.

You can also use keyboard shortcuts to open tool windows. Table 1 lists the shortcuts for the most common windows.

Table 1. Keyboard shortcuts for some useful tool windows.

Tool window	**Windows and Linux**	**Mac**
Project	Alt+1	Command+1
Version Control	Alt+9	Command+9
Run	Shift+F10	Control+R
Debug	Shift+F9	Control+D
Logcat	Alt+6	Command+6
Return to Editor	Esc	Esc
Hide All Tool Windows	Control+Shift+F12	Command+Shift+F12

If you want to hide all toolbars, tool windows, and editor tabs, click **View > Enter Distraction Free Mode**. This enables *Distraction Free Mode*. To exit Distraction Free Mode, click **View > Exit Distraction Free Mode**.

You can use *Speed Search* to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

Code completion

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

Table 2. Keyboard shortcuts for code completion.

Type	**Description**	**Windows and Linux**	**Mac**
Basic Completion	Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static	Control+ Space	Control+ Space

	members.		
Smart Completion	Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice in a row, you see more results, including chains.	Control+Shift+Space	Control+Shift+Space
Statement Completion	Completes the current statement for you, adding missing parentheses, brackets,	Control+Shift+Enter	Shift+Command+Enter

	braces, formattin g, etc.		

Android Studio Installation

To install Android Studio on Windows, proceed as follows:

1. If you downloaded an .exe file (recommended), double-click to launch it.

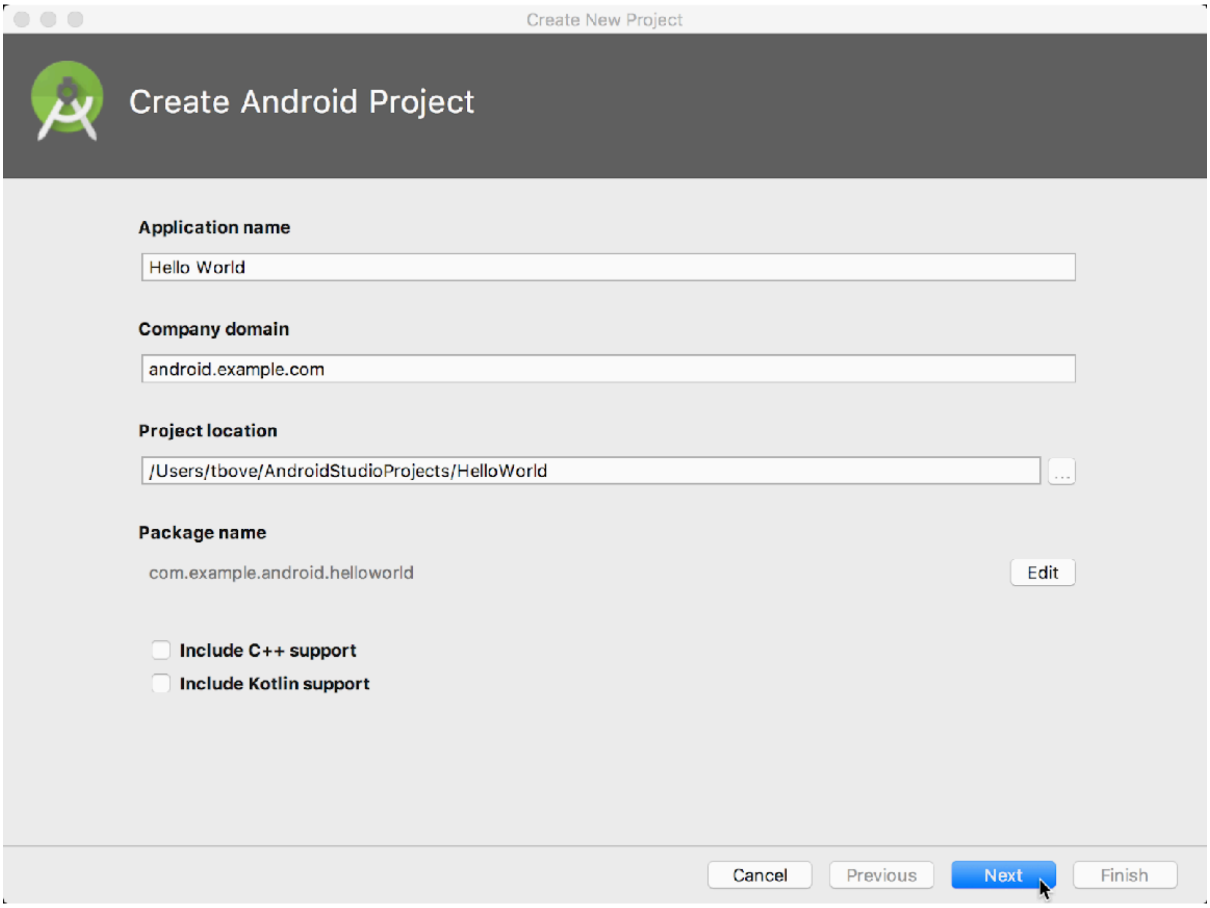
If you downloaded a .zip file, unpack the ZIP, copy the **android-studio** folder into your **Program Files** folder, and then open the **android-studio > bin** folder and launch studio64.exe (for 64-bit machines) or studio.exe (for 32-bit machines).

2. Follow the setup wizard in Android Studio and install any SDK packages that it recommends.

Create "Hello World" application

After you successfully install Android Studio, you will create, from a template, a new project for the Hello World app. This simple app displays the string "Hello World" on the screen of the Android virtual or physical device.

1. **Studio** window, click **Start a new Android Studio project**.
2. In the **Create Android Project** window, enter **Hello World** for the **Application name**.



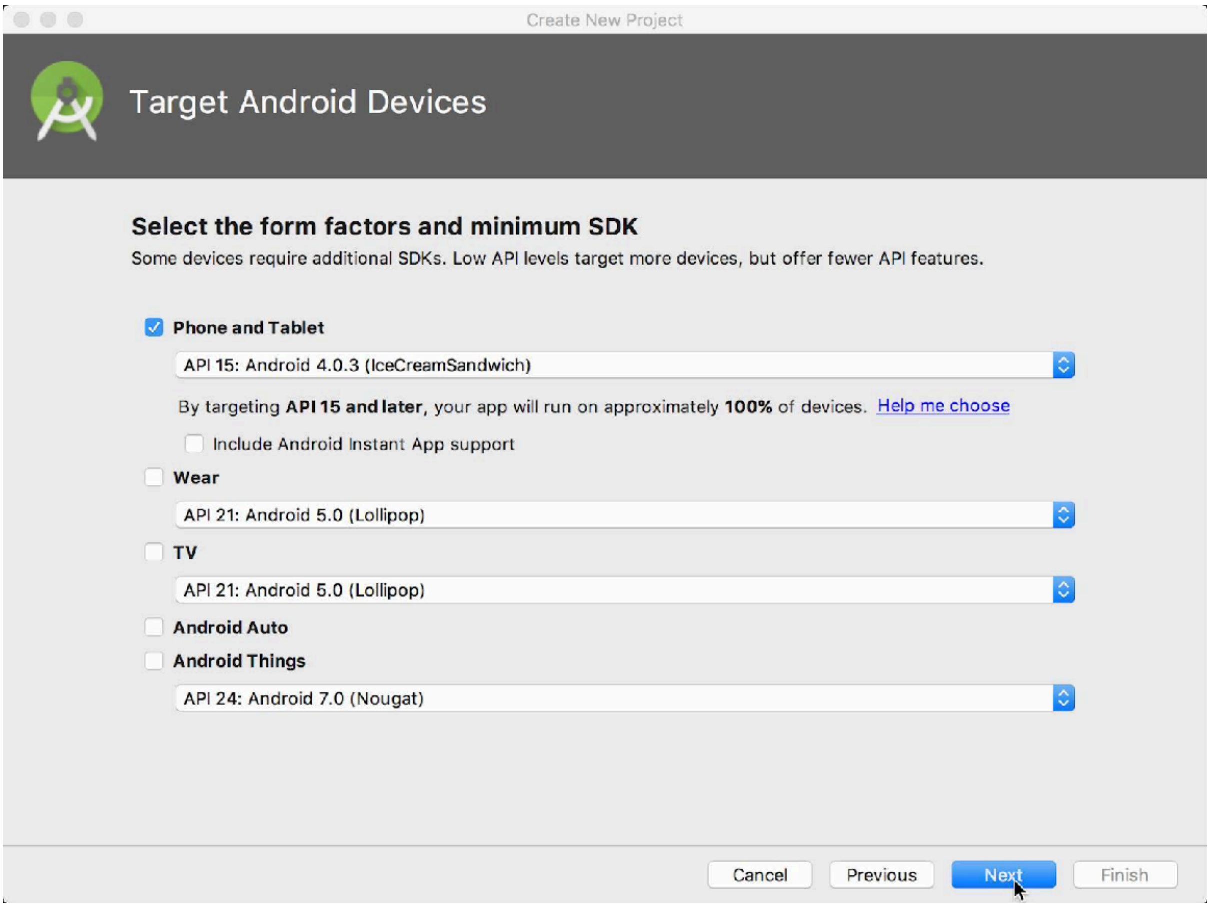
3. Verify that the default **Project location** is where you want to store your Hello World app and other Android Studio projects, or change it to your preferred directory.

4. Accept the default **android.example.com** for **Company Domain**, or create a unique company domain.

If you are not planning to publish your app, you can accept the default. Be aware that changing the package name of your app later is extra work.

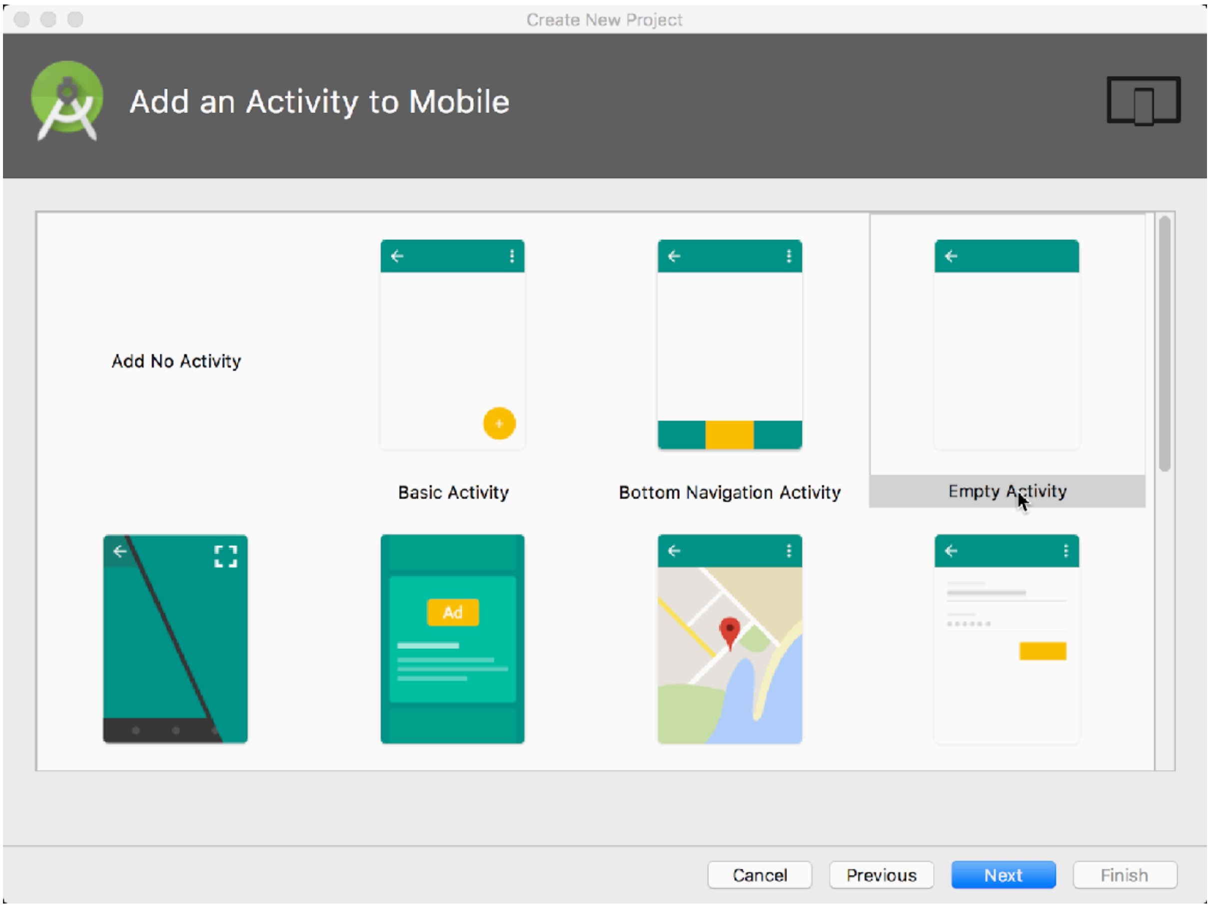
5. Leave unchecked the options to **Include C++ support** and **Include Kotlin support**, and click **Next**.

6. On the **Target Android Devices** screen, **Phone and Tablet** should be selected. Ensure that **API 15: Android 4.0.3 IceCreamSandwich** is set as the Minimum SDK; if it is not, use the popup menu to set it.

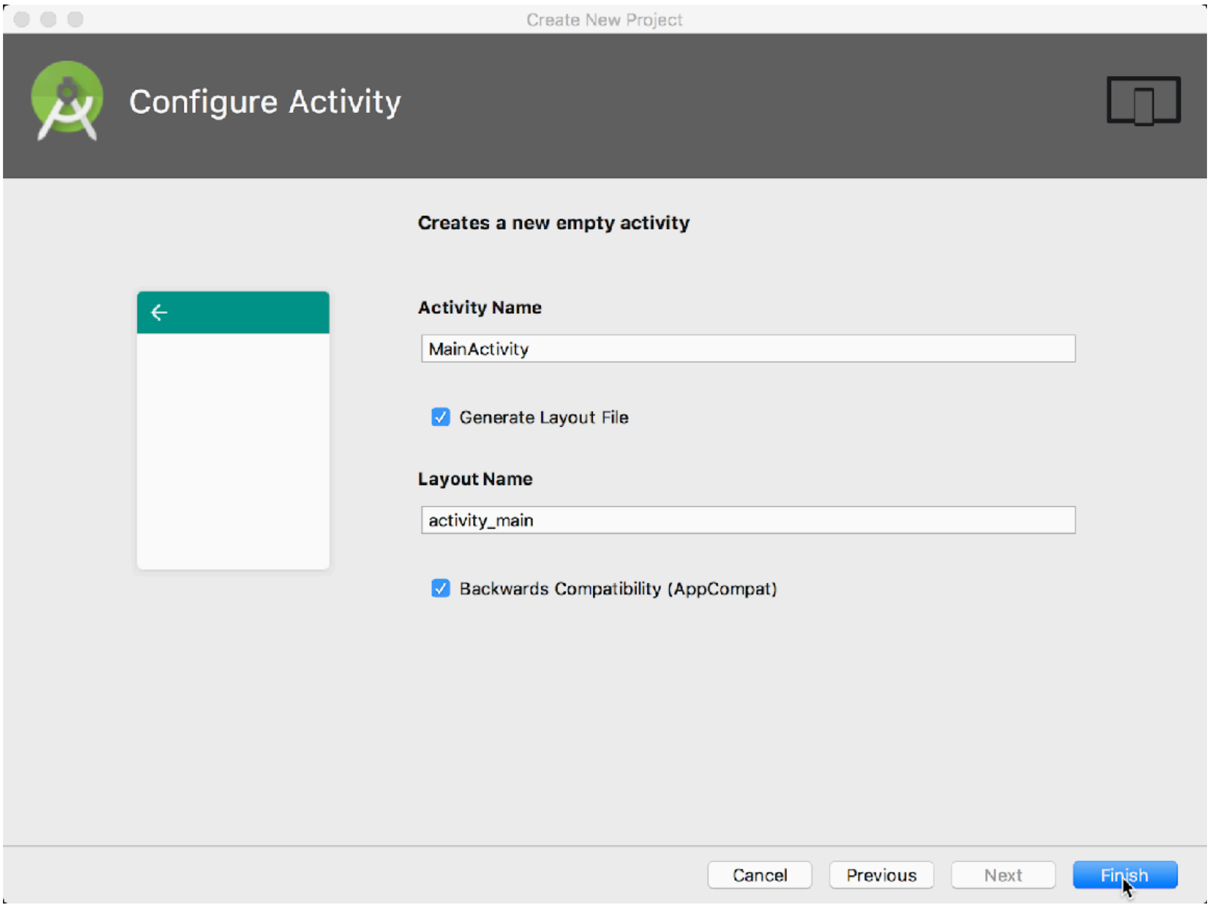


7. Leave unchecked the **Include Instant App support** and all other options. Then click **next**. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically.

8. The **Add an Activity** window appears. An Activity is a single, focused thing that the user can do. It is a crucial component of any Android app. An Activity typically has a layout associated with it that defines how UI elements appear on a screen. Android Studio provides Activity templates to help you get started. For the Hello World project, choose **Empty Activity** as shown below, and click **next**.



9. The **Configure Activity** screen appears (which differs depending on which template you chose in the previous step). By default, the empty Activity provided by the template is named MainActivity. You can change this if you want, but this lesson uses MainActivity.



10. Make sure that the **Generate Layout file** option is checked. The layout name by default is activity_main. You can change this if you want, but this lesson uses activity_main.

11. Make sure that the **Backwards Compatibility (App Compat)** option is checked. This ensures that your app will be backwards-compatible with previous versions of Android.

12. Click **Finish**.

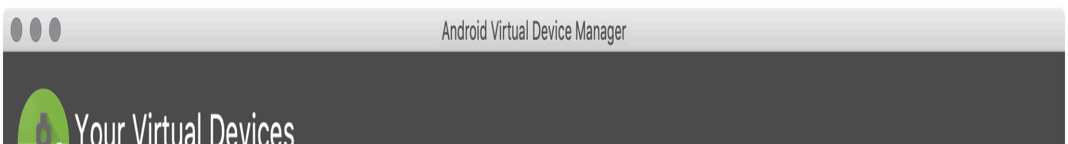
Create an AVD

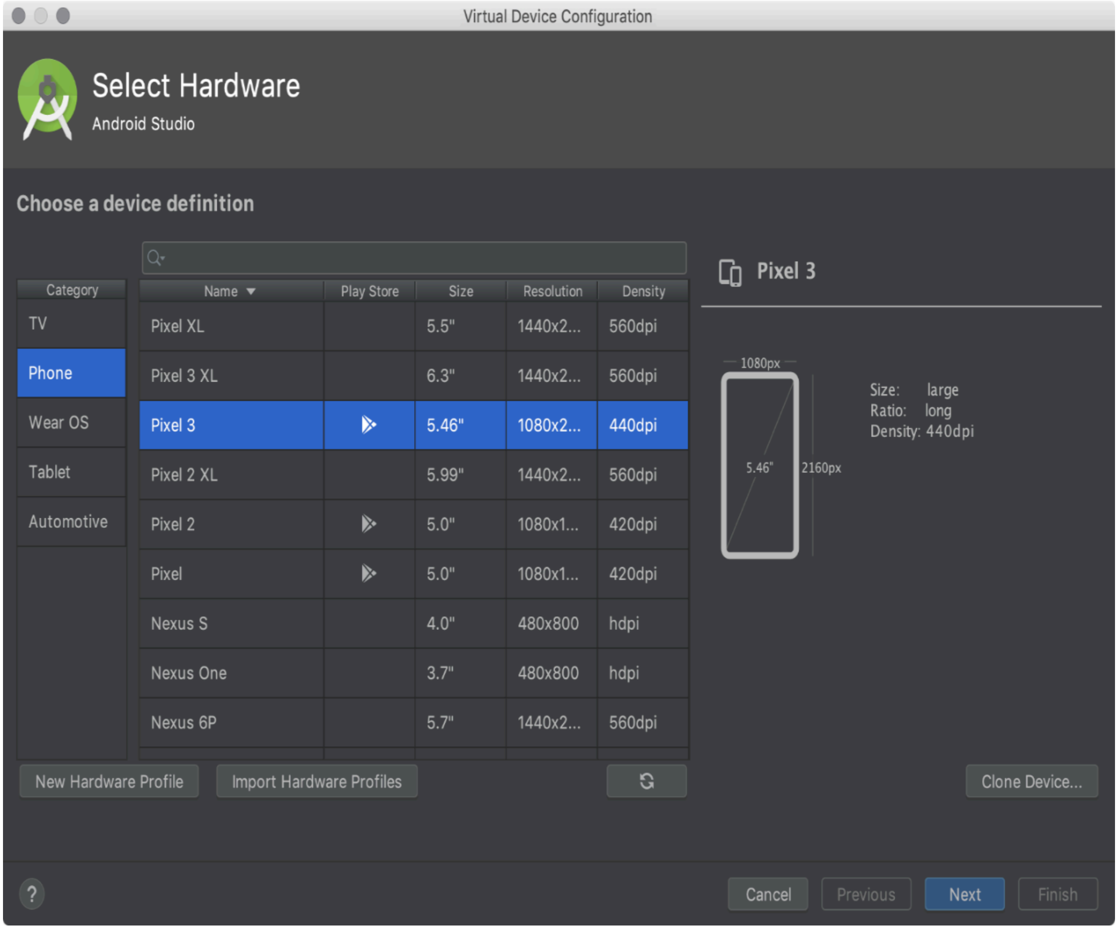
To create a new AVD:

1. Open the AVD Manager by clicking **Tools > AVD Manager**.

2. Click **Create Virtual Device**, at the bottom of the AVD Manager dialog.

The **Select Hardware** page appears.



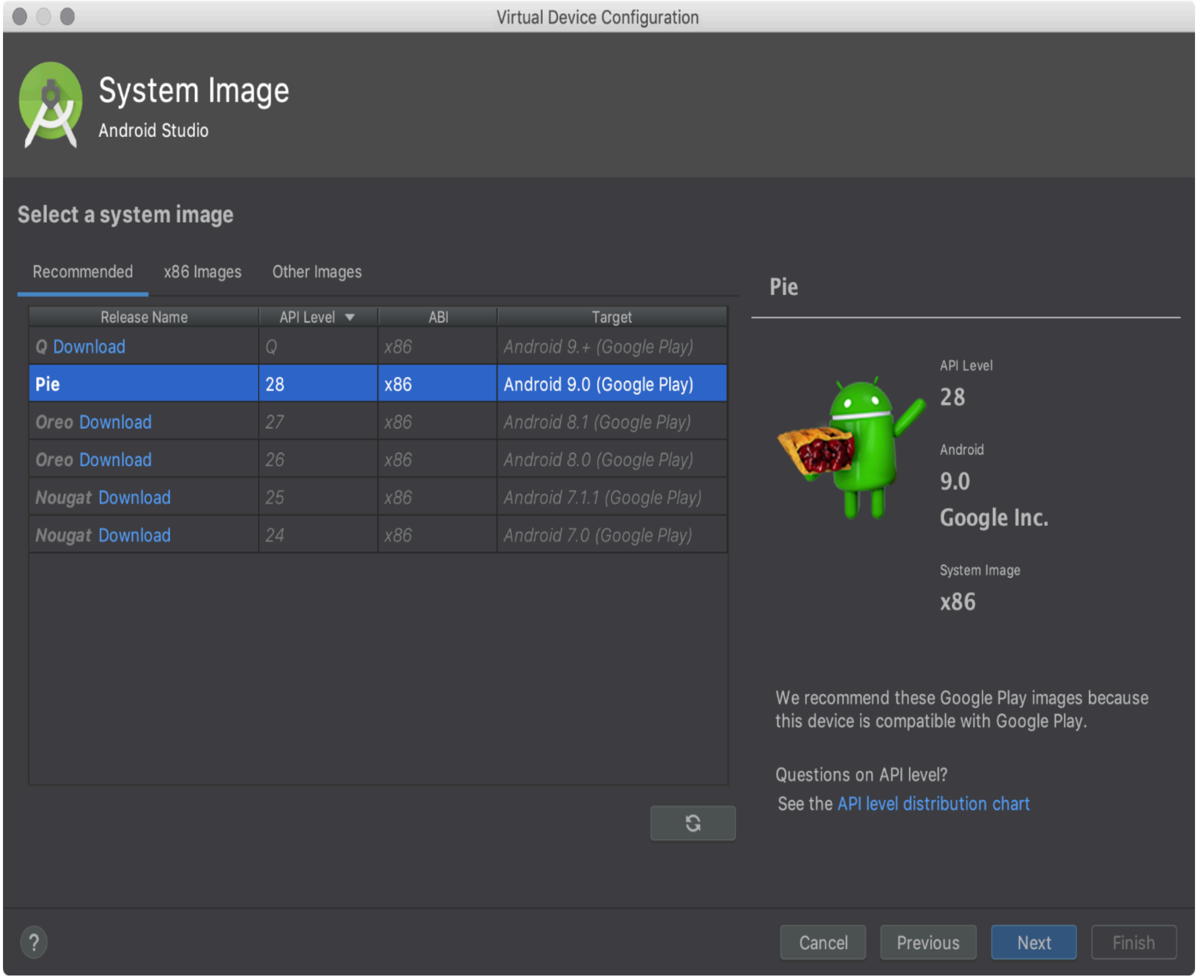


Notice that only some hardware profiles are indicated to include **Play Store**. This indicates that these profiles are fully CTS compliant and may use system images that include the Play Store app.

3. Select a hardware profile, and then click **Next**.

If you don't see the hardware profile you want, you can create or import a hardware profile.

The **System Image** page appears.



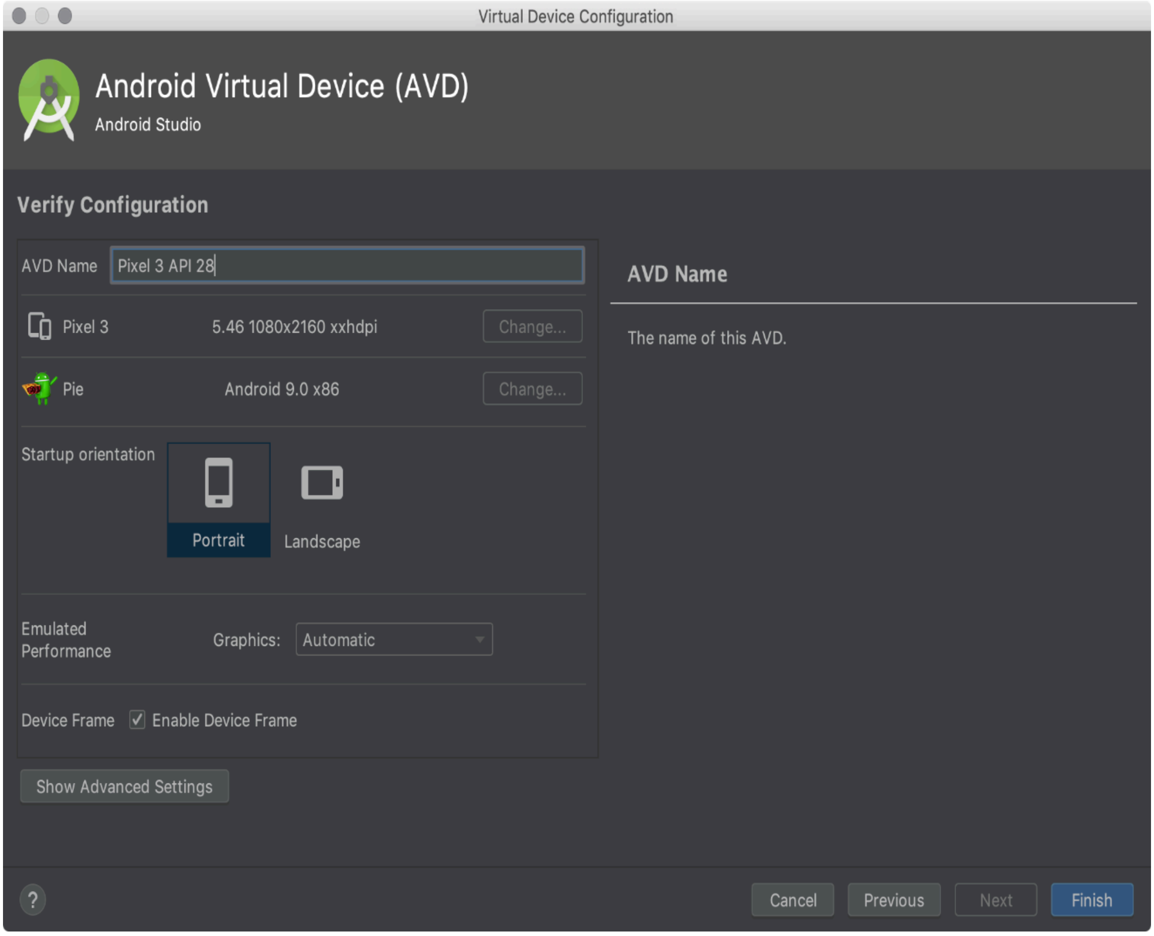
4. Select the system image for a particular API level, and then click **Next**.

The **Recommended** tab lists recommended system images. The other tabs include a more complete list. The right pane describes the selected system image. x86 images run the fastest in the emulator. If you see **Download** next to the system image, you need to click it to download the system image. You must be connected to the internet to download it.

The API level of the target device is important, because your app won't be able to run on a system image with an API level that's less than that required by your app, as specified in the minSdkVersion attribute of the app manifest file. For more information about the relationship between system API level and minSdkVersion, see Versioning Your Apps.

If your app declares a <uses-library> element in the manifest file, the app requires a system image in which that external library is present. If you want to run your app on an emulator, create an AVD that includes the required library. To do so, you might need to use an add-on component for the AVD platform; for example, the Google APIs add-on contains the Google Maps library.

The **Verify Configuration** page appears.



5. Change AVD properties as needed, and then click **Finish**.

Click **Show Advanced Settings** to show more settings, such as the skin.

The new AVD appears in the **Your Virtual Devices** page or the **Select Deployment Target** dialog.

To create an AVD starting with a copy:

1. From the **Your Virtual Devices** page of the AVD Manager, right-click an AVD and select **Duplicate**.

Or click Menu ▾ and select **Duplicate**.

The **Verify Configuration** page appears.

2. Click **Change** or **Previous** if you need to make changes on the **System Image** and **Select Hardware** pages.

3. Make your changes, and then click **Finish**.

The AVD appears in the **Your Virtual Devices** page.

LIST OF EXPERIMENTS

1. Create Hello World Android App using **Android Studio** and explain each step in detail.
2. Create an Activity that receive name form the user and displays **Hello Name** to the user using Android Studio.
3. Create an Activity that demonstrates the Life Cycle of an Activity.
4. Create an Android Application which receives URL form the user and open appropriate page in the system browser with the help of Implicit Intents using Android Studio.
5. Create an Android App which receives name form the user and displays welcome name in Second Activity.
6. Create Login Screen Application which shows Home screen if Login success otherwise displays error message using Android Studio.
7. Write an Android application program that demonstrate the use of
   a. RelativeLayout.
   b. LinearLayout.
   c. GridLayout.
   d. TableLayout.
8. Write an Android application program that demonstrates the use ImageView.
9. Write an Android application program that demonstrates the use of ListView and ArrayAdapter.
10. Write an Android application program that demonstrates how to create Custom ListView and Custom Adapters.

11. Write an Android application program that demonstrates the use of SQLite Database and Cursor.
12. Write an Android application program that demonstrates the use AsyncTask.
13. Write an Android application program that demonstrates Notifications.
14. Write an Android application program that demonstrates Shared Preferences.
15. Write an Android application program that connect to the internet, gets JSON data and displays the result in UI by parsing JSON data.

EXPERIMENT – I

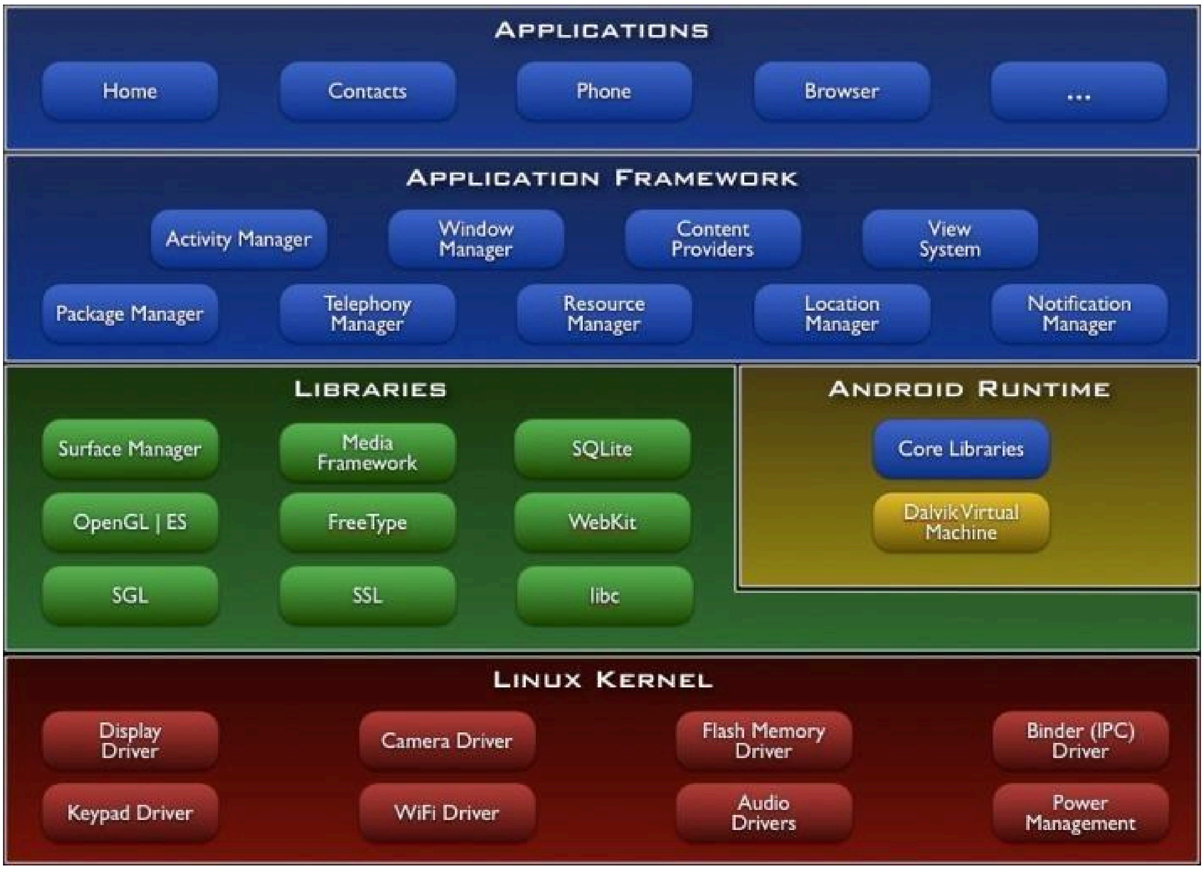
AIM: To study Android Studio and android studio installation. Create "Hello World" application.

Android:

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touch screen mobile devices such as smart phones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google. Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The current stable version is Android 10, released on September 3, 2019.

Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



Linux kernel

At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

Libraries

On top of Linux kernel there is a set of libraries including open -source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.

Android UI

An Android application user interface is everything that the user can see and interact with.

Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Project Structure

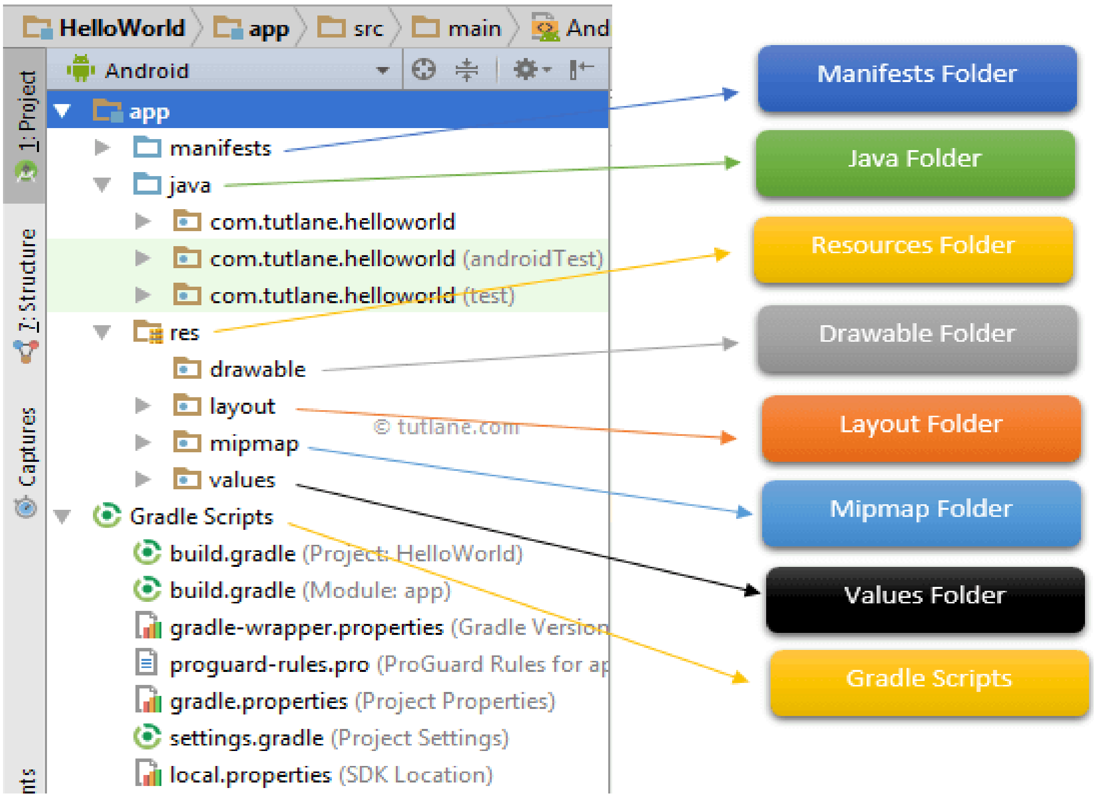


Figure 1. The project files in Android view

Each project in Android Studio contains one or more modules with source code files and resource files.

Types of modules include:

- **Android app modules**
- **Library modules**
- **Google App Engine modules**

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- **manifests:** Contains the AndroidManifest.xml file.
- **java:** Contains the Java source code files, including JUnit test code.
- **res:** Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**). You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

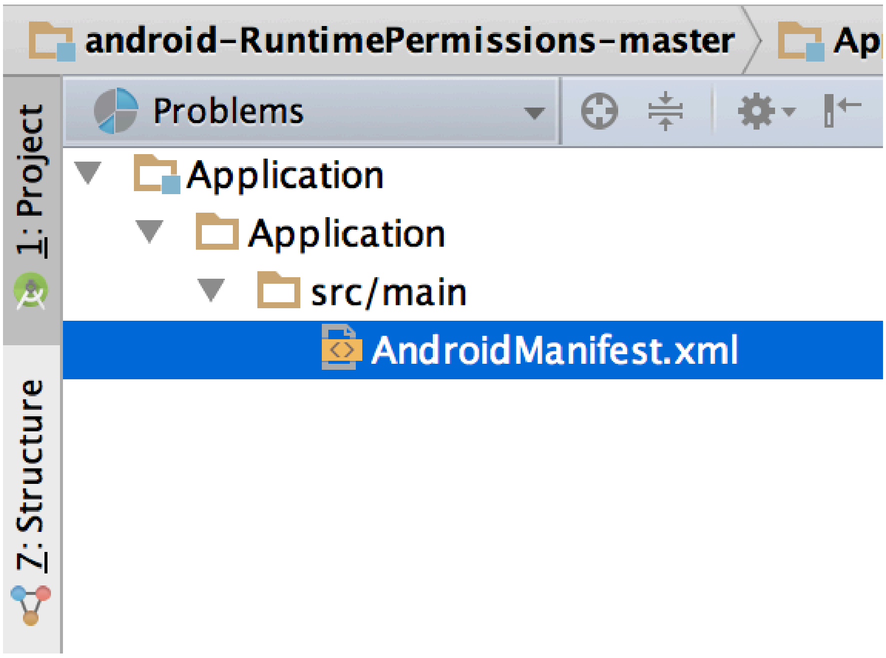


Figure 2. The project files in Problems view, showing a layout file with a problem.

The user interface

The Android Studio main window is made up of several logical areas identified in figure 3.

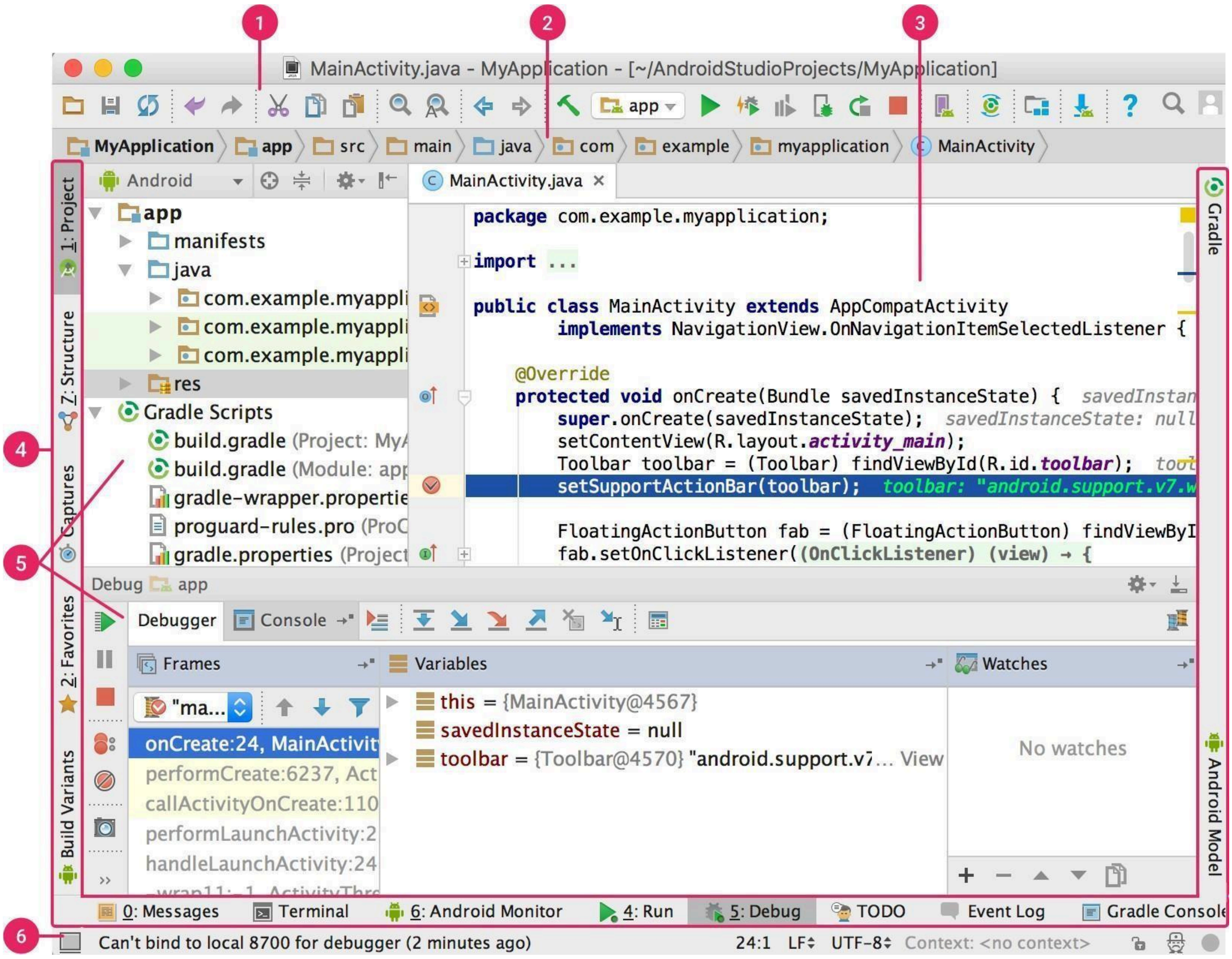


Figure 3. The Android Studio main window.

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.

2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.

3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

6. The **status bar** displays the status of your project and the IDE itself, as well as any

warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Tool windows

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

- To expand or collapse a tool window, click the tool's name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.
- To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.
- To show or hide the entire tool window bar, click the window icon in the bottom left-hand corner of the Android Studio window.
- To locate a specific tool window, hover over the window icon and select the tool window from the menu. You can also use keyboard shortcuts to open tool windows. Table 1 lists the shortcuts for the most common windows.

Table 1. Keyboard shortcuts for some useful tool windows.

Tool window	Windows and Linux	Mac
Project	Alt+1	Command+1
Version Control	Alt+9	Command+9
Run	Shift+F10	Control+R
Debug	Shift+F9	Control+D
Logcat	Alt+6	Command+6
Return to Editor	Esc	Esc
Hide All Tool Windows	Control+Shift+F12	Command+Shift+F12

If you want to hide all toolbars, tool windows, and editor tabs, click **View > Enter Distraction Free Mode**. This enables *Distraction Free Mode*. To exit Distraction Free Mode, click **View > Exit**

Distraction Free Mode.

You can use *Speed Search* to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

Code completion

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

Table 2. Keyboard shortcuts for code completion.

Type	Description	Windows and Linux	Mac
Basic Completion	Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static members.	Control + Space	Control+Space
Smart Completion	Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice in a row, you see more results, including chains.	Control + Shift + Space	Control+Shift+S pace

Statement Completion	Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc.	Control + Shift + Enter	Shift+Command +Enter

Android Studio Installation

To install Android Studio on Windows, proceed as follows:

1. If you downloaded an .exe file (recommended), double-click to launch it.

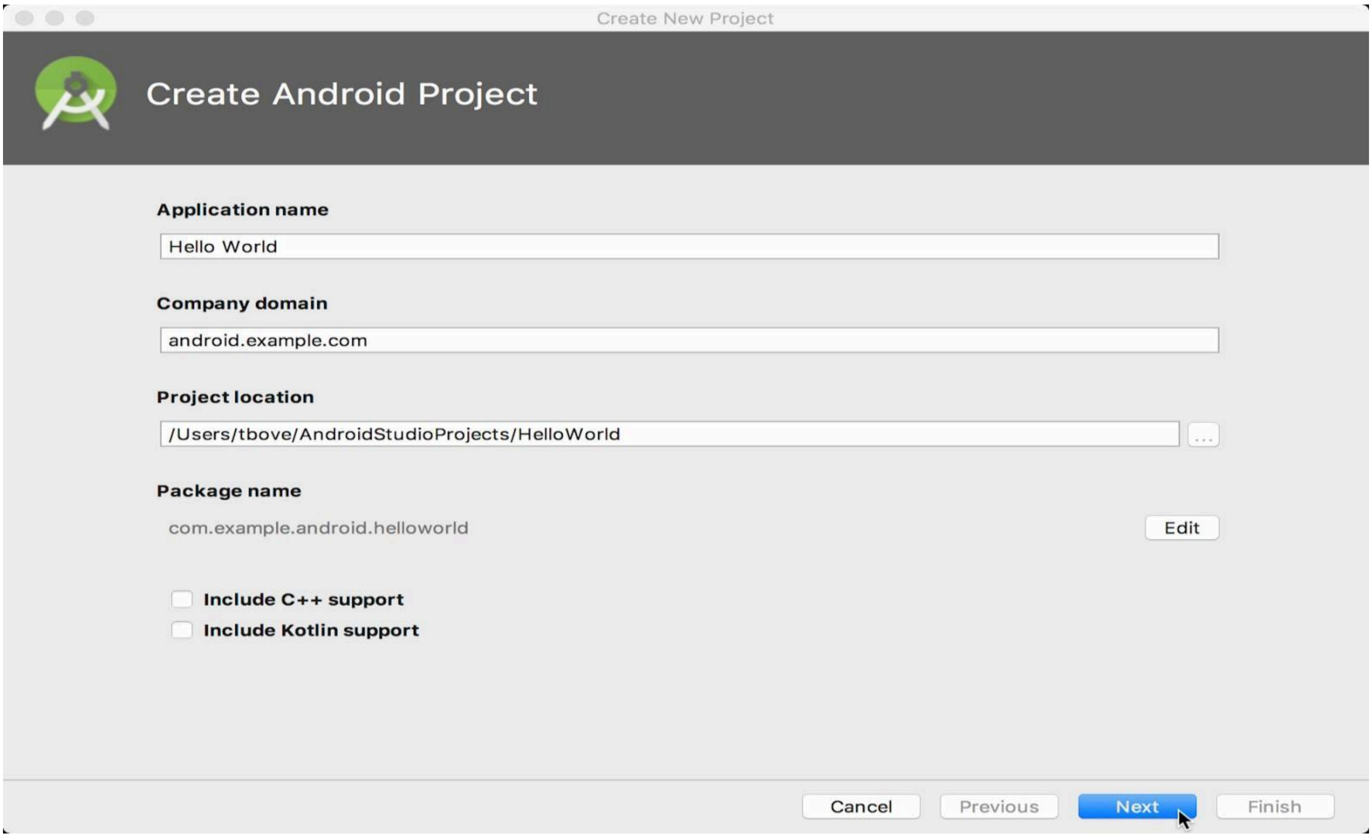
If you downloaded a .zip file, unpack the ZIP, copy the **android-studio** folder into your **Program Files** folder, and then open the **android-studio > bin** folder and launch studio64.exe (for 64-bit machines) or studio.exe (for 32-bit machines).

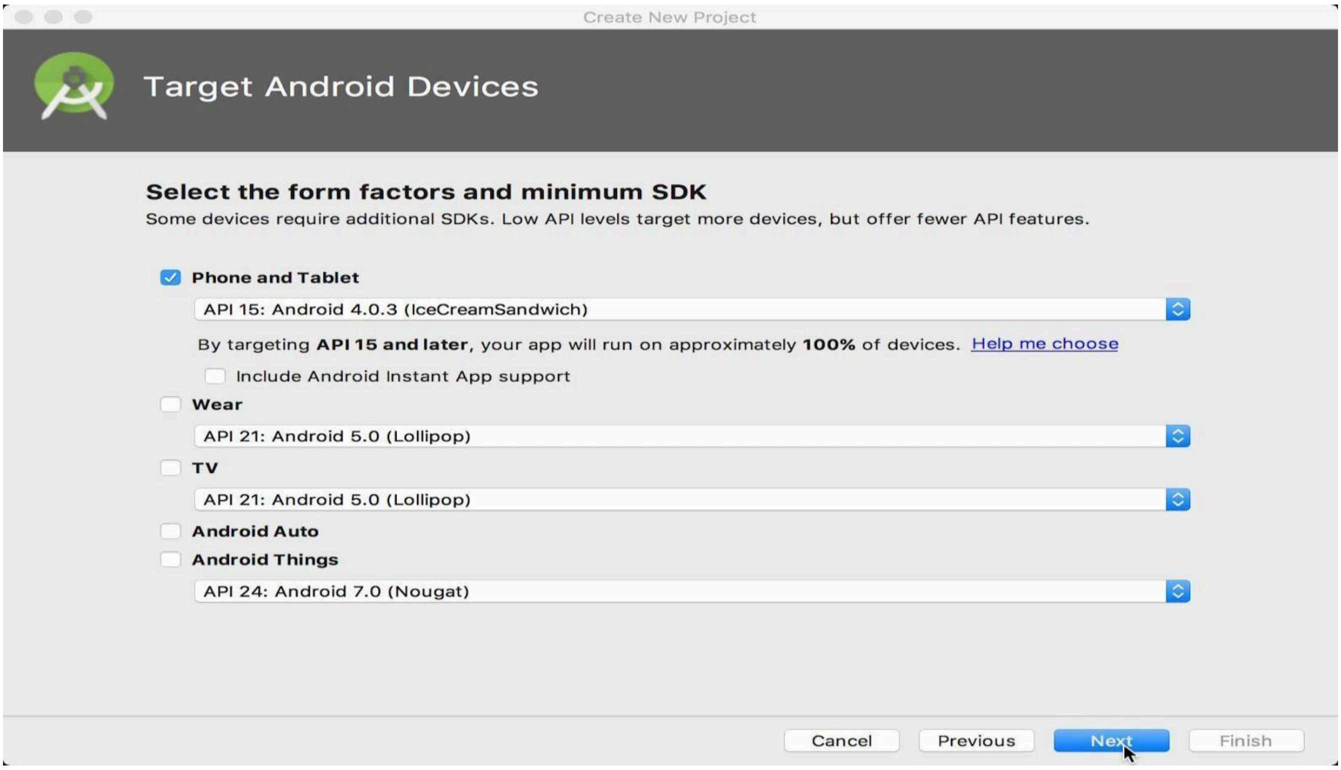
2. Follow the setup wizard in Android Studio and install any SDK packages that it recommends.

Create "Hello World" Application

After you successfully install Android Studio, you will create, from a template, a new project for the Hello World app. This simple app displays the string "Hello World" on the screen of the Android virtual or physical device.

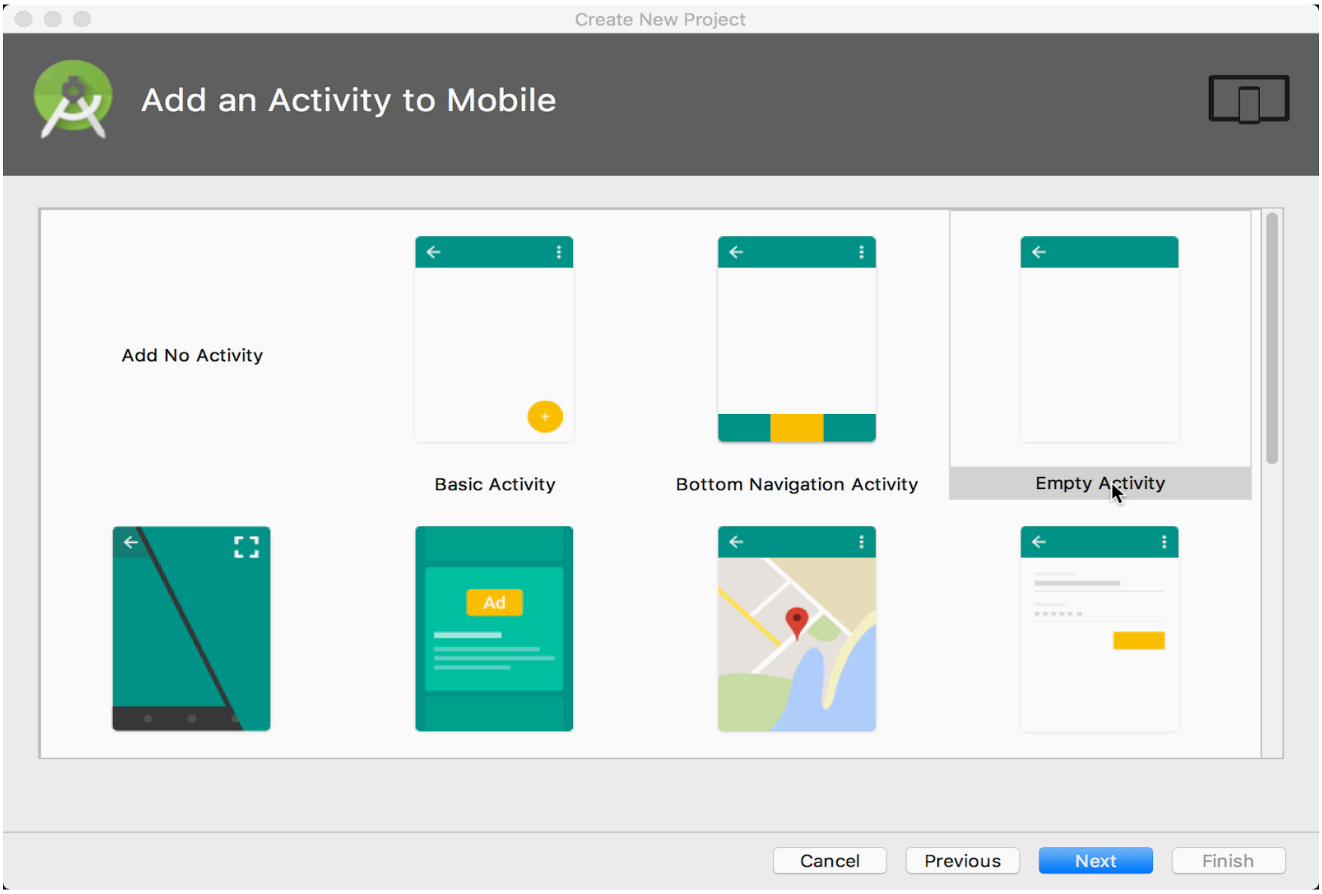
1. **Studio** window, click **Start a new Android Studio project**.
2. In the **Create Android Project** window, enter **Hello World** for the **Application name**.
3. Verify that the default **Project location** is where you want to store your Hello World app and other Android Studio projects, or change it to your preferred directory.
4. Accept the default **android.example.com** for **Company Domain**, or create a unique company domain. If you are not planning to publish your app, you can accept the default. Be aware that changing the package name of your app later is extra work.
5. Leave unchecked the options to **Include C++ support** and **Include Kotlin support**, and click **Next**.
6. On the **Target Android Devices** screen, **Phone and Tablet** should be selected. Ensure that **API 15: Android 4.0.3 IceCreamSandwich** is set as the Minimum SDK; if it is not, use the popup menu to set it.


Create New Project
Create Android Project
Application name
Hello World
Company domain
android.example.com
Project location
/Users/tbove/AndroidStudioProjects/HelloWorld
Package name
com.example.android.helloworld
Edit
Include C++ support
Include Kotlin support
Cancel
Previous
Next
Finish


Create New Project
Target Android Devices
Select the form factors and minimum SDK
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.
Phone and Tablet
API 15: Android 4.0.3 (IceCreamSandwich)
By targeting API 15 and later, your app will run on approximately 100% of devices. Help me choose
Include Android Instant App support
Wear
API 21: Android 5.0 (Lollipop)
TV
API 21: Android 5.0 (Lollipop)
Android Auto
Android Things
API 24: Android 7.0 (Nougat)
Cancel
Previous
Next
Finish

7. Leave unchecked the **Include Instant App support** and all other options. Then click **next**. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically.

8. The **Add an Activity** window appears. An Activity is a single, focused thing that the user can do. It is a crucial component of any Android app. An Activity typically has a layout associated with it that defines how UI elements appear on a screen. Android Studio provides Activity templates to help you get started. For the Hello World project, choose **Empty Activity** as shown below, and click **next**.

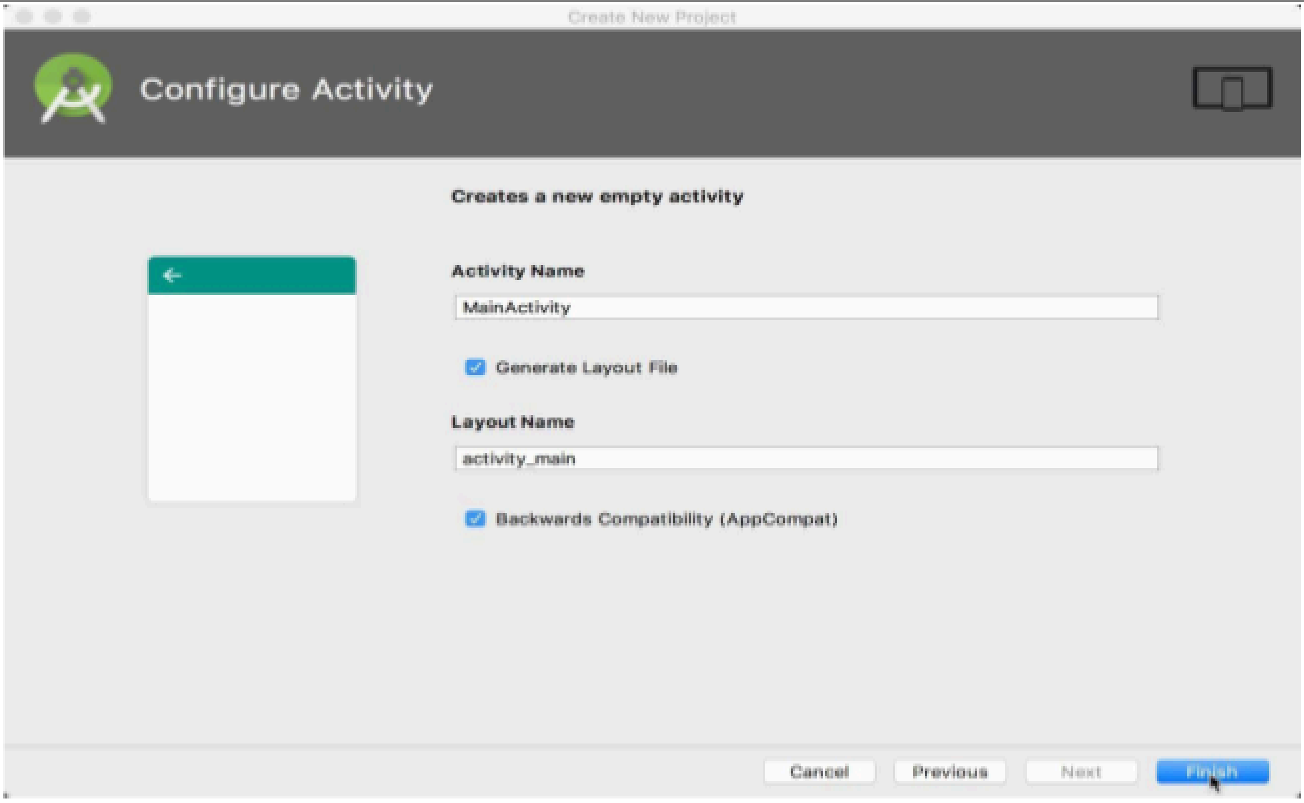


9. The **Configure Activity** screen appears (which differs depending on which template you chose in the previous step). By default, the empty Activity provided by the template is named MainActivity. You can change this if you want, but this lesson uses MainActivity.

10. Make sure that the **Generate Layout file** option is checked. The layout name by default is activity_main. You can change this if you want, but this lesson uses activity_main.

11. Make sure that the **Backwards Compatibility (App Compat)** option is checked. This ensures that your app will be backwards-compatible with previous versions of Android.

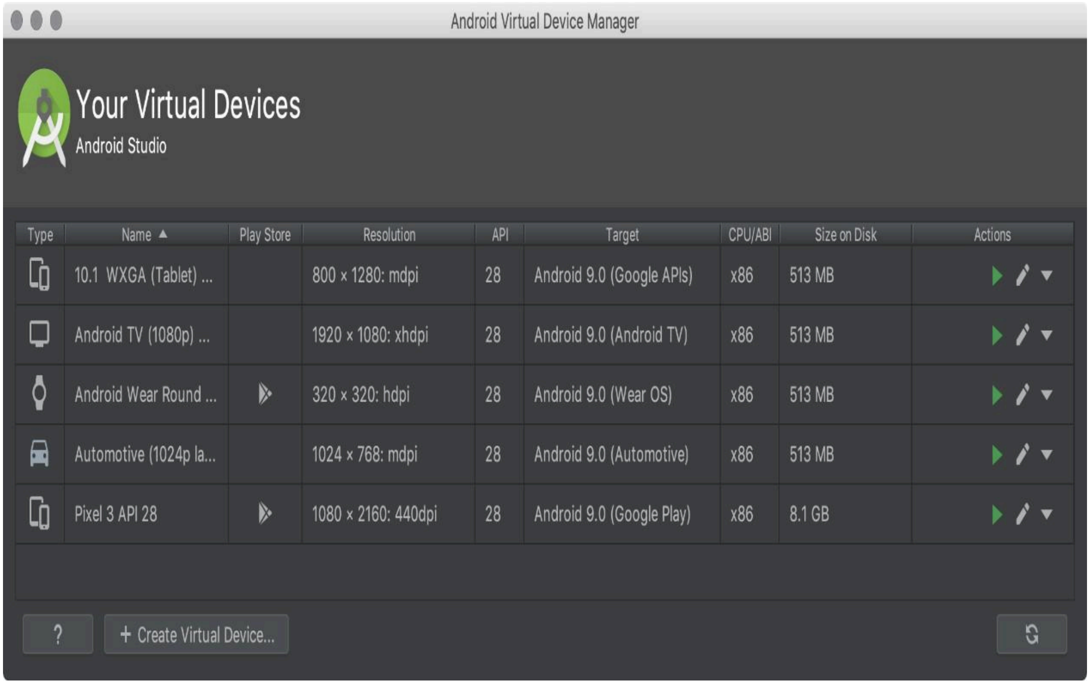
12. Click **Finish**.

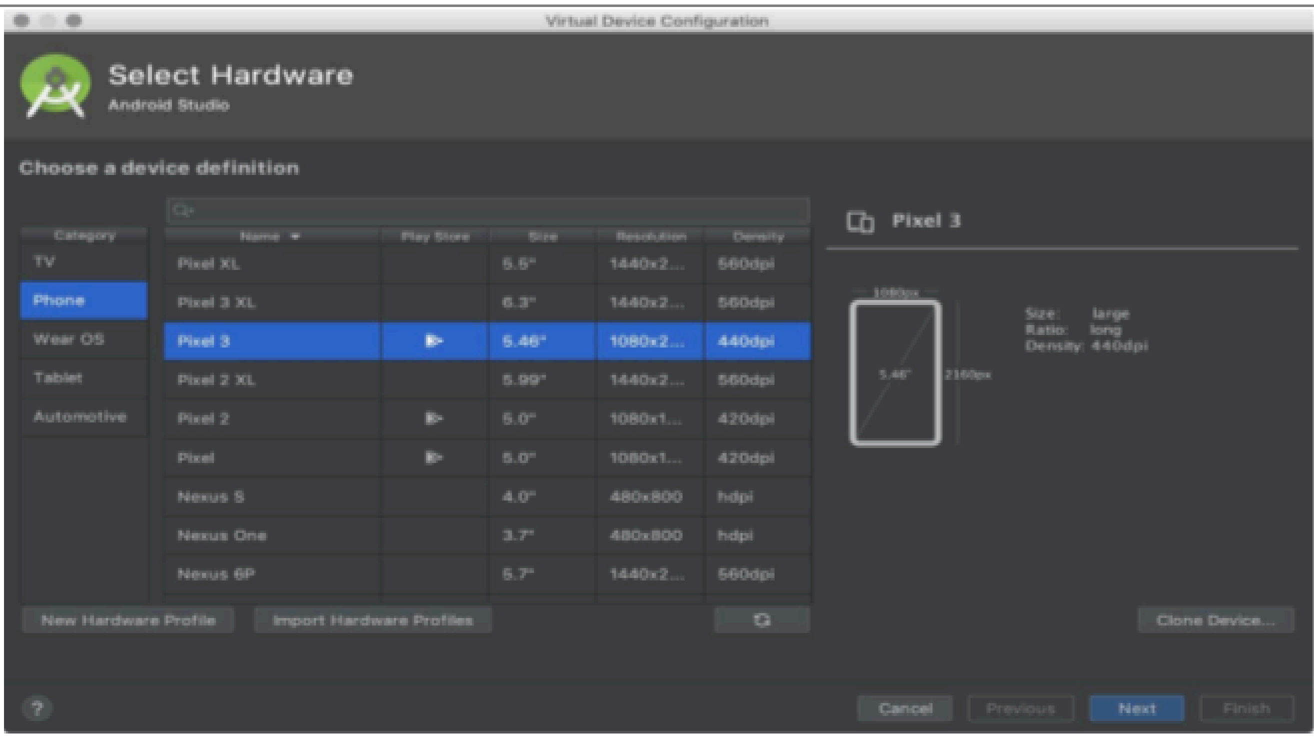


Create an AVD

To create a new AVD:

1. Open the AVD Manager by clicking **Tools > AVD Manager**.
2. Click **Create Virtual Device**, at the bottom of the AVD Manager dialog
3. The **Select Hardware** page appears.

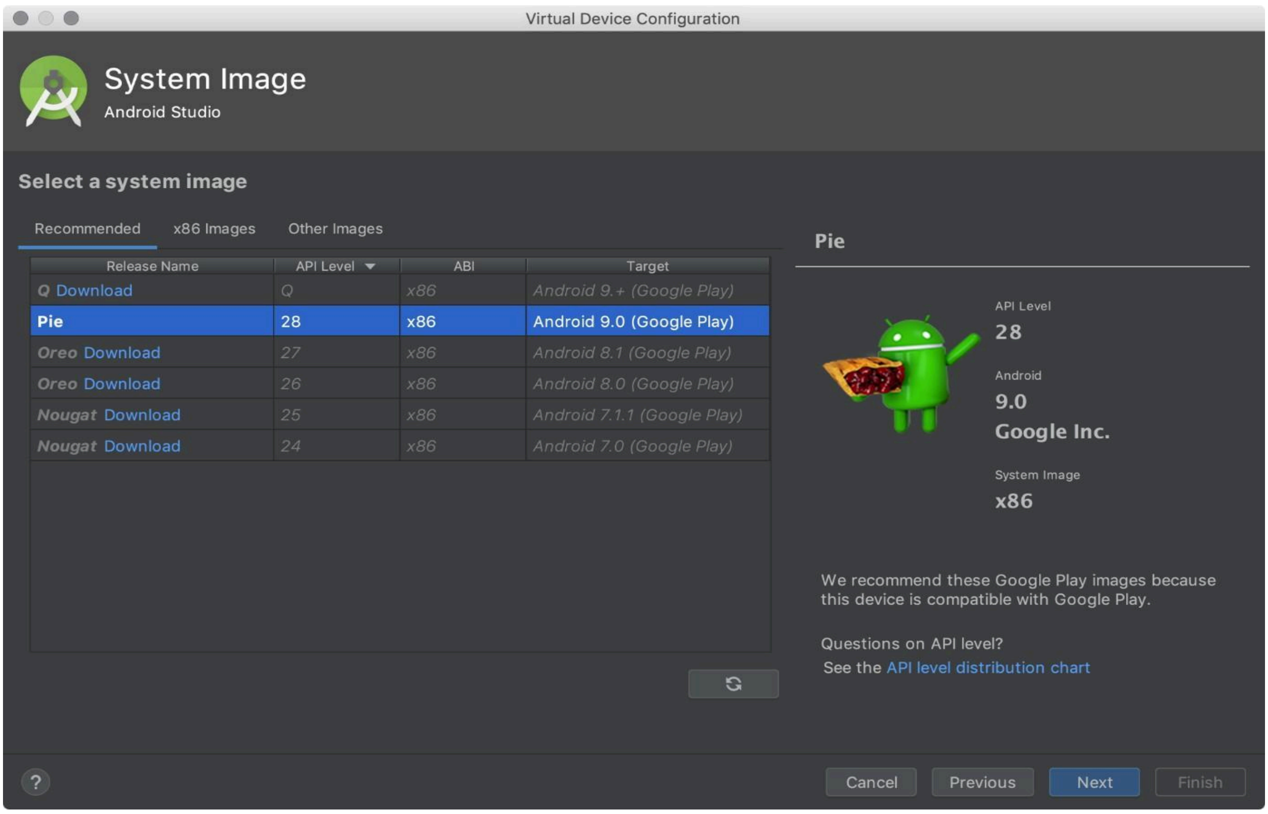




Notice that only some hardware profiles are indicated to include **Play Store**. This indicates that these profiles are fully CTS compliant and may use system images that include the Play Store app.

- Select a hardware profile, and then click **Next**.

If you don't see the hardware profile you want, you can create or import a hardware profile. The **System Image** page appears.



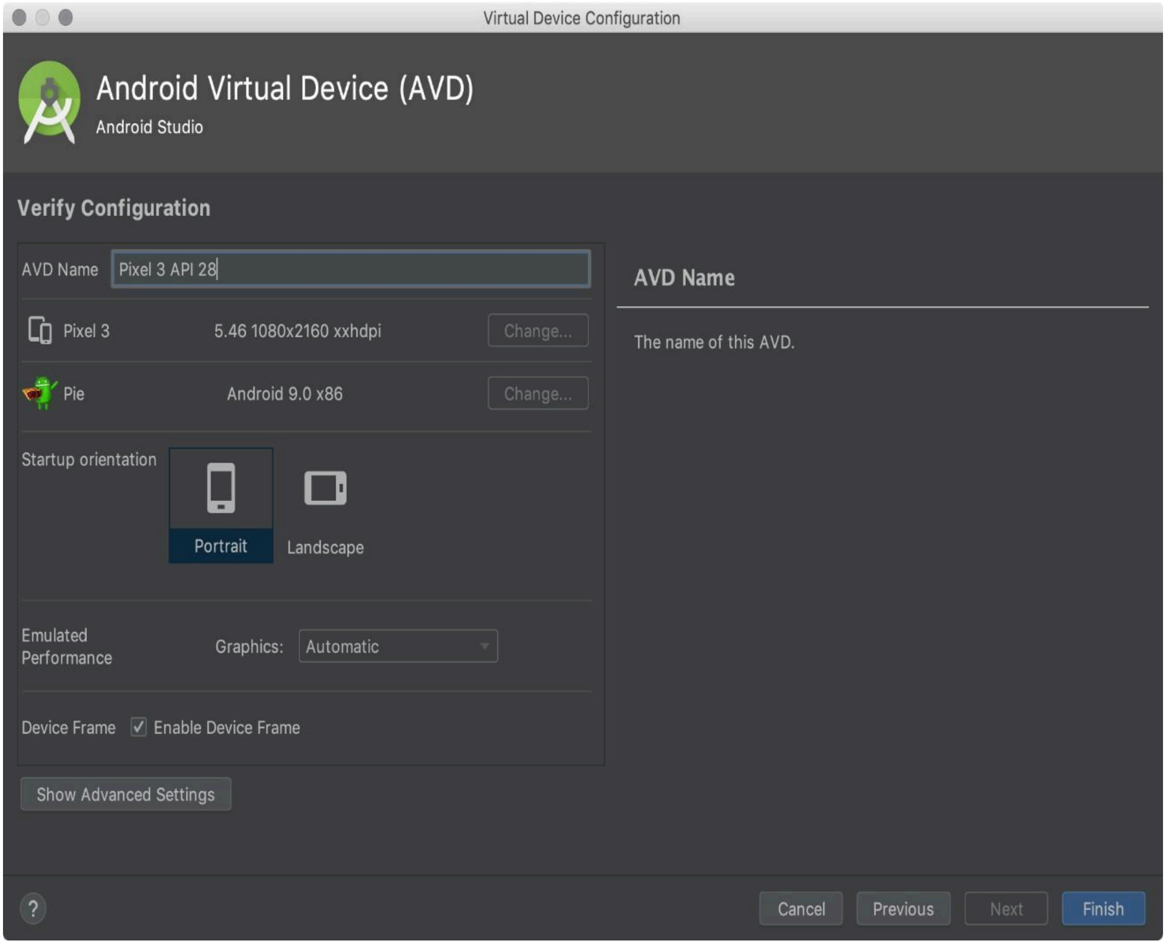
- Select the system image for a particular API level, and then click **Next**.

The **Recommended** tab lists recommended system images. The other tabs include a more complete list. The right pane describes the selected system image. x86 images run the fastest in the emulator. If you see **Download** next to the system image, you need to click it to download the system image. You must be connected to the internet to download it.

The API level of the target device is important, because your app won't be able to run on a system image with an API level that's less than that required by your app, as specified in the minSdkVersion attribute of the app manifest file. For more information about the relationship between system API level and minSdkVersion, see Versioning Your Apps.

If your app declares a <uses-library> element in the manifest file, the app requires a system image in which that external library is present. If you want to run your app on an emulator, create an AVD that includes the required library. To do so, you might need to use an add-on component for the AVD platform; for example, the Google APIs add-on contains the Google Maps library.

The **Verify Configuration** page appears.



- Change AVD properties as needed, and then click **Finish**.

Click **Show Advanced Settings** to show more settings, such as the skin.

The new AVD appears in the **Your Virtual Devices** page or the **Select Deployment Target** dialog.

To create an AVD starting with a copy:

1. From the **Your Virtual Devices** page of the AVD Manager, right-click an AVD and select **Duplicate**.

Or click Menu ▼ and select **Duplicate**. The **Verify Configuration** page appears.

2. Click **Change** or **Previous** if you need to make changes on the **System Image** and **Select Hardware** pages.

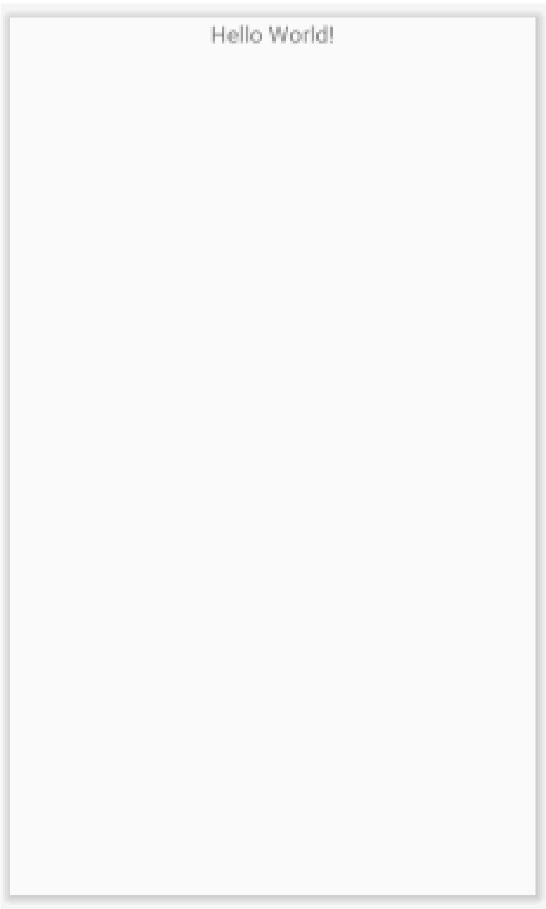
3. Make your changes, and then click **Finish**.

The AVD appears in the **Your Virtual Devices** page.

activity_main.xml:

```
<?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">
<TextView android:layout_width="wrap_content" android:layout_height="wrap_content"
android:text="Hello World!" android:layout_gravity="center" />
</LinearLayout>
```

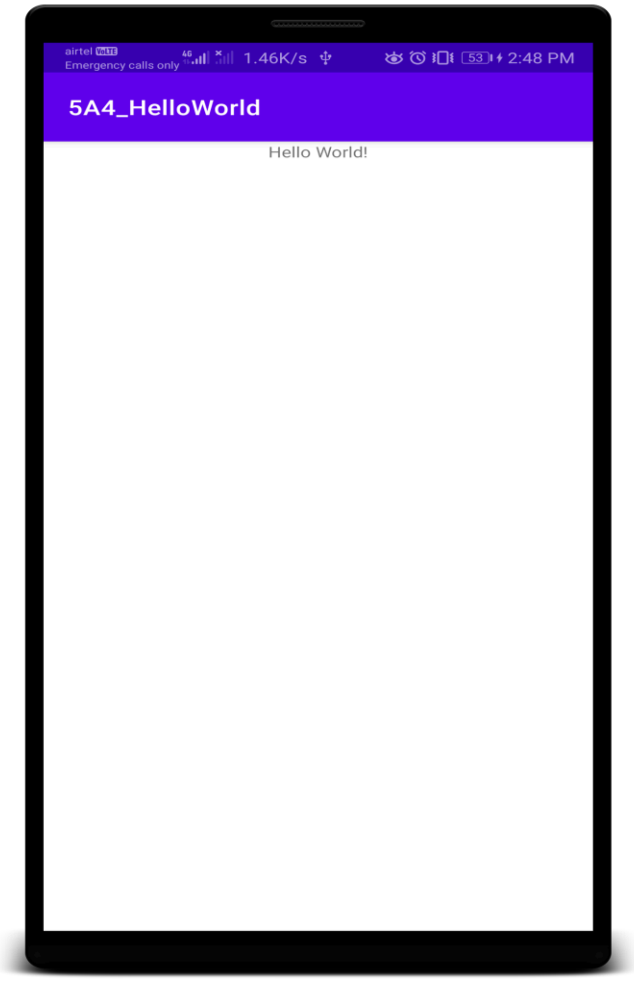
Layout Preview:



MainActivity.java:

```
package com.uday.android.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);
}
}
```

Output:



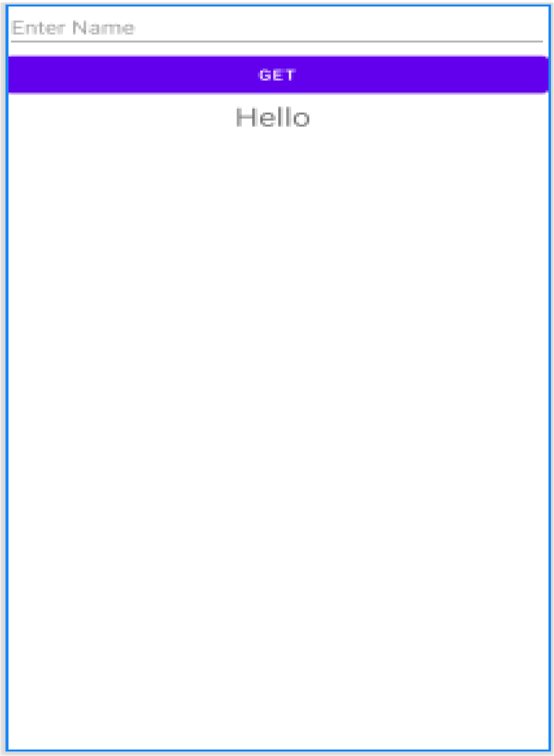
EXPERIMENT – II

AIM: **Create an Activity that receive name form the user and displays Hello Name to the user using Android Studio.**

activity_main.xml:

```
<?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/name"
android:layout_width="match_parent" android:layout_height="wrap_content" android:hint="Enter
Name"/>
<Button
android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/get" android:onClick="get" android:text="GET"
/>
<TextView android:id="@+id/wishes" android:layout_gravity="center" android:textSize="25sp"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Hello
"
/>
</LinearLayout>
```

Layout Preview:

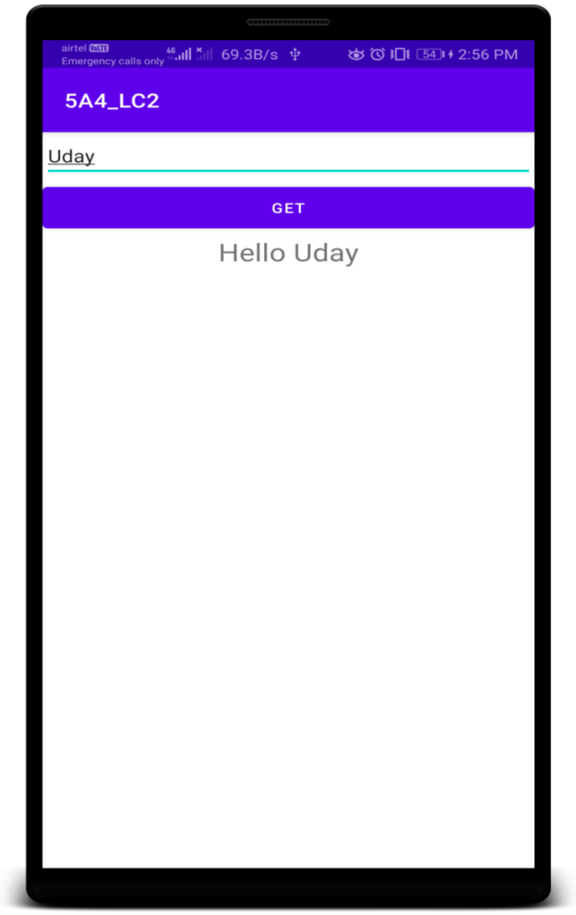


MainActivity.java:

```
package com.uday.android.lc2;

import androidx.appcompat.app.AppCompatActivity; import android.os.Bundle;
import android.view.View; import android.widget.TextView;
public class MainActivity extends AppCompatActivity { TextView name;
TextView tv; @Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main); name=findViewById(R.id.name);
tv=findViewById(R.id.wishes);
}

public void get(View view) {
```

```
tv.append(name.getText().toString());
}
}
```

Output:



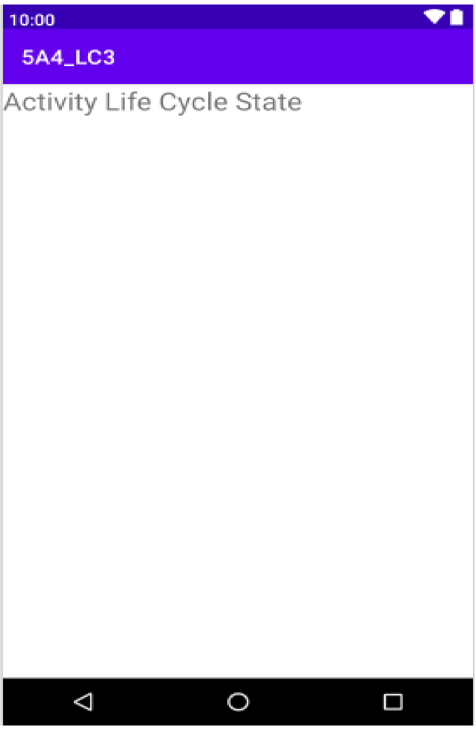
EXPERIMENT – III

AIM: **Create an Activity that demonstrates the Life Cycle of an Activity.**

activity_main.xml:

```
<?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"
tools:context=".MainActivity">
<TextView android:id="@+id/tv"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:baselineAligned="false" android:text="Activity Life Cycle State" android:textSize="25sp"
/>
</LinearLayout>
```

Layout Preview:

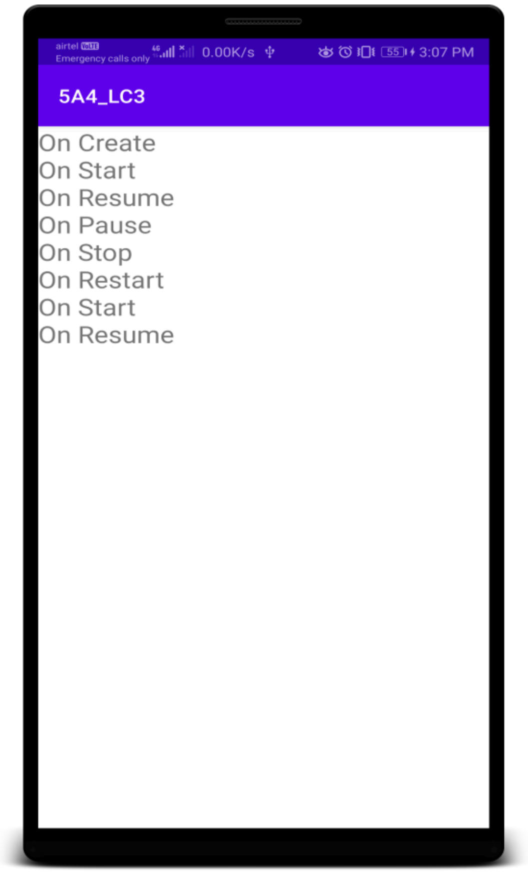


MainActivity.java:

```
package com.uday.android.a5a4_lc3;
import androidx.appcompat.app.AppCompatActivity; import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity { TextView tv;
@Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
tv = findViewById(R.id.tv); tv.setText("On Create\n");
}
@Override
protected void onStart() { super.onStart(); tv.append("On Start\n");
}
@Override
protected void onRestart() { super.onRestart(); tv.append("On Restart\n");
}
@Override
protected void onResume() { super.onResume(); tv.append("On Resume\n");
}
@Override
protected void onPause() { super.onPause(); tv.append("On Pause\n");
}
@Override
protected void onStop() {
```

```
super.onStop(); tv.append("On Stop\n");
}
@Override
protected void onDestroy() { super.onDestroy(); tv.setText("On Destroy\n");
}
}
```

Output:



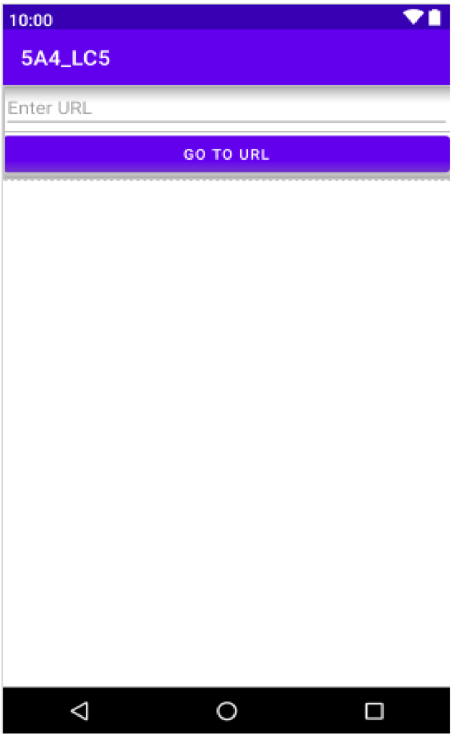
EXPERIMENT – IV

AIM: **Create an Android Application which receives URL form the user and open appropriate page in the system browser with the help of Implicit Intents using Android Studio.**

activity_main.xml:

```
<?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="wrap_content" tools:context=".MainActivity"
android:orientation="vertical">
<EditText android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/url" android:hint="Enter URL"/>
<Button
android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/button1" android:text="Go to URL" android:layout_gravity="center"
android:onClick="URI"/>
</LinearLayout>
```

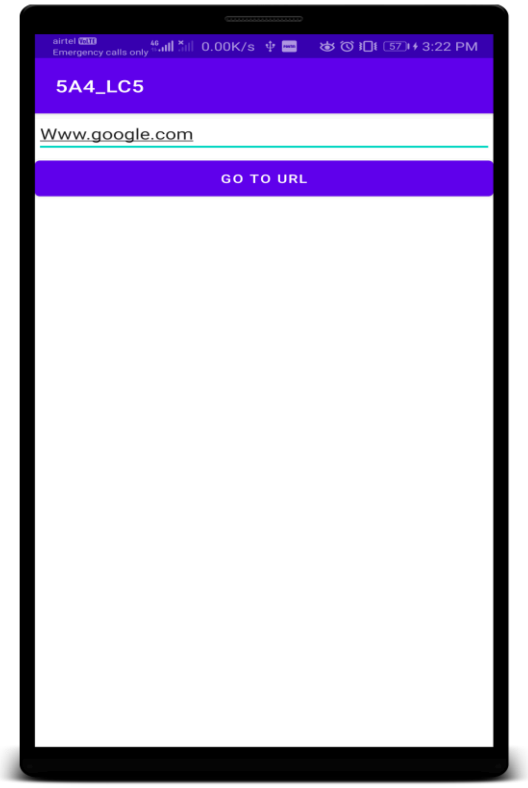
Layout Preview:



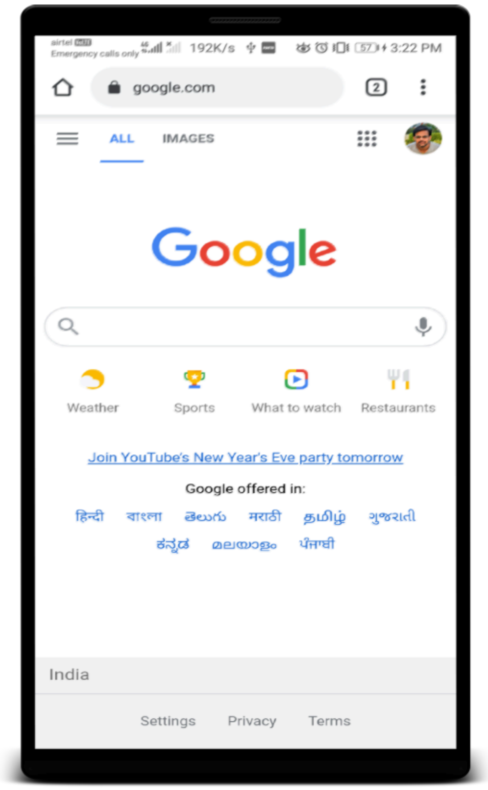
MainActivity.java:

```
package com.uday.android.a5a4_lc5; import android.content.Intent; import android.net.Uri;
import android.os.Bundle; import android.view.View;
import androidx.appcompat.app.AppCompatActivity; public class MainActivity extends
AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
public void URI(View view) {
Uri u = Uri.parse("https://www.google.com"); Intent i = new Intent(Intent.ACTION_VIEW,u);
startActivity(i);
}
}
```

Output:



5A4_LC5
Www.google.com
GO TO URL



google.com
ALL
IMAGES
Google
Weather
Sports
What to watch
Restaurants
Join YouTube's New Year's Eve party tomorrow
Google offered in:
हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી
ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ
India
Settings
Privacy
Terms

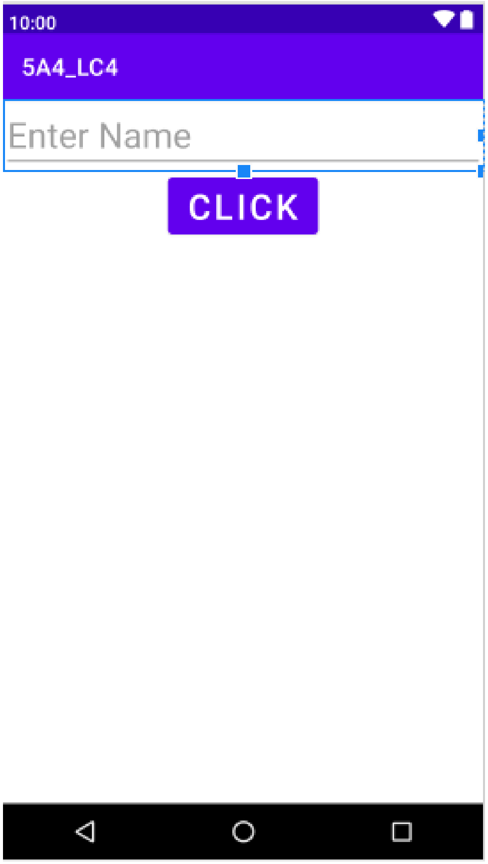
EXPERIMENT – V

AIM: Create an Android App which receives name form the user and displays welcome name in Second Activity.

activity_main.xml:

```
<?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:layout_width="match_parent" android:layout_height="wrap_content"
android:textSize="30sp" android:hint="Enter Name" android:id="@+id/edit"/>
<Button
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:textSize="30sp" android:text="CLICK" android:layout_gravity="center"
android:onClick="Next"/>
</LinearLayout>
```

Layout Preview:



activity_2.xml:

```
<?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" tools:context=".Activity2">
<TextView android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/text" android:textSize="50sp"/>

</LinearLayout>
```

Layout Preview:

MainActivity.java:

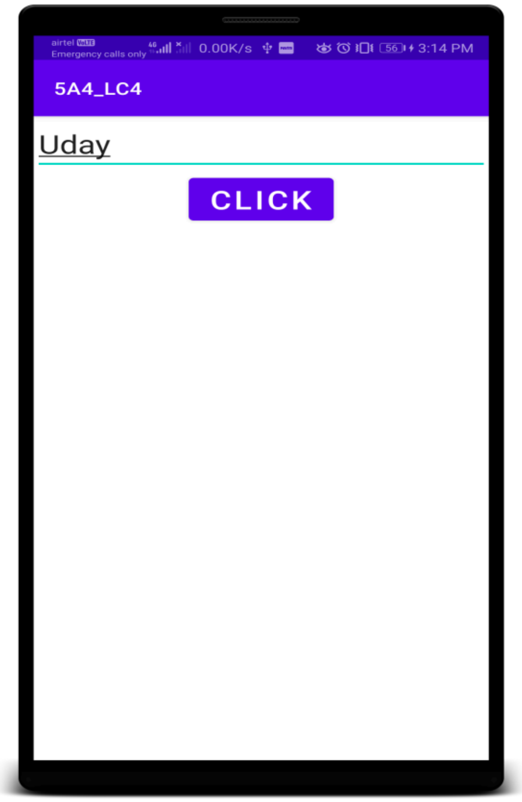
```
package com.uday.android.a5a4_lc4; import android.content.Intent; import android.os.Bundle;
import android.view.View; import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity; public class MainActivity extends
AppCompatActivity {
EditText e; @Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main); e=findViewById(R.id.edit);
}

public void Next(View view) { String s =e.getText().toString();
Intent i=new Intent(this, Activity2.class);
```

```
i.      putExtra("key",s); startActivity(i);
}
}
```

Activity2.java:

```
package com.uday.android.a5a4_lc4;


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


public class Activity2 extends AppCompatActivity { TextView tv;
@Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_2);
tv = findViewById(R.id.text);
String s= getIntent().getStringExtra("key"); tv.setText("Welcome "+s);
}
}
```

Output:


0.00K/s
3:14 PM
5A4_LC4
Uday
CLICK


0.00K/s
3:16 PM
5A4_LC4
Welcome Uday

EXPERIMENT – VI

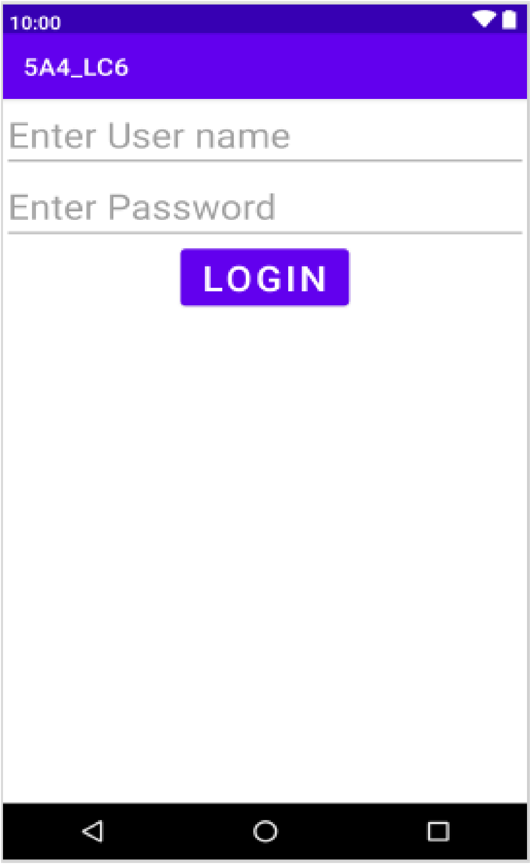
AIM: Create Login Screen Application which shows Home screen if Login success otherwise displays error message using Android Studio.

activity_main.xml:

```
<?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:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/edit1" android:hint="Enter User name" android:textSize="30sp"
android:inputType="text"/>

<EditText android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/edit2" android:textSize="30sp" android:hint="Enter Password"
android:inputType="textPassword"/>
```

```
<Button
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/button1" android:text="Login" android:layout_gravity="center"
android:textSize="30sp" android:onClick="Login"/>

<TextView android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/text" android:layout_gravity="center" android:textSize="30sp"/>
</LinearLayout>
```

Layout Preview-1:



activity_home_screen.xml:

```
<?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" tools:context=".HomeScreen">
<TextView android:layout_width="match_parent" android:layout_height="match_parent"
android:id="@+id/text" android:text="Welcome Dummy" android:textSize="30sp"/>
</LinearLayout>
```

Layout Preview-2:

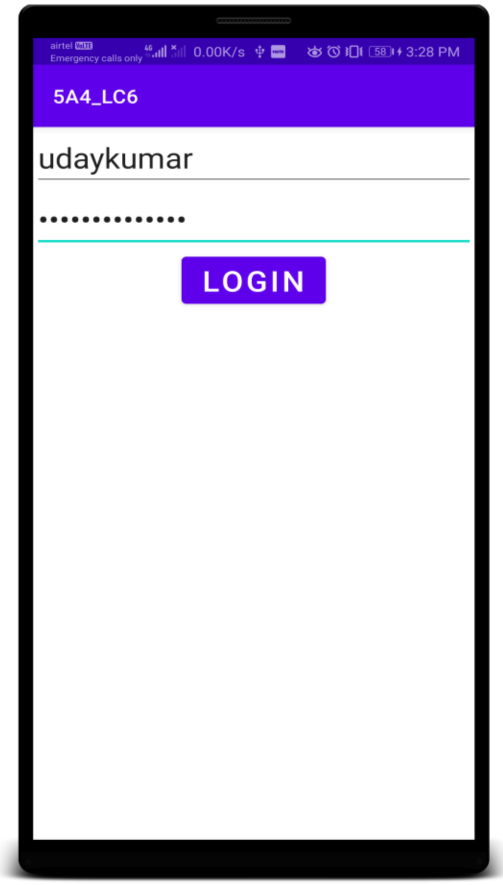
MainActivity.java:

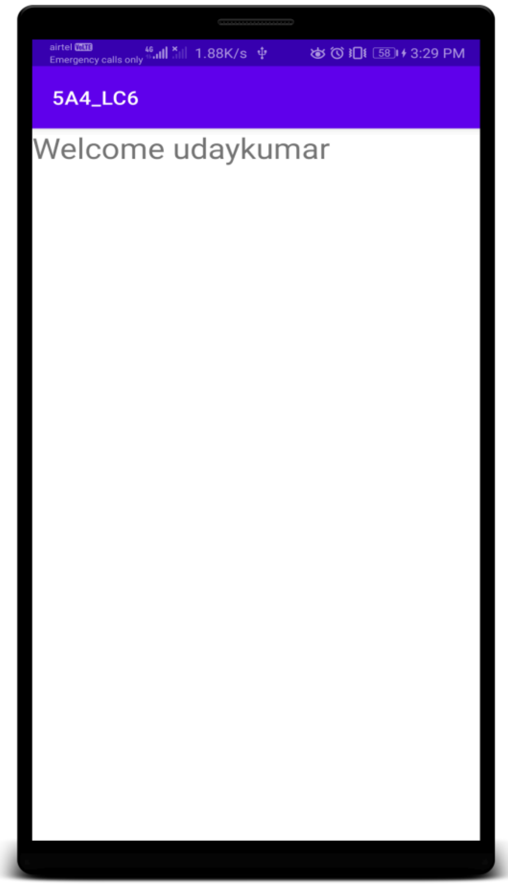
```
package com.uday.android.a5a4_lc6; import android.content.Intent; import android.os.Bundle;
import android.view.View; import android.widget.Button; import android.widget.EditText; import
android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity; public class MainActivity extends
AppCompatActivity {
EditText e1,e2; Button b1; TextView t1; @Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main); e1=findViewById(R.id.edit1);
e2=findViewById(R.id.edit2); t1=findViewById(R.id.text);
}
public void Login(View view) { String s1= e1.getText().toString(); String s2=
e2.getText().toString();
if(s1.equals("udaykumar")&&s2.equals("udaykumarthota")) { Intent i = new
Intent(this,HomeScreen.class); i.putExtra("key",s1);
startActivity(i);
}
else {
t1.setText("Invalid Credentials");
}
}
}
```

HomeScreen.java:

```
package com.uday.android.a5a4_lc6; import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity; public class HomeScreen extends
AppCompatActivity {
TextView tv1; @Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_home_screen); tv1=findViewById(R.id.text);
String s =getIntent().getStringExtra("key"); tv1.setText("Welcome "+s);
}
}
```

Output:





EXPERIMENT – VII

AIM: Write an android application program that demonstrates the use of

a) RelativeLayout b) LinearLayout

c) GridLayout d) TableLayout activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<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">

<TextView android:id="@+id/tv"

android:layout_width="match_parent" android:layout_height="wrap_content"

android:textSize="25sp" android:text="Hello World!"/>

<EditText android:layout_below="@id/tv" android:id="@+id/et"

android:layout_width="match_parent" android:layout_height="wrap_content" android:text="write

something" android:textSize="25sp" />

</RelativeLayout>
```

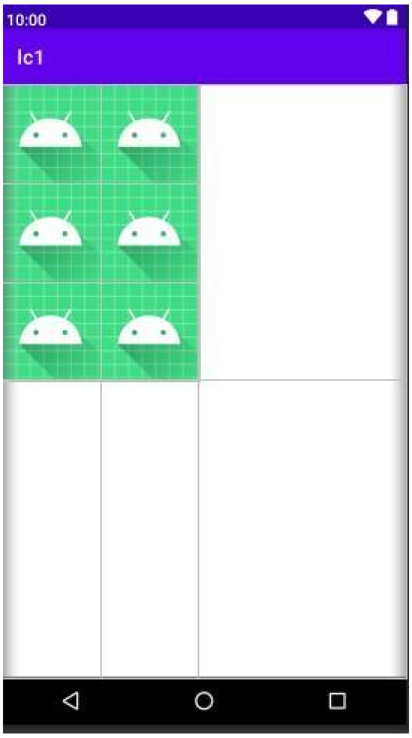
Layout Preview -1:

gridlayout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent" android:layout_height="match_parent">
<GridLayout android:layout_width="match_parent" android:layout_height="match_parent"
android:rowCount="3" android:columnCount="2">
<ImageView android:layout_width="100dp" android:layout_height="100dp"
android:src="@mipmap/ic_launcher" />
<ImageView android:layout_width="100dp" android:layout_height="100dp"
android:src="@mipmap/ic_launcher" />
```

```
<ImageView android:layout_width="100dp" android:layout_height="100dp"
android:src="@mipmap/ic_launcher" />
<ImageView android:layout_width="100dp" android:layout_height="100dp"
android:src="@mipmap/ic_launcher" />
<ImageView android:layout_width="100dp" android:layout_height="100dp"
android:src="@mipmap/ic_launcher" />
<ImageView android:layout_width="100dp" android:layout_height="100dp"
android:src="@mipmap/ic_launcher" />
                                        </GridLayout>
                                      </LinearLayout>
```

Layout Preview – 2:



linearlayout.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent" android:layout_height="match_parent"

android:orientation="vertical"

android:weightSum="1">

<TextView android:layout_width="match_parent" android:layout_height="wrap_content"

android:layout_weight="0.5" android:text="Hello World!" android:textSize="30sp" />

<LinearLayout android:layout_width="match_parent" android:layout_height="0dp"

android:layout_weight="0.5" android:weightSum="1">

<TextView android:layout_width="wrap_content" android:layout_height="wrap_content"

android:layout_weight="0.4" android:text="Hello World!" android:textSize="30sp" />

<TextView
```

```
android:layout_width="0dp" android:layout_height="wrap_content" android:layout_weight="0.3"

android:text="Hello World!" android:textSize="30sp" />

<TextView android:layout_width="0dp"

android:layout_height="wrap_content" android:layout_weight="0.3" android:text="Hello World!"

android:textSize="30sp" />

</LinearLayout>

</LinearLayout>
```

Layout Preview – 3:

tablelayout.xml:

```
<?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">
<TableRow>
<EditText android:layout_width="wrap_content" android:layout_height="wrap_content"
android:text="Type something!" />
<EditText android:layout_width="wrap_content" android:layout_height="wrap_content"
android:text="Type something!" />
</TableRow>
<TableRow>
<EditText android:layout_width="wrap_content" android:layout_height="wrap_content"
android:text="Type something!" />
<EditText android:layout_width="wrap_content" android:layout_height="wrap_content"
android:text="Type something!" />
</TableRow>
</TableLayout>
```

Layout Preview:

MyGridLayout.java

package com.uday.android.lc1;

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

public class MyGridLayout extends AppCompatActivity { @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.gridlayout); } }

MyLinearLayout.java:

package com.uday.android.lc1;

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

public class MyGridLayout extends AppCompatActivity {

```
@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.gridlayout);        } }
```

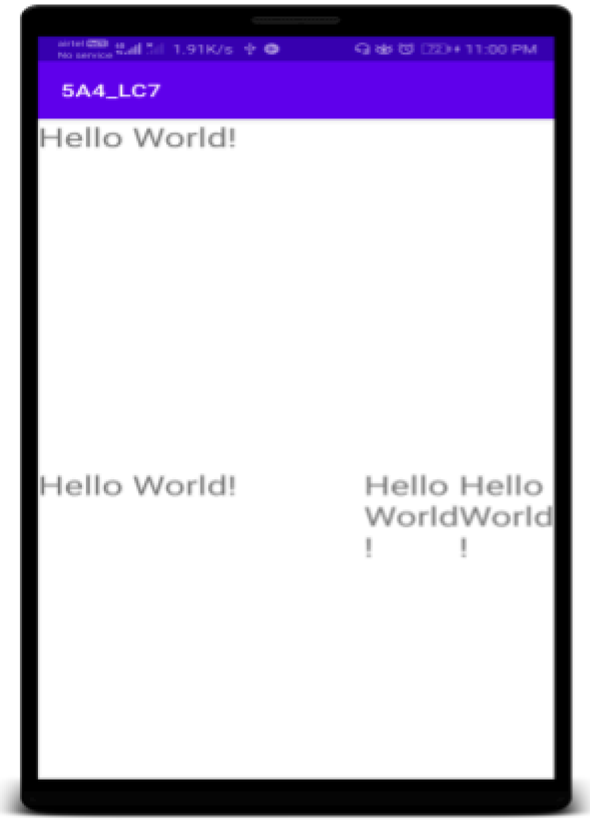
MyRelativeLayout.java:

```
package com.uday.android.lc1;

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

public class MyRelativeLayout extends AppCompatActivity { @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main1);        } }
```

MyTableLayout.java:

```
package com.uday.android.lc1;

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

public class MyTableLayout extends AppCompatActivity { @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.tablelayout);

}

}
```

Output:



Linear Layout

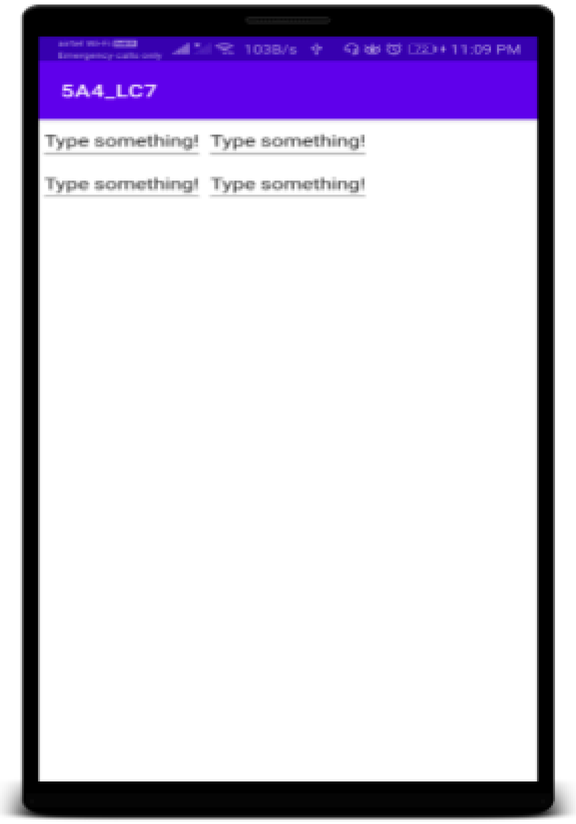
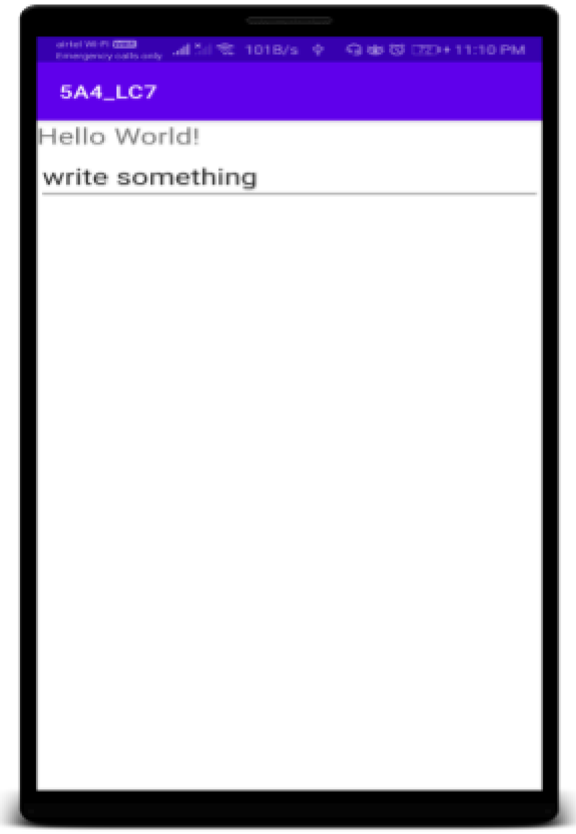
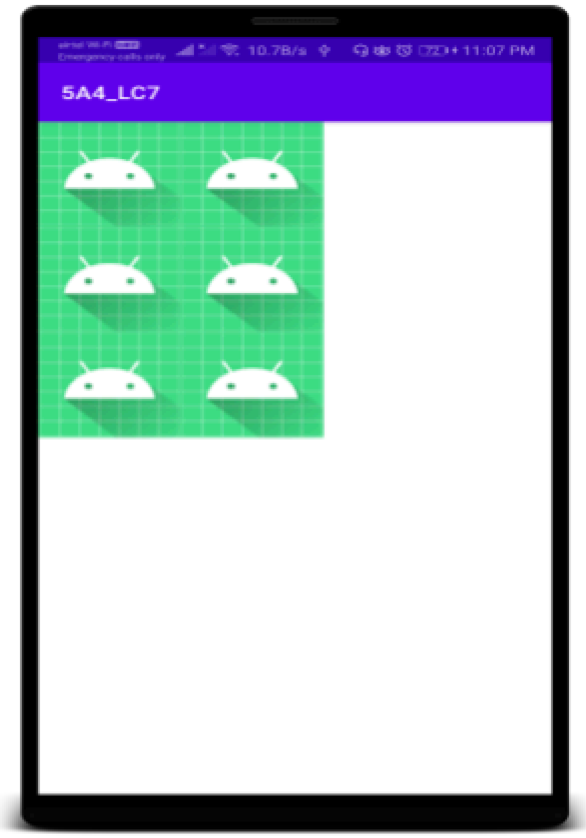


Table Layout



Relative Layout



Grid Layout

EXPERIMENT – VIII

AIM: Write an Android application program that demonstrates the use ImageView. activity_main.xml:

```
<?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" tools:context=".MainActivity">

<ImageView android:layout_width="match_parent" android:layout_height="match_parent"
android:src="@drawable/tree"/>

</LinearLayout>
```

Layout Preview:

MainActivity.java:

```
package com.example.imageview;
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);
}
}
```

Output:

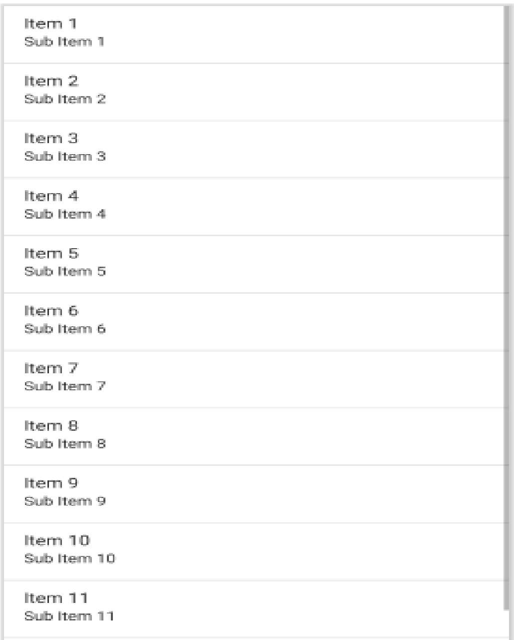
EXPERIMENT – IX

AIM: Write an Android application program that demonstrates the use of ListView and ArrayAdapter.

activity_main.xml:

```
<?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"
tools:context=".MainActivity">

<ListView android:layout_width="match_parent" android:layout_height="match_parent"
android:id="@+id/listview"/>
</LinearLayout>
```

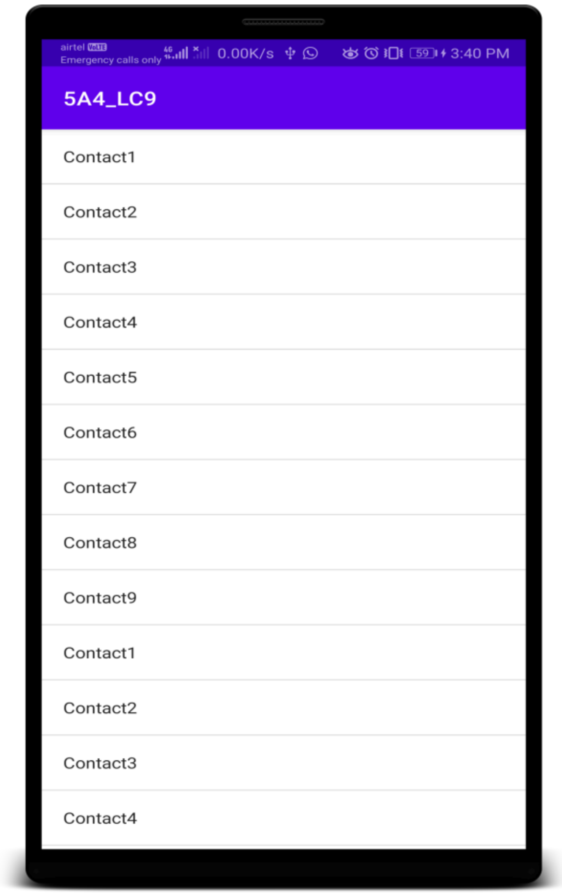
Layout Preview:



MainActivity.java:

```
package com.uday.android.a5a4_lc9;


import androidx.appcompat.app.AppCompatActivity; import android.os.Bundle;
import android.widget.ArrayAdapter; import android.widget.ListView;


public class MainActivity extends AppCompatActivity { ListView lv1;


@Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
lv1 = findViewById(R.id.listview);
String s[] ={"Contact1","Contact2","Contact3","Contact4","Contact5","Contact6",
"Contact7","Contact8","Contact9","Contact1","Contact2","Contact3","Contact4"
,"Contact5","Contact6","Contact7","Contact8","Contact9"};


ArrayAdapter<String> adapter=new ArrayAdapter<> (this,android.R.layout.simple_list_item_1,s);


lv1.setAdapter(adapter);
}
}
```

Output:


0.00K/s
3:40 PM
5A4_LC9
Contact1
Contact2
Contact3
Contact4
Contact5
Contact6
Contact7
Contact8
Contact9
Contact1
Contact2
Contact3
Contact4

EXPERIMENT – X

AIM: Write an Android application program that demonstrates how to create Custom ListView and Custom Adapters.

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<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"

android:paddingBottom="@dimen/activity_vertical_margin"

android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"

android:paddingTop="@dimen/activity_vertical_margin"

tools:context="com.example.test.listviewwithimage.MainActivity">


<ListView android:id="@+id/list"

android:layout_width="match_parent"                    android:layout_height="wrap_content"

android:layout_marginBottom="50dp">

</ListView>

</RelativeLayout>
```

mylist.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent" android:layout_height="match_parent"

android:orientation="horizontal" >

<ImageView android:id="@+id/icon" android:layout_width="60dp"

android:layout_height="60dp" android:padding="5dp" />

<LinearLayout android:layout_width="wrap_content" android:layout_height="wrap_content"

android:orientation="vertical">

<TextView android:id="@+id/title"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:text="Medium Text" android:textStyle="bold"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout_marginLeft="10dp" android:layout_marginTop="5dp"

android:padding="2dp" android:textColor="#4d4d4d" />
```

<TextView android:id="@+id/subtitle"

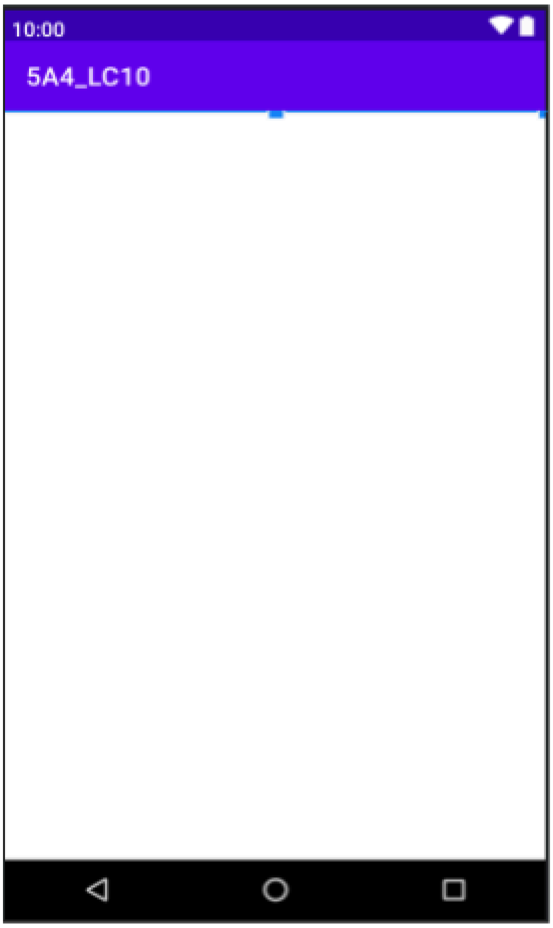
android:layout_width="wrap_content" android:layout_height="wrap_content"

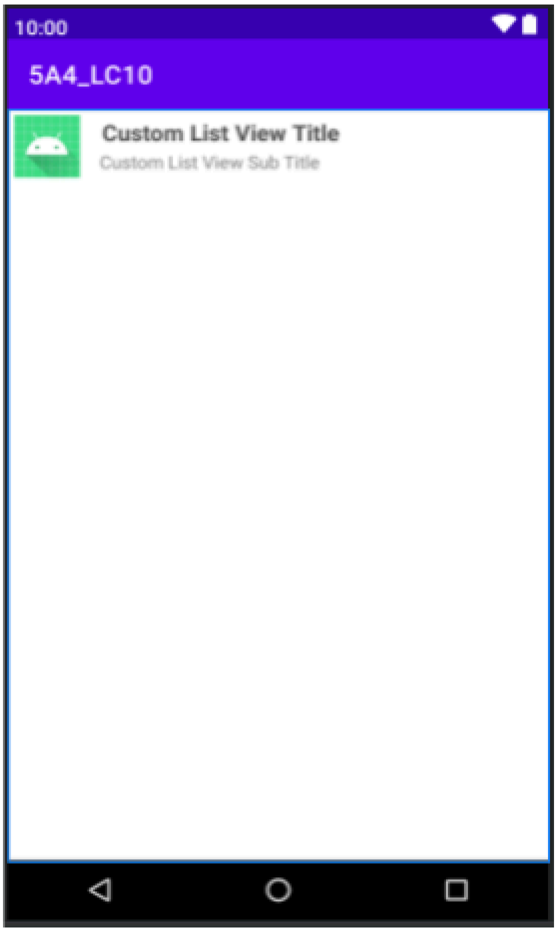
android:text="TextView" android:layout_marginLeft="10dp"/>

</LinearLayout>

</LinearLayout>

Layout Preview:





activity_main.xml **mylist.xml**

MainActivity.java:

```
package com.example.test.listviewwithimage; import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView; import android.widget.ListView; import

android.widget.Toast;

public class MainActivity extends AppCompatActivity { ListView list;

String[] maintitle ={ "Title 1","Title 2", "Title 3","Title 4", "Title 5" };
String[] subtitle ={ "Sub Title 1","Sub Title 2", "Sub Title 3","Sub Title 4", "Sub Title 5"};
Integer[] imgid={ R.drawable.download_1, R.drawable.download_2,

R.drawable.download_3, R.drawable.download_4, R.drawable.download_5 }; @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

MyListAdapter adapter=new MyListAdapter(this, maintitle, subtitle,imgid);

list=(ListView)findViewById(R.id.list);

list.setAdapter(adapter);

list.setOnItemClickListener(new AdapterView.OnItemClickListener() { @Override

public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
```

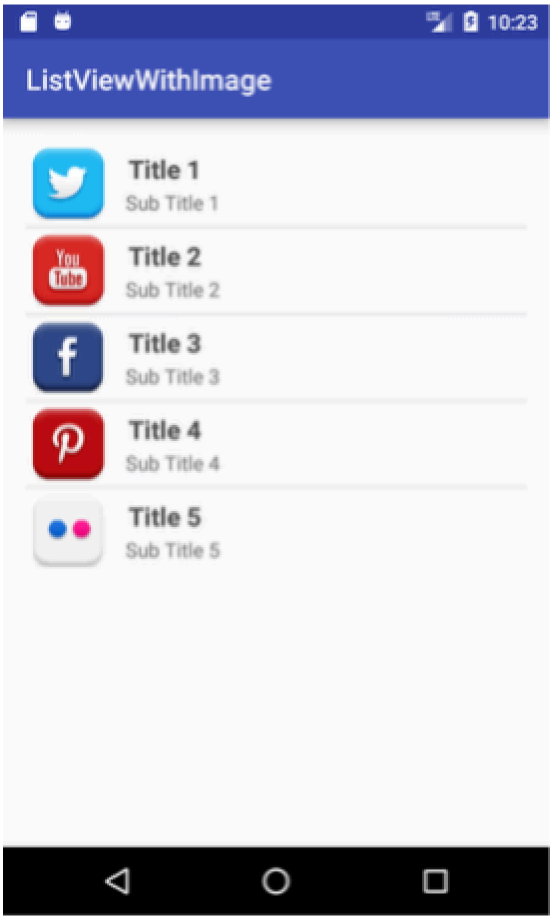
```
if(position == 0) {

Toast.makeText(getApplicationContext(),"Place Your First Option
Code",Toast.LENGTH_SHORT).show();

}

else if(position == 1) {

Toast.makeText(getApplicationContext(),"Place Your Second Option
Code",Toast.LENGTH_SHORT).show();

}
else if(position == 2) { Toast.makeText(getApplicationContext(),"Place Your Third Option
Code",Toast.LENGTH_SHORT).show();

}
else if(position == 3) { Toast.makeText(getApplicationContext(),"Place Your Forth Option
Code",Toast.LENGTH_SHORT).show();

}
else if(position == 4) { Toast.makeText(getApplicationContext(),"Place Your Fifth Option
Code",Toast.LENGTH_SHORT).show();

}

                                                                    }

                                                                   });

}

}
```

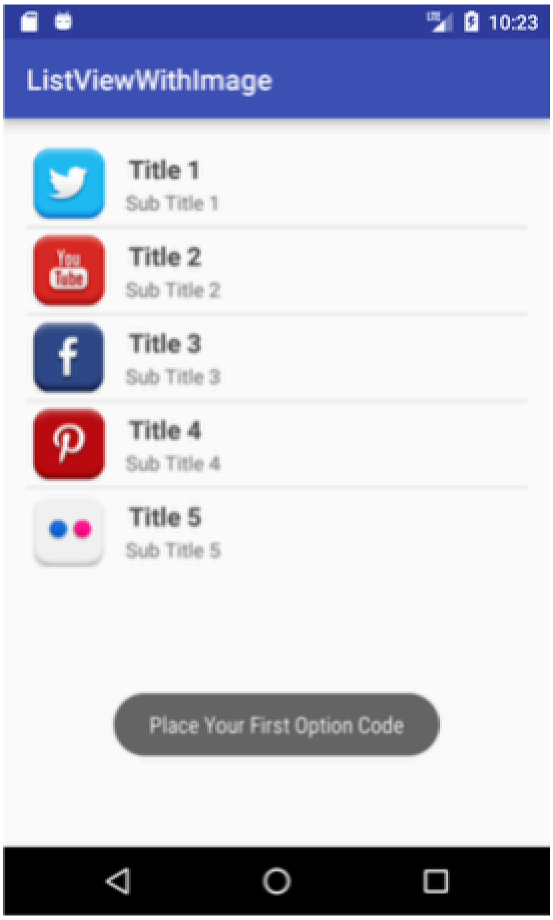
MyListView.java:

```
package com.example.test.listviewwithimage; import android.app.Activity;

import android.view.LayoutInflater; import android.view.View;

import android.view.ViewGroup; import android.widget.ArrayAdapter; import

android.widget.ImageView; import android.widget.TextView;

public class MyListAdapter extends ArrayAdapter<String> { private final Activity context;

private final String[] maintitle,subtitle; private final Integer[] imgid;

public MyListAdapter(Activity context, String[] maintitle,String[] subtitle, Integer[] imgid) {

super(context, R.layout.mylist, maintitle);

this.context=context; this.maintitle=maintitle; this.subtitle=subtitle; this.imgid=imgid;

}

public View getView(int position,View view,ViewGroup parent) { LayoutInflater

inflater=context.getLayoutInflater();

View rowView=inflater.inflate(R.layout.mylist, null,true);

TextView titleText = (TextView) rowView.findViewById(R.id.title); ImageView imageView =

(ImageView) rowView.findViewById(R.id.icon);
```

```
TextView subtitleText = (TextView) rowView.findViewById(R.id.subtitle);

titleText.setText(maintitle[position]); imageView.setImageResource(imgid[position]);

subtitleText.setText(subtitle[position]);

return rowView;

};

}
```

Output:





Before clicking After clicking

EXPERIMENT – XI

AIM: Write an Android application program that demonstrates the use of SQLite Database and Cursor.

SQLite:

Android SQLite is a very lightweight database which comes with Android OS. Android SQLite combines a clean SQL interface with a very small memory footprint and decent speed. For Android, SQLite is "baked into" the Android runtime, so every Android application can create its own SQLite databases.

Android SQLite native API is not JDBC, as JDBC might be too much overhead for a memory-limited smartphone. Once a database is created successfully its located in **data/data//databases/** accessible from Android Device Monitor.

SQLite is a typical **relational database**, containing tables (which consists of rows and columns), indexes etc. We can create our own tables to hold the data accordingly. This structure is referred to as a **schema**.

Android SQLite SQLiteOpenHelper

Android has features available to handle changing database schemas, which mostly depend on using the SQLiteOpenHelper class.

SQLiteOpenHelper is designed to get rid of two very common problems.

1. When the application runs the first time – At this point, we do not yet have a database. So we will have to create the tables, indexes, starter data, and so on.
2. When the application is upgraded to a newer schema – Our database will still be on the old schema from the older edition of the app. We will have option to alter the database schema to match the needs of the rest of the app.

SQLiteOpenHelper wraps up these logic to create and upgrade a database as per our specifications.

For that we'll need to create a custom subclass of SQLiteOpenHelper implementing at least the following three methods.

1. **Constructor** : This takes the Context (e.g., an Activity), the name of the database, an optional cursor factory (we'll discuss this later), and an integer representing the version of the database schema you are using (typically starting from 1 and increment later).

```
  public DatabaseHelper(Context context) {
      super(context, DB_NAME, null, DB_VERSION);
  }
```

2. **onCreate(SQLiteDatabase db)** : It's called when there is no database and the app needs one. It passes us a SQLiteDatabase object, pointing to a newly-created database, that we can populate with tables and initial data.

3. **onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)** : It's called when the schema version we need does not match the schema version of the database, It passes us a **SQLiteDatabase** object and the old and new version numbers. Hence we can figure out the best way to convert the database from the old schema to the new one.

We define a DBManager class to perform all database CRUD(Create, Read, Update and Delete) operations.

Opening and Closing Android SQLite Database Connection

Before performing any database operations like insert, update, delete records in a table, first open the database connection by calling **getWritableDatabase()** method as shown below:

```
public DBManager open() throws SQLException {
  dbHelper = new DatabaseHelper(context);
  database = dbHelper.getWritableDatabase();
  return this;
  }
```

The **dbHelper** is an instance of the subclass of SQLiteOpenHelper. To close a database connection

the following method is invoked.

```
 public void close() {

  dbHelper.close();
  }
```

fragment_emp_list.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent"

android:layout_height="fill_parent" >

<ListView android:id="@+id/list_view"

android:layout_width="match_parent" android:layout_height="wrap_content"

android:dividerHeight="1dp" android:padding="10dp" >

</ListView>

<TextView android:id="@+id/empty"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:layout_centerInParent="true" android:text="@string/empty_list_text" />

</RelativeLayout>
```

activity_add_record.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent" android:layout_height="match_parent"

android:orientation="vertical" android:padding="20dp" >

<EditText android:id="@+id/subject_edittext" android:layout_width="match_parent"

android:layout_height="wrap_content"><requestFocus />

</EditText>

<EditText android:id="@+id/description_edittext" android:layout_width="match_parent"

android:layout_height="wrap_content" android:hint="@string/enter_desc"

android:inputType="textMultiLine"/>

<Button

android:id="@+id/add_record" android:layout_width="wrap_content"

android:layout_height="wrap_content" android:layout_gravity="center"

android:text="@string/add_record" />

</LinearLayout>
```

activity_modify_record.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent" android:layout_height="match_parent"

android:orientation="vertical" android:padding="10dp" >

<EditText android:id="@+id/subject_edittext" android:layout_width="match_parent"

android:layout_height="wrap_content" android:layout_marginBottom="10dp" android:ems="10"

android:hint="@string/enter_title" />

<EditText android:id="@+id/description_edittext" android:layout_width="match_parent"

android:layout_height="wrap_content" android:ems="10" android:hint="@string/enter_desc"

android:inputType="textMultiLine" android:minLines="5" >

</EditText>

<LinearLayout android:layout_width="fill_parent" android:layout_height="wrap_content"
```

```
android:weightSum="2" android:gravity="center_horizontal" android:orientation="horizontal" >

<Button

android:id="@+id/btn_update" android:layout_width="wrap_content"

android:layout_height="wrap_content" android:layout_weight="1"

android:text="@string/btn_update" />

<Button

android:id="@+id/btn_delete" android:layout_width="wrap_content"

android:layout_height="wrap_content" android:layout_weight="1"

android:text="@string/btn_delete" />

</LinearLayout>

</LinearLayout>
```

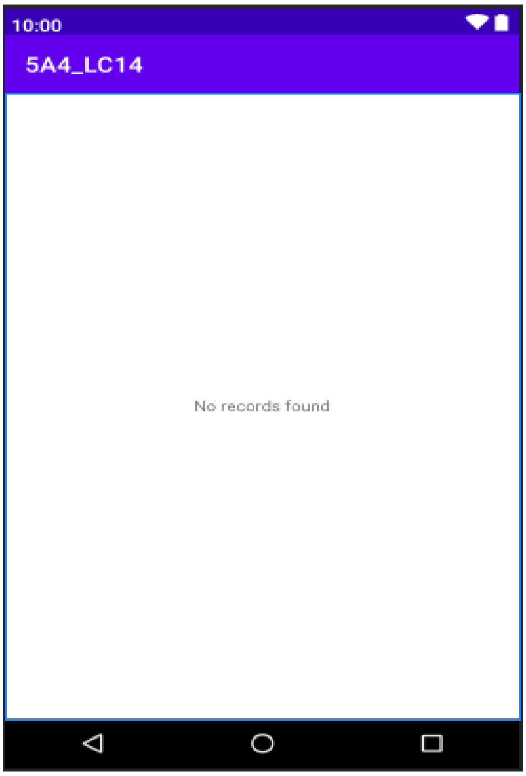
activity_view_record.xml

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent" android:layout_height="wrap_content"

android:orientation="vertical" >

<TextView android:id="@+id/id" android:layout_width="25dp"
```

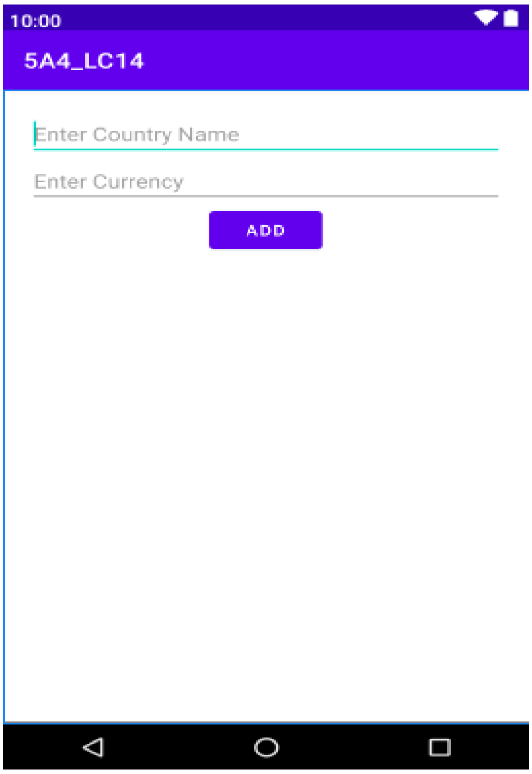
```
android:layout_height="wrap_content" android:layout_alignParentLeft="true"

android:layout_alignParentTop="true" android:layout_marginRight="6dp" android:padding="3dp"

android:visibility="visible" />

<TextView android:id="@+id/title"

android:layout_width="fill_parent" android:layout_height="wrap_content"

android:layout_marginLeft="10dp" android:layout_toRightOf="@id/id" android:maxLines="1"

android:padding="3dp" android:textSize="17sp" android:textStyle="bold" />

<TextView android:id="@+id/desc"

android:layout_width="fill_parent" android:layout_height="wrap_content"

android:layout_below="@id/title" android:layout_marginLeft="10dp"

android:layout_toRightOf="@id/id" android:ellipsize="end" android:maxLines="2"

android:padding="3dp"
```

```
android:visibility="visible" />

</RelativeLayout>
```

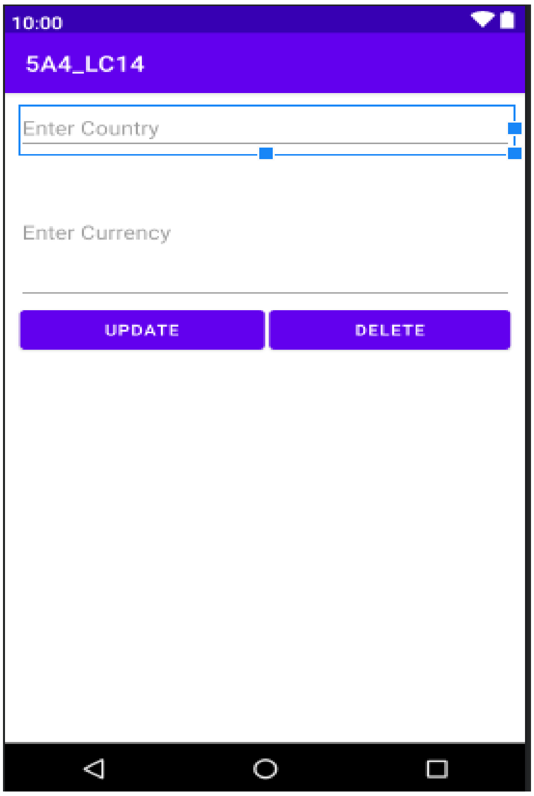
Layout Preview:



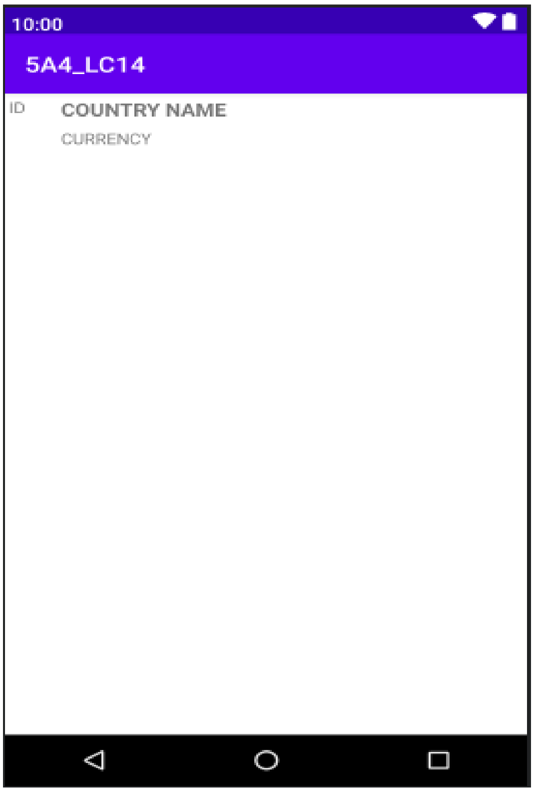
fragment_emp_list.xml



activity_add_record.xml



activity_modify_record.xml



activity_view_record.xml

CountryListActivity.java:

```
package com.uday.android.a5a4_lc11; import android.content.Intent;

import android.database.Cursor; import android.os.Bundle;

import android.support.v4.widget.SimpleCursorAdapter; import

android.support.v7.app.ActionBarActivity; import android.view.Menu;

import android.view.MenuItem; import android.view.View;

import android.widget.AdapterView; import android.widget.ListView; import

android.widget.TextView;


public class CountryListActivity extends ActionBarActivity { private DBManager;

private ListView;

private SimpleCursorAdapter adapter;

final String[] from = new String[] { DatabaseHelper._ID, DatabaseHelper.SUBJECT,

DatabaseHelper.DESC };

final int[] to = new int[] { R.id.id, R.id.title, R.id.desc }; @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.fragment_emp_list); dbManager = new DBManager(this);
```

```
dbManager.open();

Cursor = dbManager.fetch();

listView = (ListView) findViewById(R.id.list_view);

listView.setEmptyView(findViewById(R.id.empty));

adapter = new SimpleCursorAdapter(this, R.layout.activity_view_record, cursor,      from, to, 0);

adapter.notifyDataSetChanged();

listView.setAdapter(adapter);

// OnCLickListiner For List Items

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() { @Override

public void onItemClick(AdapterView<?> parent, View, int position, long viewId) { TextView

idTextView = (TextView) view.findViewById(R.id.id);

TextView titleTextView = (TextView) view.findViewById(R.id.title); TextView descTextView =

(TextView) view.findViewById(R.id.desc); String id = idTextView.getText().toString();

String title = titleTextView.getText().toString(); String desc = descTextView.getText().toString();

Intent modify_intent = new Intent(getApplicationContext(), ModifyCountryActivity.class);

modify_intent.putExtra("title", title);

modify_intent.putExtra("desc", desc); modify_intent.putExtra("id", id);

startActivity(modify_intent);

}

});

}
```

```
@Override

public boolean onCreateOptionsMenu(Menu menu) { getMenuInflater().inflate(R.menu.main,

menu); return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) { int id = item.getItemId();

if (id == R.id.add_record) {

Intent add_mem = new Intent(this, AddCountryActivity.class); startActivity(add_mem);

}

return super.onOptionsItemSelected(item);

}

}
```

DatabaseHelper.java:

```
package com.uday.android.a5a4_lc11; import android.content.Context;

import android.database.sqlite.SQLiteDatabase; import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper { public static final String

TABLE_NAME = "COUNTRIES"; public static final String _ID = "_id";

public static final String SUBJECT = "subject"; public static final String DESC = "description";

static final String DB_NAME = "COUNTRIES.DB";
```

```
static final int DB_VERSION = 1;

private static final String CREATE_TABLE = "create table " + TABLE_NAME + "(" + _ID

+ " INTEGER PRIMARY KEY AUTOINCREMENT, " + SUBJECT + " TEXT NOT NULL, " +
D ESC + " TEXT);";

public DatabaseHelper(Context context) { super(context, DB_NAME, null, DB_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) { db.execSQL(CREATE_TABLE);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { db.execSQL("DROP

TABLE IF EXISTS " + TABLE_NAME);

onCreate(db);

}

}
```

DBManager.java

```
package com.uday.android.a5a4_lc11; import android.content.ContentValues; import

android.content.Context; import android.database.Cursor;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase; public class DBManager {

private DatabaseHelper dbHelper;
```

```
private Context context;

private SQLiteDatabase database; public DBManager(Context c) {

context = c;

}

public DBManager open() throws SQLException { dbHelper = new DatabaseHelper(context);

database = dbHelper.getWritableDatabase(); return this;

}

public void close() { dbHelper.close();

}

public void insert(String name, String desc) { ContentValues contentValue = new ContentValues();

contentValue.put(DatabaseHelper.SUBJECT, name); contentValue.put(DatabaseHelper.DESC,

desc);

database.insert(DatabaseHelper.TABLE_NAME, null, contentValue);

}

public Cursor fetch() {

String[] columns = new String[] { DatabaseHelper._ID, DatabaseHelper.SUBJECT,
DatabaseHelper.D ESC };

Cursor cursor = database.query(DatabaseHelper.TABLE_NAME, columns, null, null, null, null,

null); if (cursor != null) {

cursor.moveToFirst();

}
```

return cursor;

}

public int update(long _id, String name, String desc) { ContentValues contentValues = new

ContentValues(); contentValues.put(DatabaseHelper.SUBJECT, name);

contentValues.put(DatabaseHelper.DESC, desc);

int i = database.update(DatabaseHelper.TABLE_NAME, contentValues, DatabaseHelper._ID + " = " +

_id, null);

return i;

}

public void delete(long _id) {

database.delete(DatabaseHelper.TABLE_NAME, DatabaseHelper._ID + "=" + _id, null);

}

}

ModifyCountryActivity.java package com.uday.android.a5a4_lc11; import

android.app.Activity;

import android.content.Intent; import android.os.Bundle; import android.view.View;

import android.view.View.OnClickListener; import android.widget.Button;

import android.widget.EditText;

public class ModifyCountryActivity extends Activity implements OnClickListener { private

EditText titleText;

private Button updateBtn, deleteBtn;

```
private EditText descText; private long _id;

private DBManager dbManager; @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setTitle("Modify Record"); setContentView(R.layout.activity_modify_record); dbManager = new

DBManager(this); dbManager.open();

titleText = (EditText) findViewById(R.id.subject_edittext); descText = (EditText)

findViewById(R.id.description_edittext); updateBtn = (Button) findViewById(R.id.btn_update);

deleteBtn = (Button) findViewById(R.id.btn_delete);

Intent intent = getIntent();

String id = intent.getStringExtra("id"); String name = intent.getStringExtra("title"); String desc =

intent.getStringExtra("desc");

_id = Long.parseLong(id); titleText.setText(name); descText.setText(desc);

updateBtn.setOnClickListener(this); deleteBtn.setOnClickListener(this);

}

@Override

public void onClick(View v) {
```

```
switch (v.getId()) { case R.id.btn_update:

String title = titleText.getText().toString(); String desc = descText.getText().toString();

dbManager.update(_id, title, desc); this.returnHome();

break;

case R.id.btn_delete: dbManager.delete(_id); this.returnHome(); break;

}

}

public void returnHome() {

Intent home_intent = new Intent(getApplicationContext(), CountryListActivity.class)

.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP); startActivity(home_intent);

}

}
```

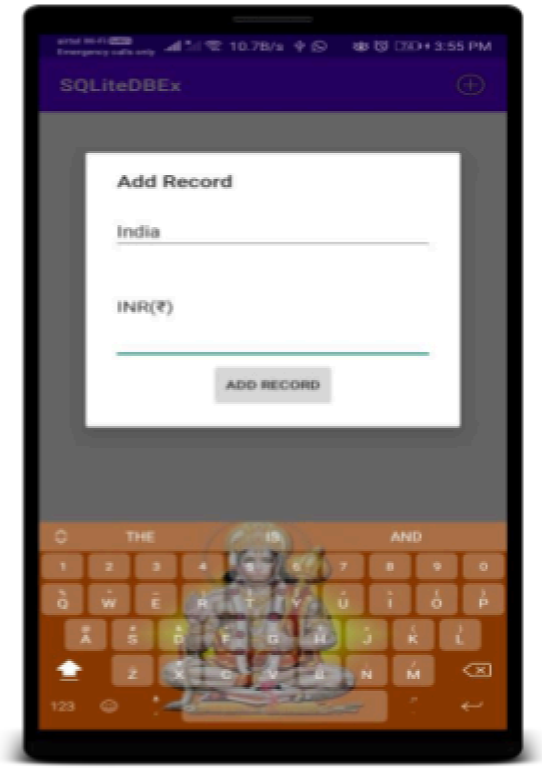
AddCountryActivity.java

```
package com.uday.android.a5a4_lc11; import android.app.Activity;

import android.content.Intent; import android.os.Bundle; import android.view.View;

import android.view.View.OnClickListener; import android.widget.Button;

import android.widget.EditText;

public class AddCountryActivity extends Activity implements OnClickListener { private Button

addTodoBtn;

private EditText subjectEditText; private EditText descEditText; private DBManager dbManager;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setTitle("Add Record"); setContentView(R.layout.activity_add_record);

subjectEditText = (EditText) findViewById(R.id.subject_edittext); descEditText = (EditText)

findViewById(R.id.description_edittext); addTodoBtn = (Button) findViewById(R.id.add_record);

dbManager = new DBManager(this);

dbManager.open(); addTodoBtn.setOnClickListener(this);     }
```

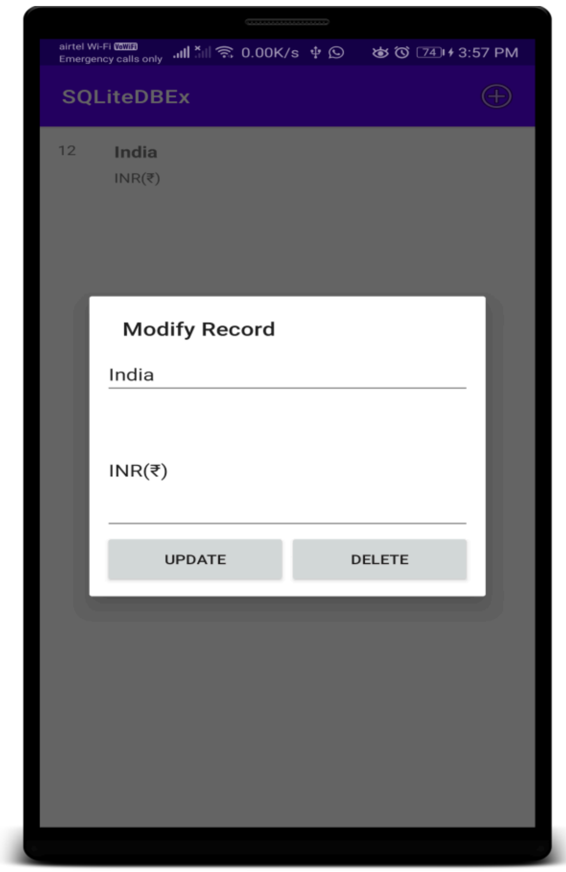
```
@Override

public void onClick(View v) { switch (v.getId()) {

case R.id.add_record:

final String name = subjectEditText.getText().toString(); final String desc =

descEditText.getText().toString(); dbManager.insert(name, desc);

Intent main = new Intent(AddCountryActivity.this, CountryListActivity.class)

.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP); startActivity(main);

break;

}

}

}
```

OUTPUTS:

fragment_emp_list.xml



activity_add_record.xml



activity_modify_record.xml

activity_view_record.xml

EXPERIMENT – XII

AIM: **Write an Android application program that demonstrates the use AsyncTask. activity_main.xml:**

```
<?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">
<TextView android:layout_width="match_parent" android:layout_height="wrap_content"
android:text="Async Task" android:gravity="center" android:textSize="40sp"
/>
<ImageView android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/iv"/>
</LinearLayout>
```

Layout Preview:

MainActivity.java:

package com.uday.android.a5a4_lc12;

import androidx.appcompat.app.AppCompatActivity; import android.graphics.Bitmap;

import android.graphics.BitmapFactory; import android.os.AsyncTask;

import android.os.Bundle;

import android.widget.ImageView; import android.widget.Toast;

import java.io.BufferedInputStream; import java.io.IOException;

import java.io.InputStream;

import java.net.MalformedURLException; import java.net.URL;

import java.net.URLConnection; import java.nio.Buffer;

public class MainActivity extends AppCompatActivity { ImageView iv;

String

```
url1="https://originalshrewsbury.co.uk/sites/default/files/styles/original_flyer/public/paddington-
2.jpg?itok=dlzeS1un";
@Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main); iv=findViewById(R.id.iv);
new AsyncExmp().execute();

}
public class AsyncExmp extends AsyncTask<String,Void, Bitmap>{ @Override
protected void onPreExecute() { super.onPreExecute();
Toast.makeText(MainActivity.this, "This is onPreExecute", Toast.LENGTH_SHORT).show();
}

@Override
protected Bitmap doInBackground(String... strings) { Bitmap bm=null;
try {
URL url=new URL(url1);
URLConnection urlConnection=url.openConnection(); urlConnection.connect();
InputStream inputStream=urlConnection.getInputStream(); BufferedInputStream bis=new
BufferedInputStream(inputStream); bm= BitmapFactory.decodeStream(bis);
bis.close(); inputStream.close();
} catch (MalformedURLException e) { e.printStackTrace();
} catch (IOException e) { e.printStackTrace();
}
return bm;
}
```

```
@Override
protected void onPostExecute(Bitmap bitmap) { super.onPostExecute(bitmap);
iv.setImageBitmap(bitmap);
}
}
}
```

Output:

EXPERIMENT – XIII

AIM: Write an Android application program that demonstrates Notifications

What is Notification?

A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification.

Appearances on a device

N… ns and formats, such as an icon in the status bar, a … as a badge on the app's icon, and on paired …

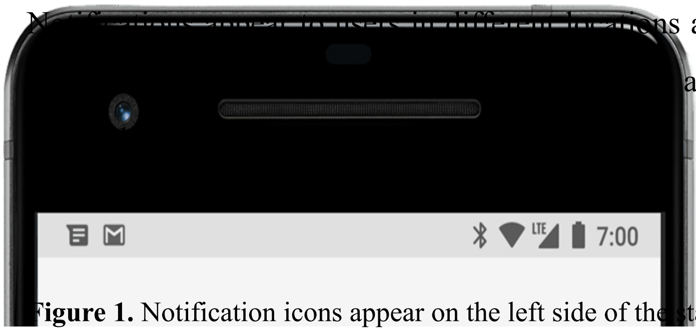
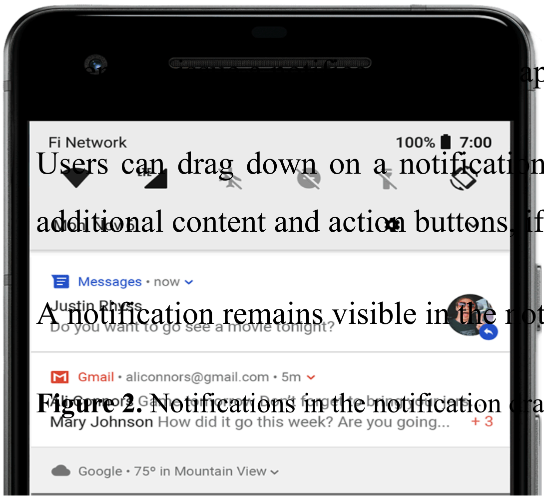


Figure 1. Notification icons appear on the left side of the status bar

Status bar and notification drawer



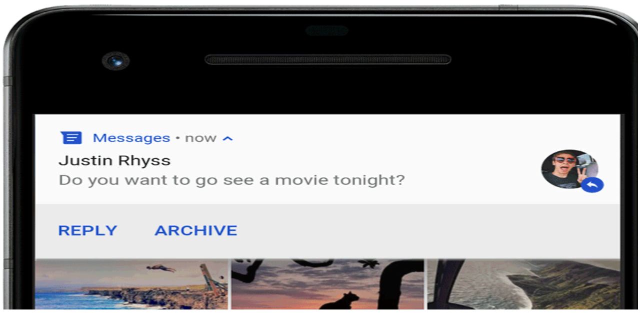
… ppears as an icon in the status bar.

Users can drag down on a notification in the drawer to reveal the expanded view, which shows additional content and action buttons, if provided.

A notification remains visible in the notification drawer until dismissed by the app or the user.

Figure 2. Notifications in the notification drawer

Heads-up Notification



Beginning with Android 5.0, notifications can briefly appear in a floating window called a *heads-up notification*. This behavior is normally for important notifications that the user should know about immediately, and it appears only if the device is unlocked.

Figure 3. A heads-up notification appears in front of the foreground app

The heads-up notification appears the moment your app issues the notification and it disappears after a moment, but remains visible in the notification drawer as usual.

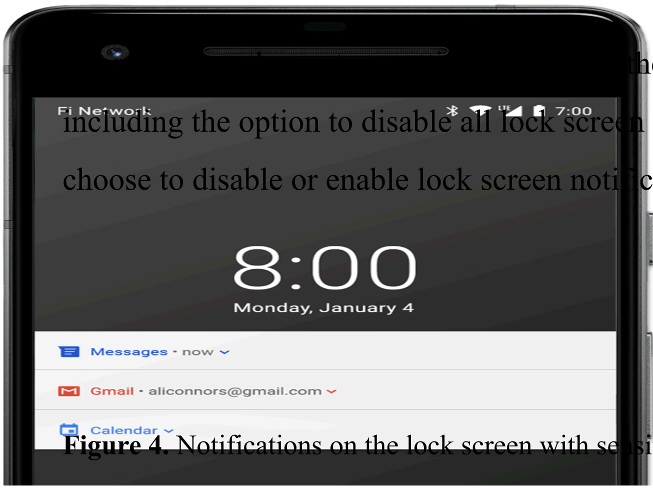
Example conditions that might trigger heads-up notifications include the following:

- The user's activity is in fullscreen mode (the app uses fullScreenIntent).
- The notification has high priority and uses ringtones or vibrations on devices running Android 7.1 (API level 25) and lower.
- The notification channel has high importance on devices running Android 8.0 (API level 26) and higher.

Lock screen

Beginning with Android 5.0, notifications can appear on the lock screen.

You can programmatically set the level of detail visible in notifications posted by your app on a secure lock screen, or even whether the notification will show on the lock screen at all.



he level of detail visible in lock screen notifications, including the option to disable all lock screen notifications. Starting with Android 8.0, users can choose to disable or enable lock screen notifications for each notification channel.

Figure 4. Notifications on the lock screen with sensitive content hidden

App icon badge

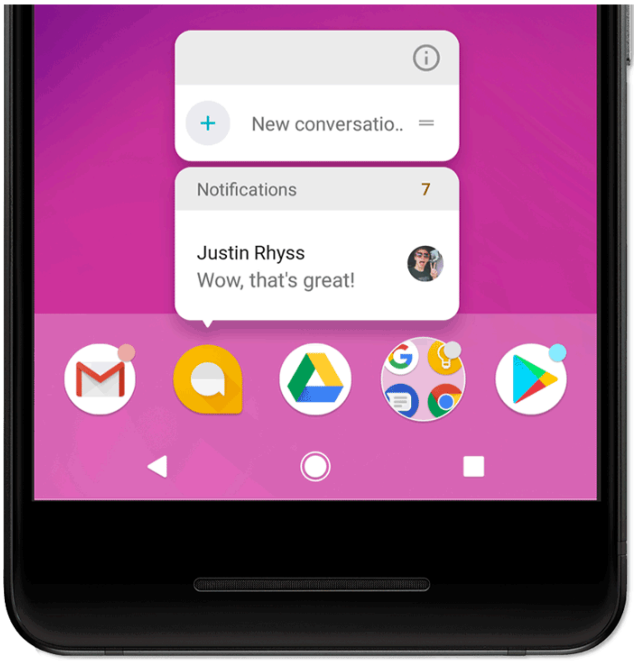
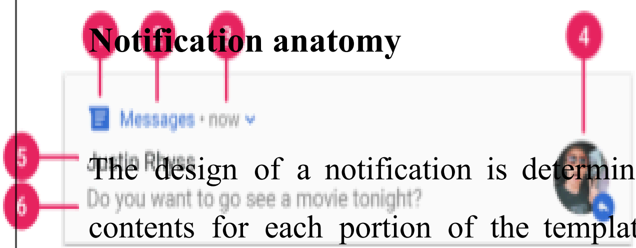


Figure 5. Notification badges and the long-press menu

In supported launchers on devices running Android 8.0 (API level 26) and higher, app icons indicate new notifications with a colored "badge" (also known as a "notification dot") on the corresponding app launcher icon.

Users can long-press on an app icon to see the notifications for that app. Users can then dismiss or act on notifications from that menu, similar to the notification drawer.

Notification anatomy



The design of a notification is determined by system templates—your app simply defines the contents for each portion of the template. Some details of the notification appear only in the expanded view.

Figure 6. A notification with basic details

The most common parts of a notification are indicated in figure 7 as follows:

1. Small icon: This is required and set with setSmallIcon().

2. App name: This is provided by the system.

3. Time stamp: This is provided by the system but you can override with setWhen() or hide it with setShowWhen(false).

4. Large icon: This is optional (usually used only for contact photos; do not use it for your app icon) and set with setLargeIcon().

5. Title: This is optional and set with setContentTitle().

6. Text: This is optional and set with setContentText().

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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">

<Button
android:id="@+id/notify" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="@string/notify_me"
app:layout_constraintBottom_toTopOf="@+id/update"
app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<Button
android:id="@+id/update" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="@string/update_me"
app:layout_constraintBottom_toTopOf="@+id/cancel"
app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/notify" />

<Button
android:id="@+id/cancel" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="@string/cancel_me"
app:layout_constraintBottom_toBottomOf="parent"
```

app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/update" />

</android.support.constraint.ConstraintLayout>

Layout Preview:

MainActivity.java:

```
package com.uday.android.a5a4_lc13; import android.app.NotificationChannel; import
android.app.NotificationManager; import android.app.PendingIntent;
import android.content.BroadcastReceiver; import android.content.Context;
import android.content.Intent; import android.content.IntentFilter; import android.graphics.Bitmap;
import android.graphics.BitmapFactory; import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat; import
android.support.v7.app.AppCompatActivity; import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
private static final String ACTION_UPDATE_NOTIFICATION =
"com.uday.android.a5a4_lc13.ACTION_UPDATE_NOTIFICATION";
private static final String PRIMARY_CHANNEL_ID = "primary_notification_channel";
private static final int NOTIFICATION_ID = 0; private Button button_notify;
private Button button_cancel; private Button button_update;
```

```
private NotificationManager mNotifyManager;

private NotificationReceiver mReceiver = new NotificationReceiver(); @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main); createNotificationChannel();

registerReceiver(mReceiver,

new IntentFilter(ACTION_UPDATE_NOTIFICATION)); button_notify =

findViewById(R.id.notify); button_notify.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) { sendNotification();

}

});

button_update = (Button) findViewById(R.id.update); button_update.setOnClickListener(new

View.OnClickListener() {

@Override

public void onClick(View view) {

// Update the notification. updateNotification();

}

});

button_cancel = (Button) findViewById(R.id.cancel); button_cancel.setOnClickListener(new

View.OnClickListener() {
```

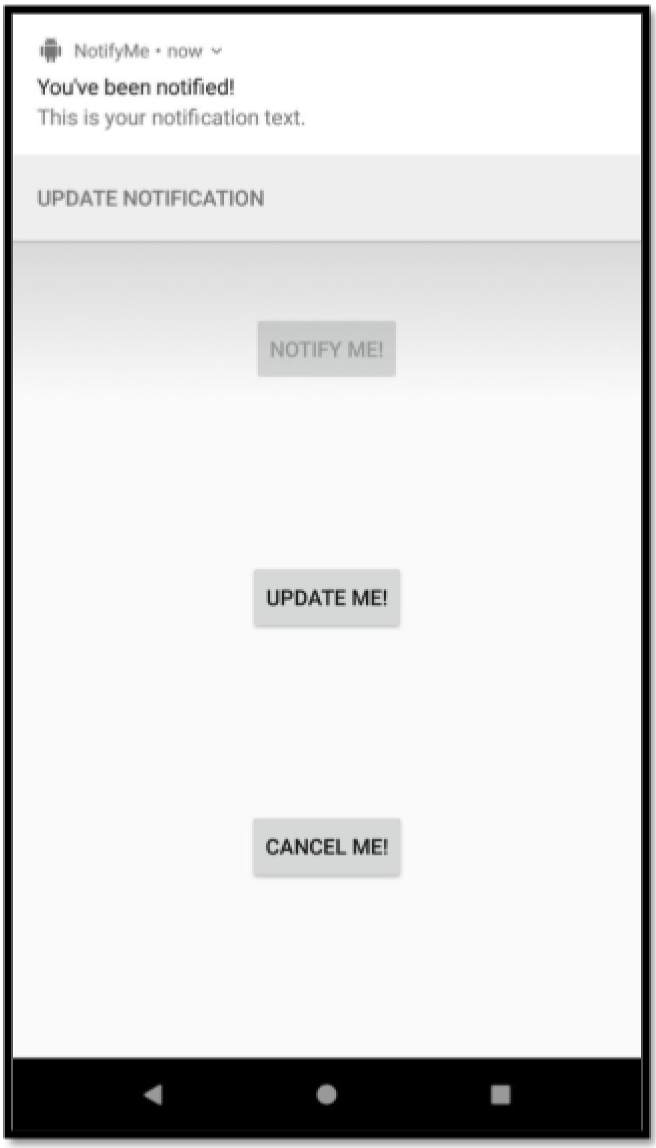
```
@Override

public void onClick(View view) { cancelNotification();

}

});

setNotificationButtonState(true, false, false);

}

@Override

protected void onDestroy() { unregisterReceiver(mReceiver); super.onDestroy();

}

public void createNotificationChannel() {

mNotifyManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE); if

(android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {

NotificationChannel = new NotificationChannel

(PRIMARY_CHANNEL_ID, getString(R.string.notification_channel_name),

NotificationManager.IMPORTANCE_HIGH);

notificationChannel.enableLights(true);
notificationChannel.setLightColor(Color.RED); notificationChannel.enableVibration(true);

notificationChannel.setDescription(getString(R.string.notification_channel_description));

mNotifyManager.createNotificationChannel(notificationChannel);

}

}

public void sendNotification() {
```

```
Intent updateIntent = new Intent(ACTION_UPDATE_NOTIFICATION); PendingIntent
updatePendingIntent = PendingIntent.getBroadcast(this,
NOTIFICATION_ID, updateIntent, PendingIntent.FLAG_ONE_SHOT);
NotificationCompat.Builder notifyBuilder = getNotificationBuilder();
// Add the action button using the pending intent. notifyBuilder.addAction(R.drawable.ic_update,
getString(R.string.update), updatePendingIntent); mNotifyManager.notify(NOTIFICATION_ID,
notifyBuilder.build()); setNotificationButtonState(false, true, true);
}
private NotificationCompat.Builder getNotificationBuilder() { Intent notificationIntent = new
Intent(this, MainActivity.class);
PendingIntent notificationPendingIntent = PendingIntent.getActivity (this, NOTIFICATION_ID,
notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT); NotificationCompat.Builder notifyBuilder = new
NotificationCompat
.Builder(this, PRIMARY_CHANNEL_ID)
.setContentTitle(getString(R.string.notification_title))
.setContentText(getString(R.string.notification_text))
.setSmallIcon(R.drawable.ic_android)
.setAutoCancel(true).setContentIntent(notificationPendingIntent)
.setPriority(NotificationCompat.PRIORITY_HIGH)
.setDefaults(NotificationCompat.DEFAULT_ALL); return notifyBuilder;
}
public void updateNotification() {
```

```
Bitmap androidImage = BitmapFactory

.decodeResource(getResources(), R.drawable.mascot_1); NotificationCompat.Builder

notifyBuilder = getNotificationBuilder(); notifyBuilder.setStyle(new

NotificationCompat.BigPictureStyle()

.bigPicture(androidImage)

.setBigContentTitle(getString(R.string.notification_updated)));

mNotifyManager.notify(NOTIFICATION_ID, notifyBuilder.build());

setNotificationButtonState(false, false, true);

}

public void cancelNotification() { mNotifyManager.cancel(NOTIFICATION_ID);

setNotificationButtonState(true, false, false);

}

void setNotificationButtonState(Boolean isNotifyEnabled, Boolean

isUpdateEnabled, Boolean isCancelEnabled) { button_notify.setEnabled(isNotifyEnabled);

button_update.setEnabled(isUpdateEnabled); button_cancel.setEnabled(isCancelEnabled);

}

public class NotificationReceiver extends BroadcastReceiver { public NotificationReceiver() {

}

@Override

public void onReceive(Context, Intent intent) {

// Update the notification. updateNotification();
```

```
}

}

}
```

OUTPUT:


NotifyMe • now
You've been notified!
This is your notification text.
UPDATE NOTIFICATION
NOTIFY ME!
UPDATE ME!
CANCEL ME!

EXPERIMENT - XIV

AIM: Write an Android application program that demonstrates Shared Preferences. Shared Preferences:

Shared preferences allow you to store small amounts of primitive data as key/value pairs in a file on the device. To get a handle to a preference file, and to read, write, and manage preference data, use
the `SharedPreferences` class. The Android framework manages the shared preferences file itself. The file is accessible to all the components of your app, but it is not accessible to other apps.
For managing large amounts of data, use an SQLite database.

Shared preferences vs. saved instance state:

Shared preferences	**Saved instance state**
Persists across user sessions, even if your app is stopped and restarted, or if the device is rebooted.	Preserves state data across activity instances in the same user session.
Used for data that should be remembered across user sessions, such as a user's preferred settings or their game score.	Used for data that should not be remembered across sessions, such as the currently selected tab, or any current state of an activity.
Represented by a small number of key/value pairs.	Represented by a small number of key/value pairs.
Data is private to the app.	Data is private to the app.
Common use is to store user preferences.	Common use is to recreate state after the device has been rotated.

Note: The `SharedPreference` APIs are different from the `Preference` APIs. The `Preference` APIs can be used to build a user interface for a settings page, and they use shared preferences for their underlying implementation.

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.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"

android:padding="16dp" tools:context=".MainActivity">

<TextView android:id="@+id/count_textview" android:layout_width="0dp"

android:layout_height="0dp"

android:background="@color/default_background" android:gravity="center"

android:text="@string/default_count" android:textColor="@android:color/white"

android:textSize="112sp"

app:layout_constraintBottom_toTopOf="@+id/guideline_upper"

app:layout_constraintDimensionRatio="1:1" app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent"/>


<android.support.constraint.Guideline
```

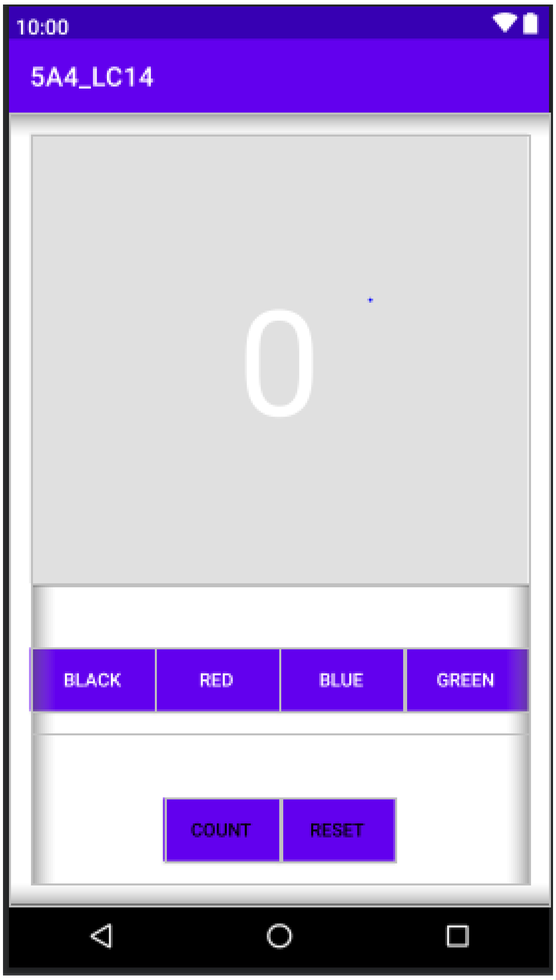
```
android:id="@+id/guideline_upper" android:layout_width="wrap_content"

android:layout_height="wrap_content" android:orientation="horizontal"

app:layout_constraintGuide_end="120dp"/>


<Button

android:id="@+id/black_background_button" style="@style/AppTheme.Button.Colored"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:background="@android:color/black" android:onClick="changeBackground"

android:text="@string/black_button"

app:layout_constraintBottom_toTopOf="@+id/guideline_lower"

app:layout_constraintEnd_toStartOf="@+id/red_background_button"

app:layout_constraintHorizontal_chainStyle="packed"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="@+id/guideline_upper"/>


<Button

android:id="@+id/red_background_button" style="@style/AppTheme.Button.Colored"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:background="@color/red_background"
```

```
android:onClick="changeBackground" android:text="@string/red_button"

app:layout_constraintBottom_toTopOf="@+id/guideline_lower"

app:layout_constraintEnd_toStartOf="@+id/blue_background_button"

app:layout_constraintStart_toEndOf="@+id/black_background_button"

app:layout_constraintTop_toTopOf="@+id/guideline_upper"/>


<Button

android:id="@+id/blue_background_button" style="@style/AppTheme.Button.Colored"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:background="@color/blue_background" android:onClick="changeBackground"

android:text="@string/blue_button"

app:layout_constraintBottom_toTopOf="@+id/guideline_lower"

app:layout_constraintEnd_toStartOf="@+id/green_background_button"

app:layout_constraintStart_toEndOf="@+id/red_background_button"

app:layout_constraintTop_toTopOf="@+id/guideline_upper"/>

<Button

android:id="@+id/green_background_button" style="@style/AppTheme.Button.Colored"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:background="@color/green_background" android:onClick="changeBackground"
```

```
android:text="@string/green_button"

app:layout_constraintBottom_toTopOf="@+id/guideline_lower"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toEndOf="@+id/blue_background_button"

app:layout_constraintTop_toTopOf="@+id/guideline_upper"/>

<android.support.constraint.Guideline android:id="@+id/guideline_lower"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:orientation="horizontal" app:layout_constraintGuide_end="56dp"/>

<Button

android:id="@+id/count_button" style="@style/AppTheme.Button"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:layout_marginEnd="16dp" android:layout_marginRight="16dp"

android:onClick="countUp" android:text="@string/count_button"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toStartOf="@+id/reset_button"

app:layout_constraintHorizontal_chainStyle="packed"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/guideline_lower"/>

<Button
```

```
android:id="@+id/reset_button" style="@style/AppTheme.Button"

android:layout_width="wrap_content" android:layout_height="wrap_content"

android:onClick="reset" android:text="@string/reset_button"

app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toEndOf="@+id/count_button"

app:layout_constraintTop_toBottomOf="@+id/guideline_lower"/>

</android.support.constraint.ConstraintLayout>
```

Layout Preview:



MainActivity.java:

package com.example.android.hellosharedprefs; import android.content.SharedPreferences;

import android.graphics.drawable.ColorDrawable; import

android.support.v4.content.ContextCompat; import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View; import android.widget.TextView;

public class MainActivity extends AppCompatActivity { private int mCount,mColor;;

private TextView mShowCountTextView; private final String COUNT_KEY = "count"; private

final String COLOR_KEY = "color"; private SharedPreferences mPreferences;

private String sharedPrefFile = "com.example.android.hellosharedprefs"; @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main); mShowCountTextView =

findViewById(R.id.count_textview);

mColor = ContextCompat.getColor(this, R.color.default_background); mPreferences =

getSharedPreferences(sharedPrefFile, MODE_PRIVATE); mCount =

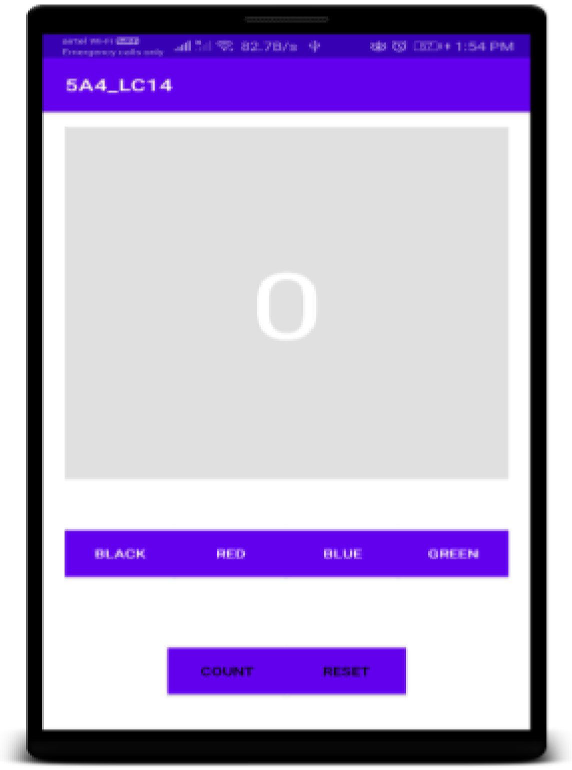
mPreferences.getInt(COUNT_KEY, 0); mShowCountTextView.setText(String.format("%s",

mCount));

mColor = mPreferences.getInt(COLOR_KEY, mColor);

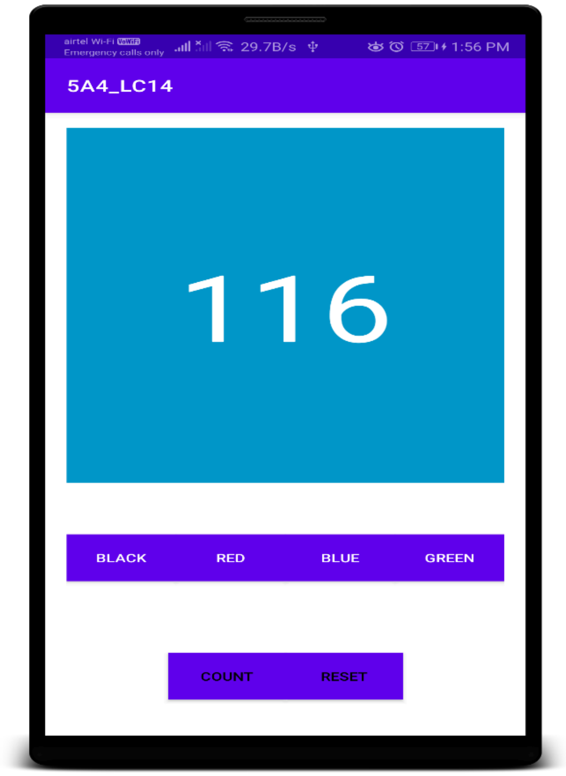
```
mShowCountTextView.setBackgroundColor(mColor);

}

public void changeBackground(View view) {

int color = ((ColorDrawable) view.getBackground()).getColor();

mShowCountTextView.setBackgroundColor(color);

mColor = color;

}

public void countUp(View view) { mCount++;

mShowCountTextView.setText(String.format("%s", mCount));

}

public void reset(View view) { mCount = 0;

mShowCountTextView.setText(String.format("%s", mCount)); mColor =

ContextCompat.getColor(this, R.color.default_background);

mShowCountTextView.setBackgroundColor(mColor); SharedPreferences.Editor preferencesEditor

= mPreferences.edit(); preferencesEditor.clear();

preferencesEditor.apply(); } @Override

protected void onPause() { super.onPause();

SharedPreferences.Editor preferencesEditor = mPreferences.edit();

preferencesEditor.putInt(COUNT_KEY, mCount); preferencesEditor.putInt(COLOR_KEY,

mColor); preferencesEditor.apply(); } }
```

OUTPUT:

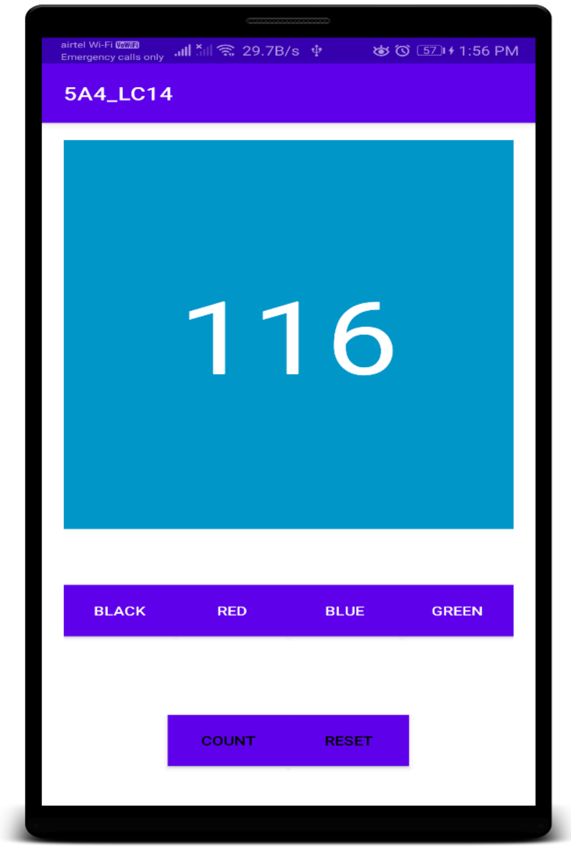


color

When app is opened for the first time



App after clicking count for 116 times and changing color



App after reopening by applying changes in the screen

EXPERIMENT – XV

AIM: **Write an Android application program that connect to the internet, gets JSON data and displays the result in UI by parsing JSON data.**

JavaScript Object Notation(JSON):

- JSON is often used when data is sent from a server to a web page
- JSON is "self-describing" and easy to understand
- JSON is a lightweight format for storing and transporting data

JSON Syntax Rules:

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

BOOKSAPI URL: “https://www.googleapis.com/books/v1/volumes?q=<any book name>”

JSON Data:

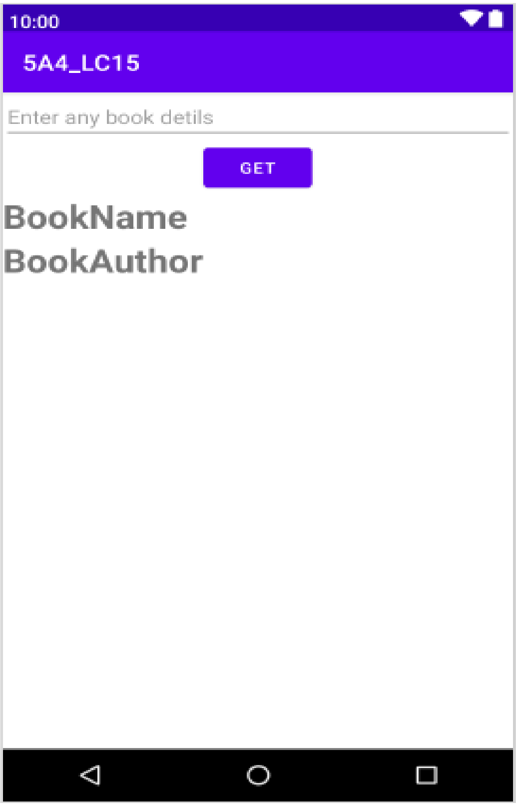
{ "kind": "books#volumes", "totalItems": 2370,

"items": [{
"kind": "books#volume", "id": "96gMhjaAviMC", "etag": "9H4ISDT5PhY",
"selfLink": "https://www.googleapis.com/books/v1/volumes/96gMhjaAviMC", "volumeInfo": {
"title": "Core Java(Tm) Volume 1-Fundamentals, 8/E", "authors": [
"Horstmann"] }
}] }

activity_main.xml:

```
<?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"
tools:context=".MainActivity" android:orientation="vertical">
<EditText android:layout_width="match_parent" android:layout_height="wrap_content"
android:id="@+id/getbook" android:hint="Enter any book detils"/>
<Button
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/get" android:layout_gravity="center" android:text="GET"/>
<TextView android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/tx1" android:text="BookName" android:textStyle="bold"
android:textSize="30sp"/>
<TextView android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/tx2" android:text="BookAuthor" android:textStyle="bold"
android:textSize="30sp"/>
</LinearLayout>
```

Layout Preview:



MainActivity.java:

package com.uday.android.a5a4_lc15;

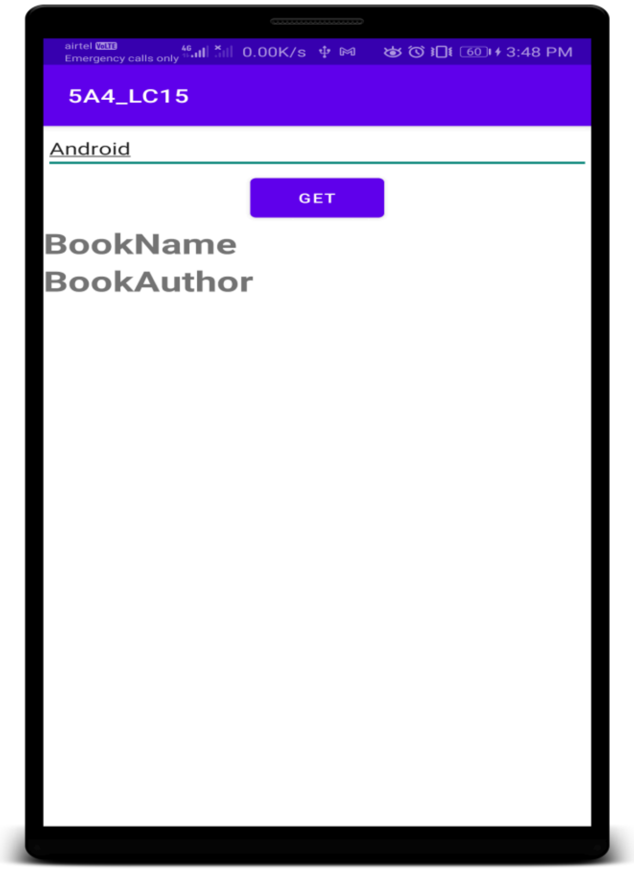
import android.os.AsyncTask; import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText;
import android.widget.TextView;

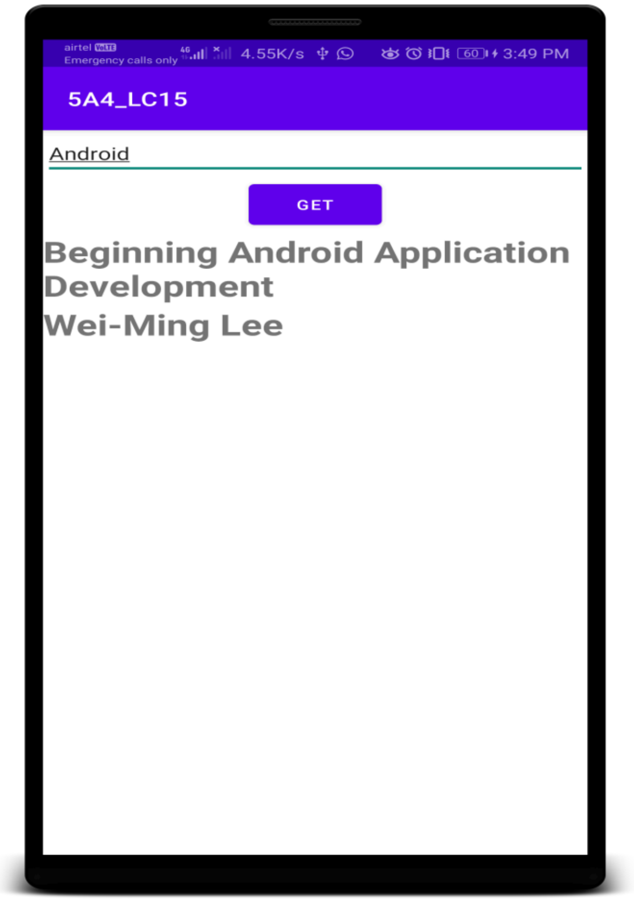
import androidx.appcompat.app.AppCompatActivity; import org.json.JSONArray;
import org.json.JSONException; import org.json.JSONObject; import java.io.IOException; import java.io.InputStream;
import java.net.HttpURLConnection; import java.net.MalformedURLException;

```
import java.net.URL; import java.util.Scanner;
public class MainActivity extends AppCompatActivity { EditText ed1;
Button b1; TextView t1,t2;
String u="https://www.googleapis.com/books/v1/volumes?q="; @Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main); ed1=findViewById(R.id.getbook);
b1=findViewById(R.id.get); t1=findViewById(R.id.tx1); t2=findViewById(R.id.tx2);
b1.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
MyAynctask myAynctask=new MyAynctask(); myAynctask.execute();
}
});
}
public class MyAynctask extends AsyncTask<String,Void,String>{ @Override
protected String doInBackground(String... strings) { try {
URL urlobject=new URL(u+ed1.getText().toString());
HttpURLConnection httpURLConnection= (HttpURLConnection) urlobject.openConnection();
httpURLConnection.connect();
InputStream inputStream=httpURLConnection.getInputStream(); Scanner scanner=new
Scanner(inputStream); scanner.useDelimiter("\\A");
if(scanner.hasNext()){ return scanner.next();
}
```

```
else return null;
} catch (MalformedURLException e) { e.printStackTrace();
} catch (IOException e) { e.printStackTrace();
}
return null;
}
@Override
protected void onPostExecute(String s) { super.onPostExecute(s);
try {
JSONObject jsonObject=new JSONObject(s); JSONArray
items=jsonObject.getJSONArray("items"); JSONObject firstObject=items.getJSONObject(0);
JSONObject volumeInfo=firstObject.getJSONObject("volumeInfo"); String
title=volumeInfo.getString("title");
JSONArray authors=volumeInfo.getJSONArray("authors"); String author=authors.getString(0);
t1.setText(title); t2.setText(author);
} catch (JSONException e) { e.printStackTrace();
}
}
}
}
```

Output:


5A4_LC15
Android
GET
BookName
BookAuthor


5A4_LC15
Android
GET
Beginning Android Application Development
Wei-Ming Lee