# EXPERIMENT 1

**AIM:** Select any five Mobile Apps used in your mobile phone and note the various functionalities and their corresponding layers

**Explanation:**

**Google Chrome:**

1. Functionalities:
   - Web Browsing: Chrome provides a web browsing experience, allowing users to visit websites, interact with web content, and navigate the internet.
   - Bookmarking: Users can save favorite websites for quick access.
   - Tab Management: Chrome supports multiple tabs, enabling users to have several web pages open simultaneously.
   - Incognito Mode: Users can browse privately without saving browsing history.
   - Downloads: Chrome allows users to download files from the web.
   - Settings: Users can customize their browsing experience through settings.
2. Corresponding Layers (Android Software Stack):
   - Application Layer: This layer represents the Chrome app itself, including the user interface and user interactions such as browsing, bookmarking, and managing tabs.
   - Application Framework: Android's application framework provides essential services to apps, including activity and service management, content providers, and intent resolution.
   - Android Runtime (ART): The runtime environment in which Chrome runs. Chrome is a complex web browser with its own JavaScript engine (V8).
   - Libraries: Chrome uses various Android libraries for tasks like rendering web content, managing network connections, and handling downloads.
   - Linux Kernel: The foundation of the Android operating system, handling low-level hardware interactions and resource management.

**WhatsApp:**

1. Functionalities:
   - Messaging: WhatsApp allows users to send text messages, voice messages, images, videos, and documents to individuals or groups.
   - Voice and Video Calls: Users can make voice and video calls to their contacts.
   - Status Updates: WhatsApp enables users to share photos, videos, and text as status updates visible to their contacts.
   - Media Sharing: Users can share multimedia content such as images, videos, and documents.
   - Group Chats: WhatsApp supports group chat creation and participation.
   - Encryption: End-to-end encryption ensures secure communication.
   - Profile Settings: Users can manage their profile picture, status, and account settings.

2. Corresponding Layers (Android Software Stack):
   - Application Layer: Represents the WhatsApp app, including the user interface (chat interface, contact list, settings) and user interactions.
   - Application Framework: Provides essential Android services for managing activities, services, content providers, and intents.

- Android Runtime (ART): Executes WhatsApp's code and logic.
- Libraries: Utilized for functions like encryption, network communication, and media handling.
- Linux Kernel: The core of the Android OS, handling low-level hardware and resource management

**Google Play Store:**

1. Functionalities:
   - App Discovery: Allows users to discover and search for Android apps and games.
   - App Installation and Updates: Provides a platform for downloading, installing, and updating apps.
   - App Ratings and Reviews: Allows users to rate and review apps.
   - App Permissions: Displays information about app permissions before installation.
   - In-App Purchases: Supports in-app purchases for premium content and features.
   - App Management: Enables users to uninstall, disable, or force stop apps.
   - User Account: Requires a Google account for app installations and purchases.
   - Settings: Offers various settings related to auto-updates, data usage, and parental controls.
   - Recommendations: Provides app recommendations based on user preferences.
2. Corresponding Layers (Android Software Stack):
   - Application Layer: Represents the Google Play Store app, including the user interface for discovering, installing, and managing apps.
   - Application Framework: Android framework for managing app installations, permissions, and updates.
   - Android Runtime (ART): Executes the code specific to the Play Store app.
   - Libraries: Used for app discovery, downloading, and authentication.
   - Linux Kernel: Manages hardware resources, network connectivity, and file system operations.

**Photos (Gallery or Google Photos):**

1. Functionalities:
   - Photo Viewing: Allows users to view and manage photos and videos stored on the device or in the cloud.
   - Albums: Organizes photos and videos into user-defined albums.
   - Cloud Sync: Automatically backs up photos and videos to cloud storage.
   - Editing: Provides basic photo editing features like cropping and adjusting brightness.
   - Sharing: Enables users to share photos and videos with contacts or on social media.

2. Corresponding Layers (Android Software Stack):
   - Application Layer: Represents the Photos app, which provides the user interface for viewing and managing photos and videos.
   - Application Framework: Android framework for activity management and handling media content.
   - Android Runtime (ART): Executes the code specific to the Photos app.
   - Libraries: Used for media content management, cloud synchronization, and image processing.
   - Linux Kernel: Manages hardware resources and file system operations.

**Camera:**
1. Functionalities:
- Photo and Video Capture: Allows users to take photos and record videos using the device's camera.
- Camera Modes: Offers various modes such as portrait, panorama, and HDR.
- Filters and Effects: Provides options for applying filters and effects in real-time.
- Flash and Settings: Allows users to control camera settings like flash, resolution, and exposure.
- Gallery Integration: Supports direct access to photos and videos taken with the camera.

2. Corresponding Layers (Android Software Stack):
- Application Layer: Represents the Camera app, including the camera interface and user interactions.
- Application Framework: Android framework for managing the camera hardware and camera-related activities.
- Android Runtime (ART): Executes the Camera app's code.
- Libraries: Used for camera hardware interaction, image processing, and media storage.
- Linux Kernel: Manages hardware resources and camera drivers.

**Contacts (Contacts or Google Contacts):**
1. Functionalities:
- Contact Management: Allows users to create, edit, and delete contacts.
- Contact Sync: Synchronizes contacts with cloud services like Google Contacts.
- Contact Groups: Organizes contacts into user-defined groups.
- Contact Details: Stores various details for each contact, including name, phone number, email, and address.
- Caller ID: Displays contact information for incoming calls.

2. Corresponding Layers (Android Software Stack):
- Application Layer: Represents the Contacts app, which provides the user interface for managing contacts.
- Application Framework: Android framework for managing contacts and handling contact-related activities.
- Android Runtime (ART): Executes the Contacts app's code.
- Libraries: Used for contact data storage, synchronization, and access.
- Linux Kernel: Manages hardware resources and file system operations.

**Messages (Default SMS/MMS App):**
1. Functionalities:
- SMS Messaging: Allows users to send and receive text messages to/from other phone numbers.
- MMS Messaging: Supports the sending and receiving of multimedia messages, including images, videos, and audio.
- Conversations: Organizes messages into threaded conversations with contacts.
- Contact Integration: Accesses contacts from the device's contact list for sending messages.
- Message Search: Provides a search feature to find specific messages or conversations.
- Notifications: Displays new message notifications in the status bar and allows users to configure notification settings.

- Message Settings: Offers settings for configuring SMS/MMS options, delivery reports, and message storage.
- Backup and Restore: Some versions of the app may support backup and restore functionalities for messages.

2. Corresponding Layers (Android Software Stack):
    - Application Layer: Represents the Messages app, which includes the user interface for composing, sending, and receiving messages.
    - Application Framework: Android framework for managing SMS/MMS messaging, contact integration, and notifications.
    - Android Runtime (ART): Executes the code specific to the Messages app.
    - Libraries: Used for messaging protocols (SMS, MMS), contact access, notification handling, and message storage.
    - Linux Kernel: Manages hardware resources, communication with the cellular network, and file system operations.

**Calendar:**

1. Functionalities:
    - Event Creation: Allows users to create and manage events and appointments.
    - Calendar Views: Provides various views like daily, weekly, and monthly calendars.
    - Reminders: Sends notifications for upcoming events.
    - Sync: Synchronizes calendar events with online accounts (e.g., Google Calendar).
    - Sharing: Enables users to share events and calendars with others.
2. Corresponding Layers (Android Software Stack):
    - Application Layer: Represents the Calendar app, including the user interface for managing events and calendars.
    - Application Framework: Android framework for managing calendar data and handling event-related activities.
    - Android Runtime (ART): Executes the Calendar app's code.
    - Libraries: Used for calendar data storage, synchronization, and event notifications.
    - Linux Kernel: Manages hardware resources and file system operations.

**Clock (Clock or Google Clock):**

1. Functionalities:
    - Time and Date: Displays current time, date, and world clocks.
    - Alarms: Allows users to set alarms for waking up or other reminders.
    - Timer and Stopwatch: Provides timer and stopwatch functionality.
    - Bedtime Mode: Helps users establish healthy sleep patterns with reminders.
2. Corresponding Layers (Android Software Stack):
    - Application Layer: Represents the Clock app, including the user interface for displaying time, setting alarms, and managing timers.
    - Application Framework: Android framework for managing time-related functions, alarms, and timers.
    - Android Runtime (ART): Executes the Clock app's code.
    - Libraries: Used for time-related functions, alarms, and notifications.
    - Linux Kernel: Manages hardware resources and system time.

**Google Maps:**
1. Functionalities:
- Navigation: Provides driving, walking, and public transport directions.
- Location Sharing: Allows users to share real-time location with contacts.
- Local Business Information: Provides details about nearby restaurants, shops, and more.
- Street View: Allows users to explore streets with 360-degree photos.

2. Corresponding Layers (Android Software Stack):
- Application Layer: Represents the Google Maps app, including the map interface and navigation controls.
- Application Framework: Android framework handles activity management and provides location services.
- Android Runtime (ART): Executes the code specific to the Google Maps app.
- Libraries: Used for map rendering, location services, and network communication.
- Linux Kernel: Manages hardware resources and low-level interactions.

**Gmail:**
1. Functionalities:
- Email Management: Allows users to send, receive, and organize emails.
- Labels and Folders: Helps categorize emails for better organization.
- Search: Provides a robust email search functionality.
- Notifications: Sends notifications for new emails.

2. Corresponding Layers (Android Software Stack):
- Application Layer: Represents the Gmail app, including the email interface and user interactions.
- Application Framework: Android framework manages activities and provides email protocols (IMAP, SMTP) support.
- Android Runtime (ART): Executes Gmail app's specific code.
- Libraries: Used for email communication, network connectivity, and data storage.
- Linux Kernel: Manages hardware resources and low-level operations.

**YouTube:**
1. Functionalities:
- Video Streaming: Allows users to watch videos from a vast library of content.
- Subscriptions: Enables users to subscribe to channels for updates.
- Comments and Likes: Allows interaction with videos through comments and likes.
- Upload: Provides the capability to upload and share videos with the YouTube community.

2. Corresponding Layers (Android Software Stack):
- Application Layer: Represents the YouTube app, including the video player and user interactions.
- Application Framework: Android framework manages activities and provides media playback support.
- Android Runtime (ART): Executes YouTube app's specific code.
- Libraries: Used for video streaming, network communication, and data storage.
- Linux Kernel: Manages hardware resources and low-level operations.

**Facebook:**

1. Functionalities:
   - Social Networking: Users can connect with friends, post status updates, photos, and videos.
   - Messaging: Facebook Messenger for text and video chat.
   - News Feed: View updates from friends and pages followed.
   - Events: Create and manage events.
2. Corresponding Layers (Android Software Stack):
   - Application Layer: Facebook app with user interface and interactions.
   - Application Framework: Android framework for activity management.
   - Android Runtime (ART): Executes Facebook app's code.
   - Libraries: Utilized for networking, image rendering, and data storage.
   - Linux Kernel: Manages hardware resources.

**Instagram:**

1. Functionalities:
   - Photo and Video Sharing: Upload and share media with captions.
   - Feed: Scroll through a feed of posts from people followed.
   - Stories: View and create temporary stories.
   - Explore: Discover new content based on interests.
   - Direct Messaging: Send private messages to other users.
2. Corresponding Layers (Android Software Stack):
   - Application Layer: Instagram app with user interface and interactions.
   - Application Framework: Android framework for activity management.
   - Android Runtime (ART): Executes Instagram app's code.
   - Libraries: Used for image rendering, networking, and data storage.
   - Linux Kernel: Manages hardware resources.

**Twitter:**

1. Functionalities:
   - Tweeting: Post short messages, photos, and videos.
   - Timeline: View tweets from followed accounts.
   - Trends: Discover popular topics.
   - Notifications: Receive updates on mentions and likes.

2. Corresponding Layers (Android Software Stack):
   - Application Layer: Twitter app with user interface and tweet interactions.
   - Application Framework: Android framework for activity management.
   - Android Runtime (ART): Executes Twitter app's code.
   - Libraries: Used for networking, image rendering, and data storage.
   - Linux Kernel: Manages hardware resources.

## EXPERIMENT 2

**AIM:** Install Android Studio and Configure Latest Android SDKs and Android Virtual Devices

**Explanation:**

**Android Studio:**

Android Studio is the official IDE (Integrated Development Environment) for Android app development and it is based on JetBrains' IntelliJ IDEA software. Android Studio provides many excellent features that enhance productivity when building Android apps, such as:

- A blended environment where one can develop for all Android devices
- Apply Changes to push code and resource changes to the running app without restarting the app
- A flexible Gradle-based build system
- A fast and feature-rich emulator
- GitHub and Code template integration to assist you to develop common app features and import sample code
- Extensive testing tools and frameworks
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine, and many more.
- Provides GUI tools that simplify the less interesting parts of app development.
- Easy integration with real time database 'firebase'.

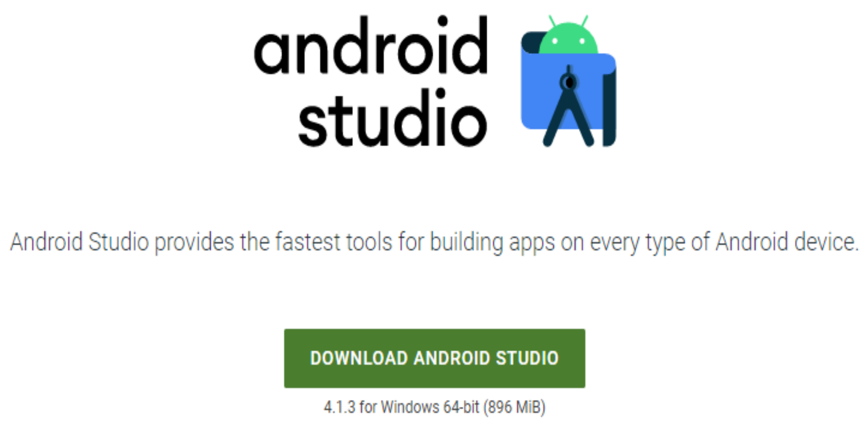
System Requirements

- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- 4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Installation Guide

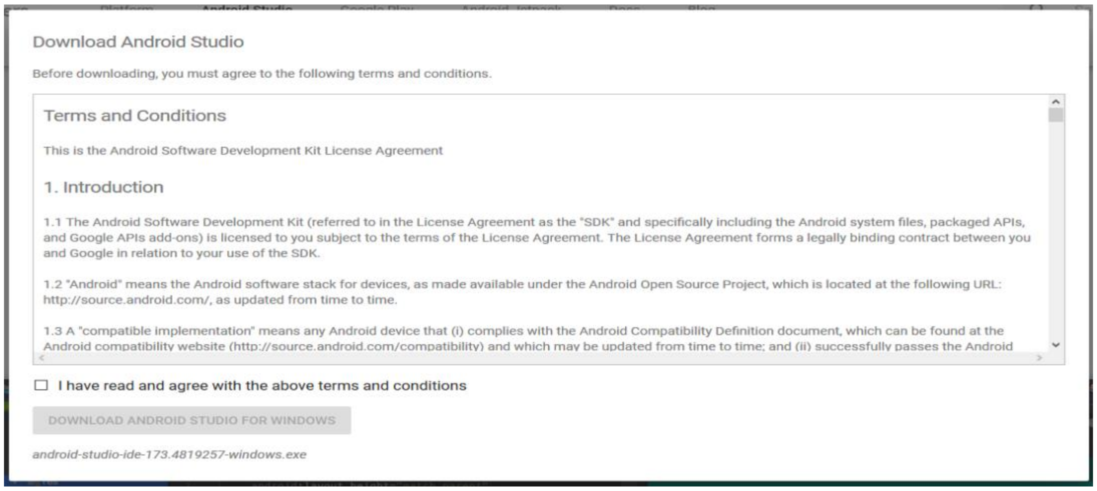
Step 1: Head over to "https://developer.android.com/studio#downloads" to get the Android Studio executable or zip file.

Step 2: Click on the Download Android Studio Button.



Android Studio provides the fastest tools for building apps on every type of Android device.

Click on the "I have read and agree with the above terms and conditions" checkbox followed by the download button.



Download Android Studio

Before downloading, you must agree to the following terms and conditions.

Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.

1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android

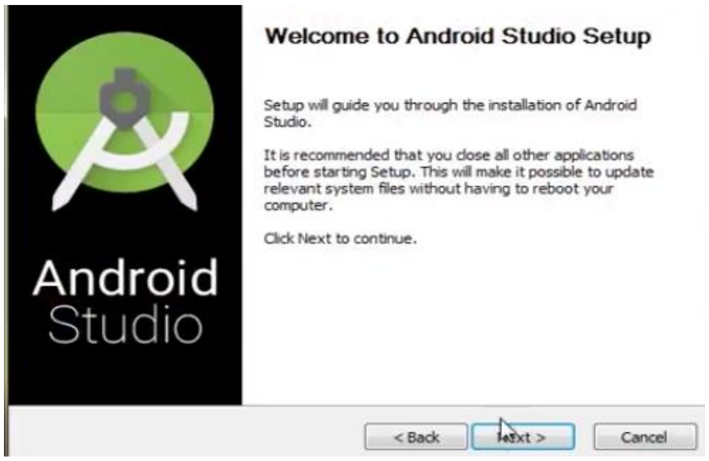
☐ I have read and agree with the above terms and conditions

DOWNLOAD ANDROID STUDIO FOR WINDOWS

android-studio-ide-173.4819257-windows.exe

Click on the Save file button in the appeared prompt box and the file will start downloading.

Step 3: After the downloading has finished, open the file from downloads and run it. It will prompt the following dialog box.



Welcome to Android Studio Setup

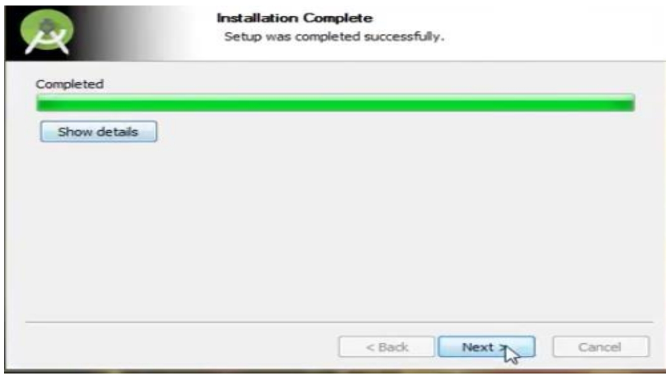
Setup will guide you through the installation of Android Studio.

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

Click Next to continue.

Click on next. In the next prompt, it'll ask for a path for installation. Choose a path and hit next.

Step 4: It will start the installation, and once it is completed, it will be like the image shown below.

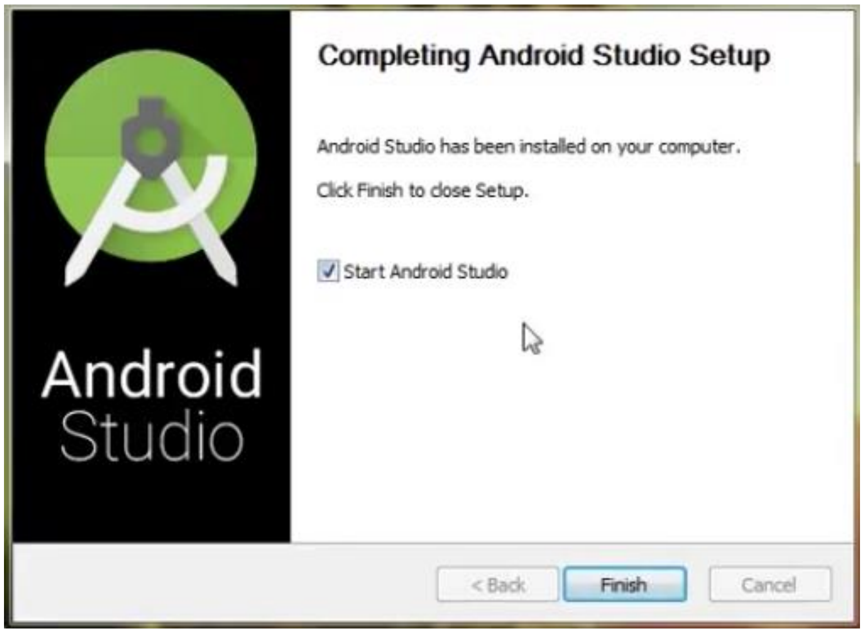


Installation Complete

Setup was completed successfully.

Completed

Click on next.

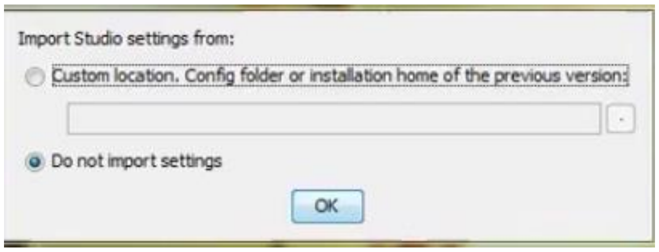

Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

[✓] Start Android Studio

Step 5: Once "Finish" is clicked, it will ask whether the previous settings need to be imported [if the android studio had been installed earlier], or not. It is better to choose the 'Don't import Settings option'.


Import Studio settings from:

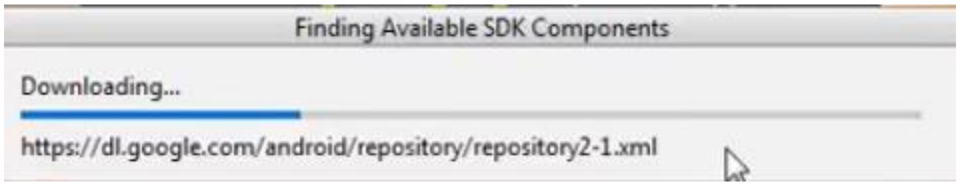
○ Custom location. Config folder or installation home of the previous version:

● Do not import settings

Click the OK button.

Step 6: This will start the Android Studio.

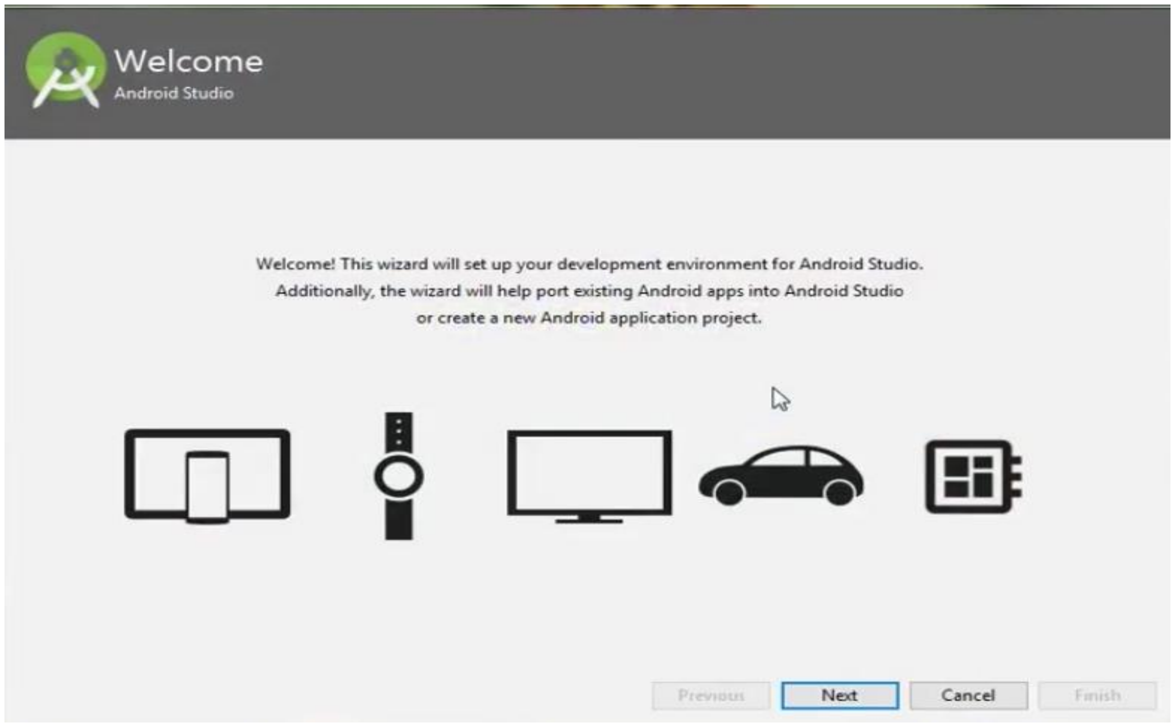
Meanwhile, it will be finding the available SDK components.


Finding Available SDK Components

Downloading...

https://dl.google.com/android/repository/repository2-1.xml

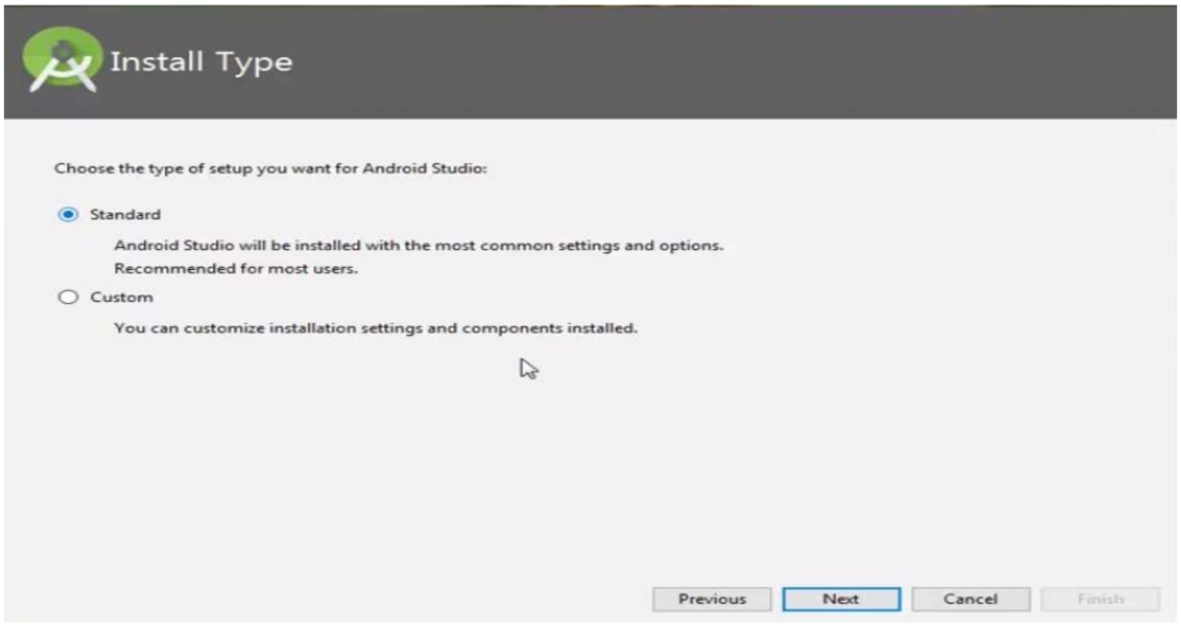
Step 7: After it has found the SDK components, it will redirect to the Welcome dialog box.


Welcome

Android Studio

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

Click on Next.


Install Type

Choose the type of setup you want for Android Studio:

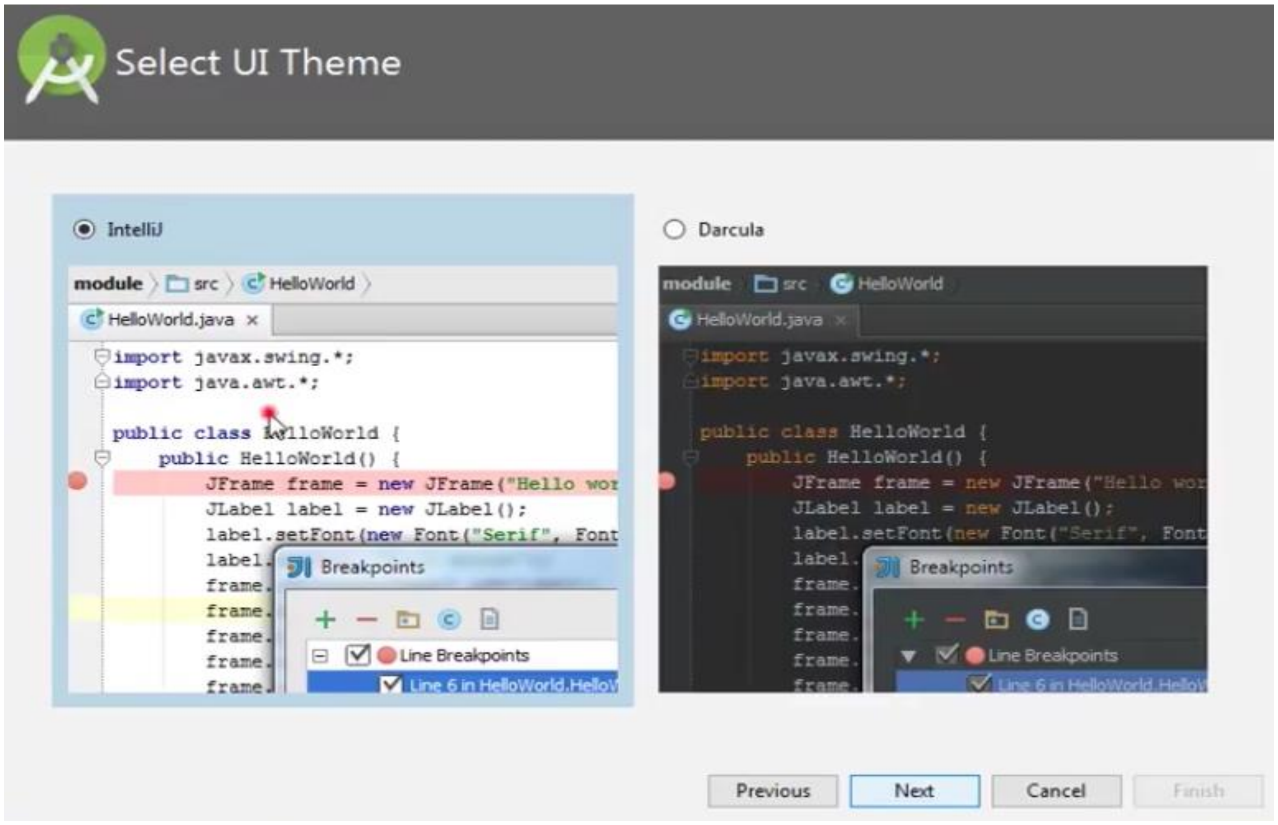
[✓ Standard]

Android Studio will be installed with the most common settings and options. Recommended for most users.

Custom

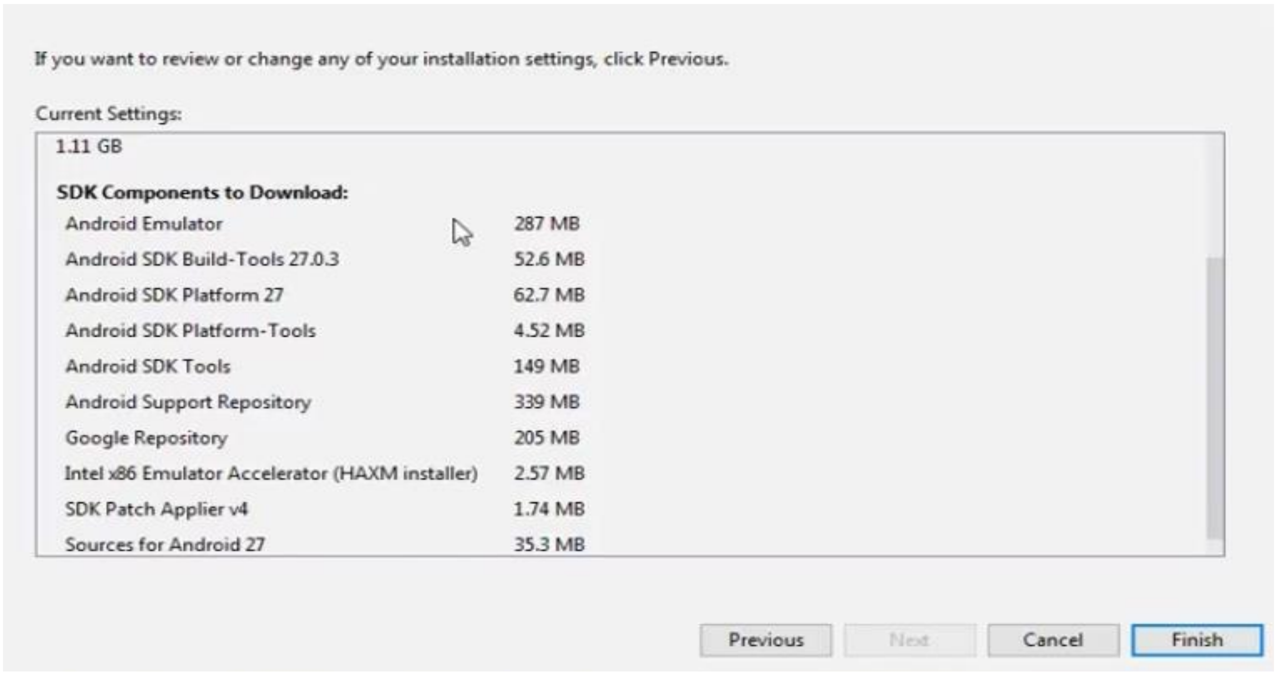
You can customize installation settings and components installed.

Choose Standard and click on Next. Now choose the theme, whether the Light theme or the Dark one . The light one is called the IntelliJ theme whereas the dark theme is called Dracula. Choose as required.



Click on the Next button.

Step 8: Now it is time to download the SDK components.



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

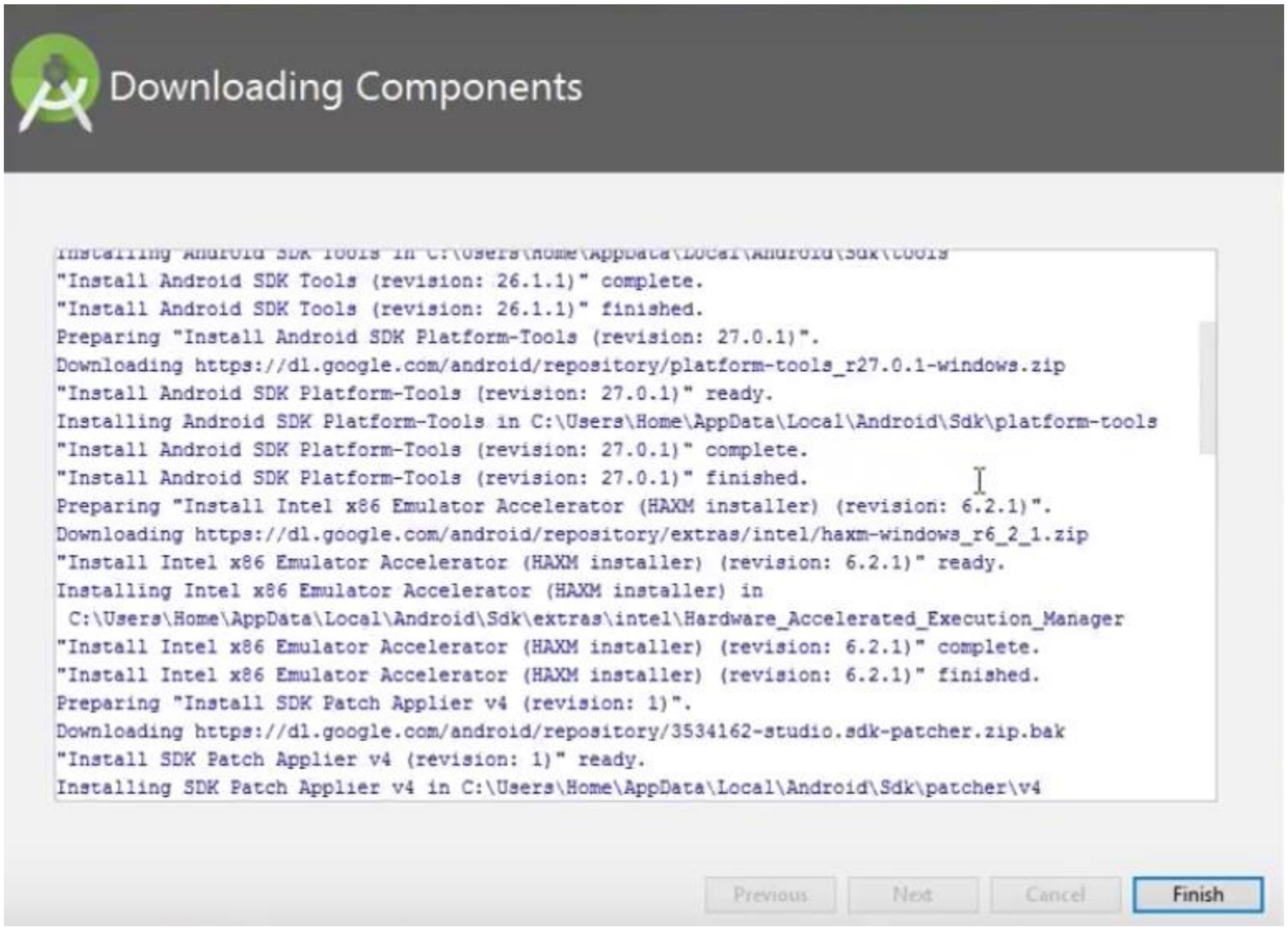
Current Settings:

1.11 GB

**SDK Components to Download:**

| | |
|---|---|
| Android Emulator | 287 MB |
| Android SDK Build-Tools 27.0.3 | 52.6 MB |
| Android SDK Platform 27 | 62.7 MB |
| Android SDK Platform-Tools | 4.52 MB |
| Android SDK Tools | 149 MB |
| Android Support Repository | 339 MB |
| Google Repository | 205 MB |
| Intel x86 Emulator Accelerator (HAXM installer) | 2.57 MB |
| SDK Patch Applier v4 | 1.74 MB |
| Sources for Android 27 | 35.3 MB |

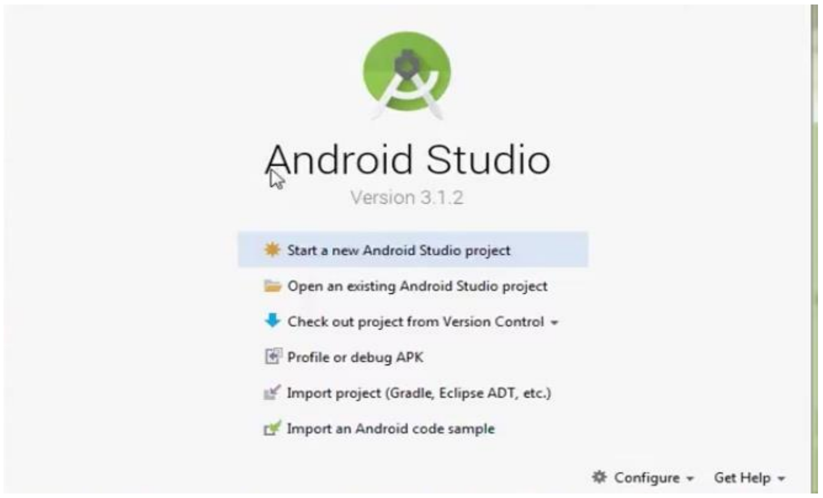
Click on Finish. Components begin to download let it complete.



```
Installing Android SDK Tools in C:\Users\Home\AppData\Local\Android\Sdk\tools
"Install Android SDK Tools (revision: 26.1.1)" complete.
"Install Android SDK Tools (revision: 26.1.1)" finished.
Preparing "Install Android SDK Platform-Tools (revision: 27.0.1)".
Downloading https://dl.google.com/android/repository/platform-tools_r27.0.1-windows.zip
"Install Android SDK Platform-Tools (revision: 27.0.1)" ready.
Installing Android SDK Platform-Tools in C:\Users\Home\AppData\Local\Android\Sdk\platform-tools
"Install Android SDK Platform-Tools (revision: 27.0.1)" complete.
"Install Android SDK Platform-Tools (revision: 27.0.1)" finished.
Preparing "Install Intel x86 Emulator Accelerator (HAXM installer) (revision: 6.2.1)".
Downloading https://dl.google.com/android/repository/extras/intel/haxm-windows_r6_2_1.zip
"Install Intel x86 Emulator Accelerator (HAXM installer) (revision: 6.2.1)" ready.
Installing Intel x86 Emulator Accelerator (HAXM installer) in
 C:\Users\Home\AppData\Local\Android\Sdk\extras\intel\Hardware_Accelerated_Execution_Manager
"Install Intel x86 Emulator Accelerator (HAXM installer) (revision: 6.2.1)" complete.
"Install Intel x86 Emulator Accelerator (HAXM installer) (revision: 6.2.1)" finished.
Preparing "Install SDK Patch Applier v4 (revision: 1)".
Downloading https://dl.google.com/android/repository/3534162-studio.sdk-patcher.zip.bak
"Install SDK Patch Applier v4 (revision: 1)" ready.
Installing SDK Patch Applier v4 in C:\Users\Home\AppData\Local\Android\Sdk\patcher\v4
```

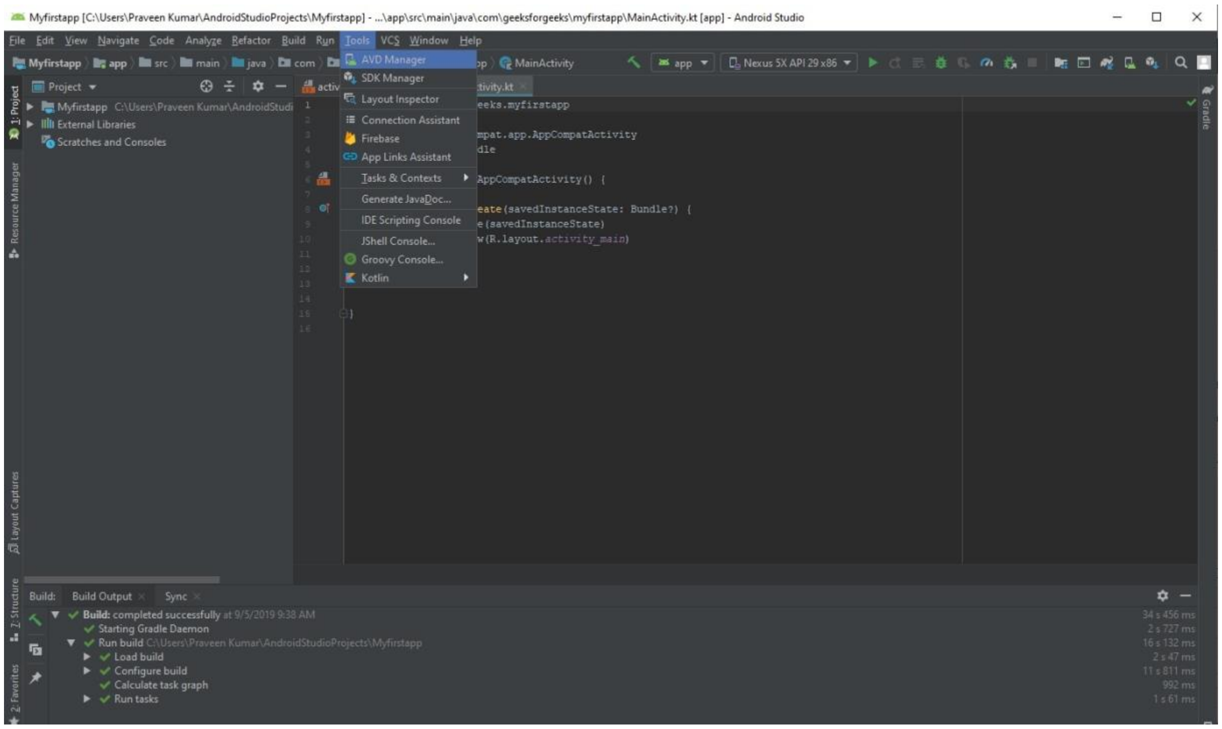
The Android Studio has been successfully configured. Now it’s time to launch and build apps. Click on the Finish button to launch it.

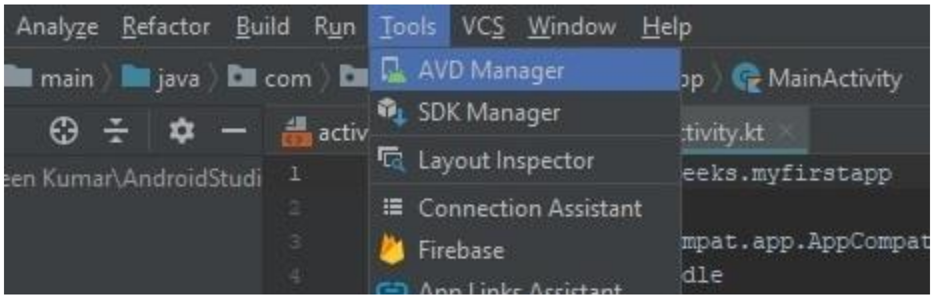
Step 9: Click on Start a new Android Studio project to build a new app.


Android Studio
Version 3.1.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

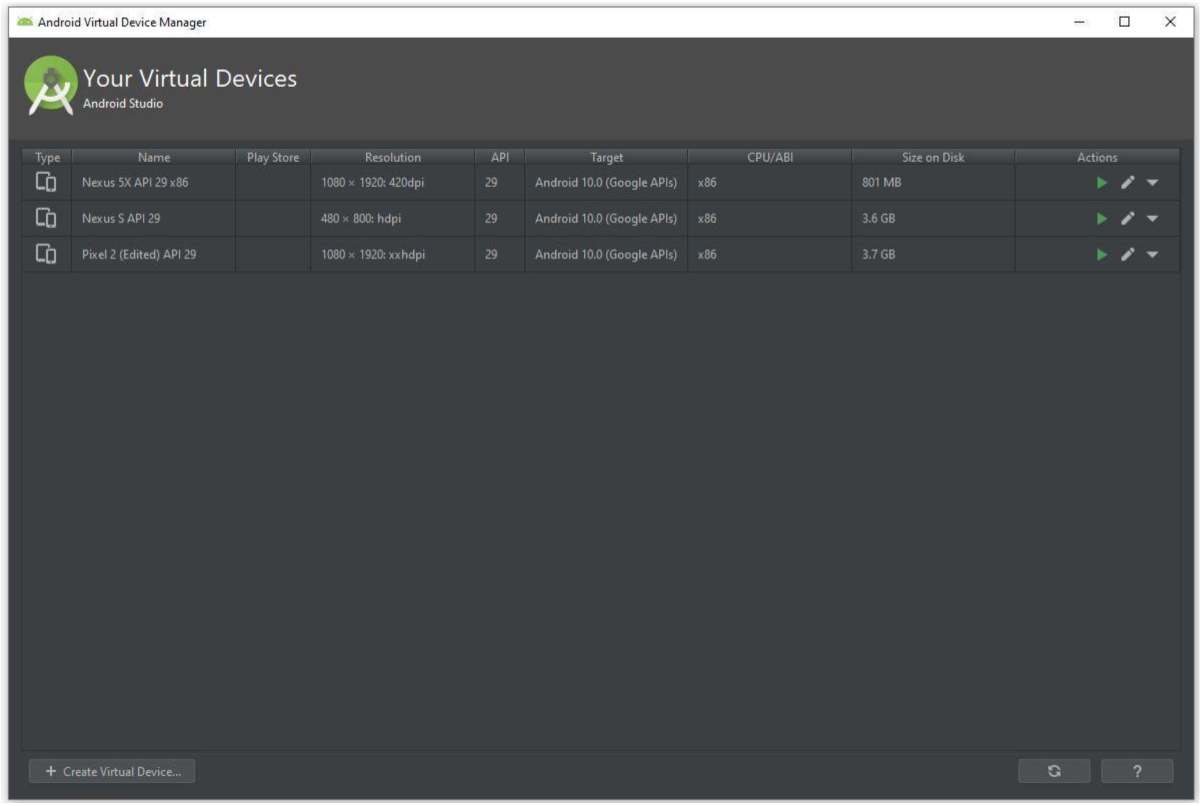
To run your first android app in Android Studio you may refer to Running your first Android app.

**Android Virtual Device:**
In android development, we need an android device to run the application. So, developers of Android Studio provide an option to install android virtual device to run it. In this article, we will learn how to install Android Virtual Device (AVD).
Follow the below steps to install Android Virtual Device.
Step 1: Go to Tools > AVD Manager.





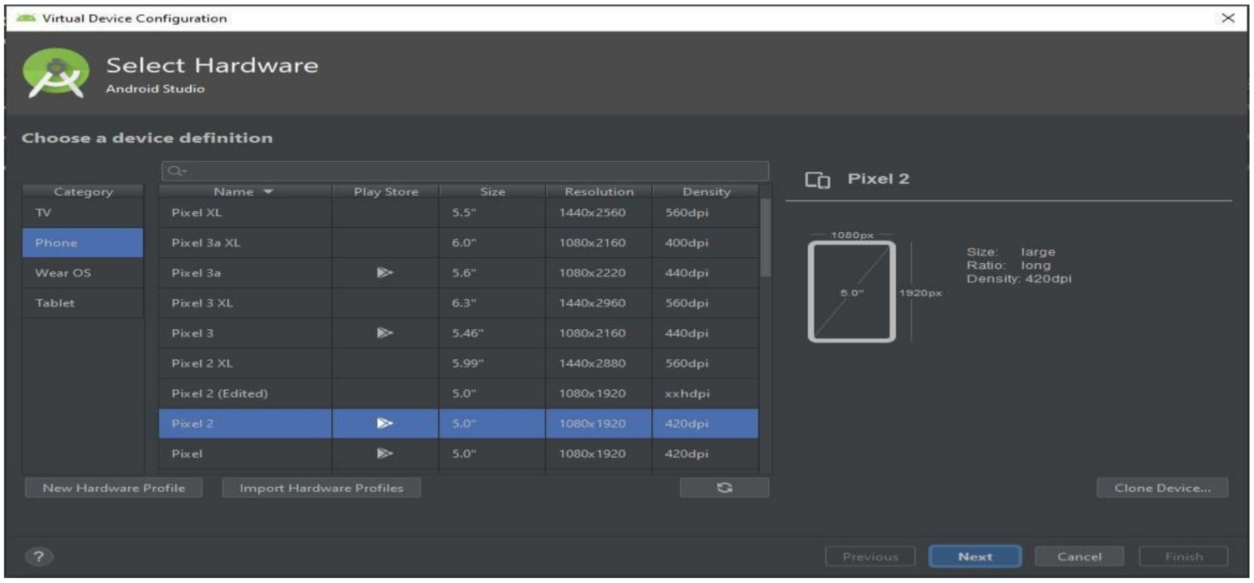
Step 2: Now click on Create Virtual Device.



Your Virtual Devices
Android Studio

| Type | Name | Play Store | Resolution | API | Target | CPU/ABI | Size on Disk | Actions |
|---|---|---|---|---|---|---|---|---|
| | Nexus 5X API 29 x86 | | 1080 × 1920: 420dpi | 29 | Android 10.0 (Google APIs) | x86 | 801 MB | |
| | Nexus S API 29 | | 480 × 800: hdpi | 29 | Android 10.0 (Google APIs) | x86 | 3.6 GB | |
| | Pixel 2 (Edited) API 29 | | 1080 × 1920: xxhdpi | 29 | Android 10.0 (Google APIs) | x86 | 3.7 GB | |

Step 3: A pop-up window will be there and here we select the category Phone because we are creating android app for mobile and select the model of mobile phone we want to install.



Select Hardware
Android Studio

Choose a device definition

| Category |
|---|
| TV |
| Phone |
| Wear OS |
| Tablet |

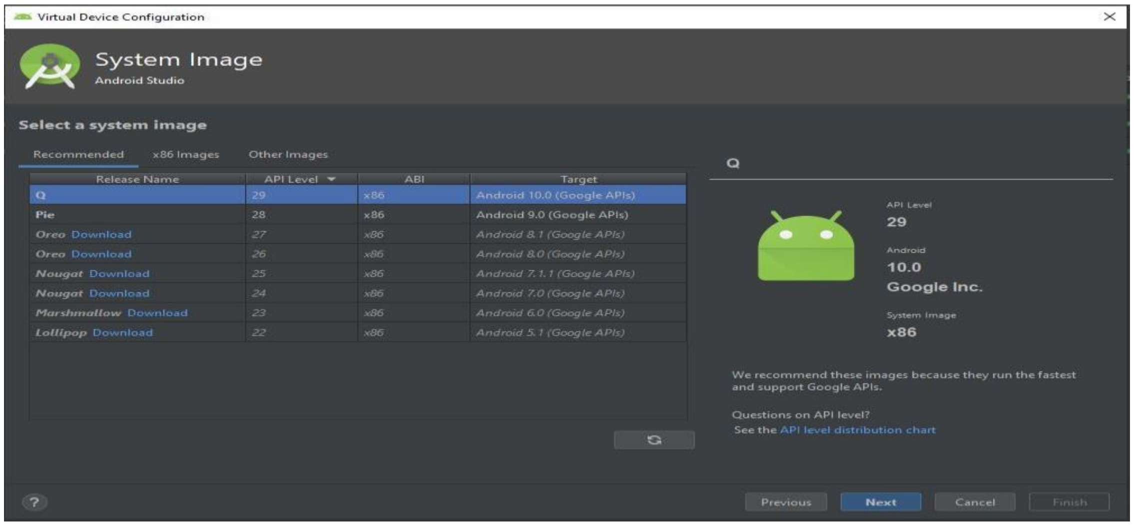
| Name | Play Store | Size | Resolution | Density |
|---|---|---|---|---|
| Pixel XL | | 5.5" | 1440x2560 | 560dpi |
| Pixel 3a XL | | 6.0" | 1080x2160 | 400dpi |
| Pixel 3a | ▶ | 5.6" | 1080x2220 | 440dpi |
| Pixel 3 XL | | 6.3" | 1440x2960 | 560dpi |
| Pixel 3 | ▶ | 5.46" | 1080x2160 | 440dpi |
| Pixel 2 XL | | 5.99" | 1440x2880 | 560dpi |
| Pixel 2 (Edited) | | 5.0" | 1080x1920 | xxhdpi |
| Pixel 2 | ▶ | 5.0" | 1080x1920 | 420dpi |
| Pixel | ▶ | 5.0" | 1080x1920 | 420dpi |

Pixel 2

1080px
5.0"
1920px

Size: large
Ratio: long
Density: 420dpi

Step 4: Here we select the android version to download like Q, Pie, Oreo etc and click Next button.



Select a system image

| Release Name | API Level | ABI | Target |
|---|---|---|---|
| Q | 29 | x86 | Android 10.0 (Google APIs) |
| Pie | 28 | x86 | Android 9.0 (Google APIs) |
| Oreo Download | 27 | x86 | Android 8.1 (Google APIs) |
| Oreo Download | 26 | x86 | Android 8.0 (Google APIs) |
| Nougat Download | 25 | x86 | Android 7.1.1 (Google APIs) |
| Nougat Download | 24 | x86 | Android 7.0 (Google APIs) |
| Marshmallow Download | 23 | x86 | Android 6.0 (Google APIs) |
| Lollipop Download | 22 | x86 | Android 5.1 (Google APIs) |

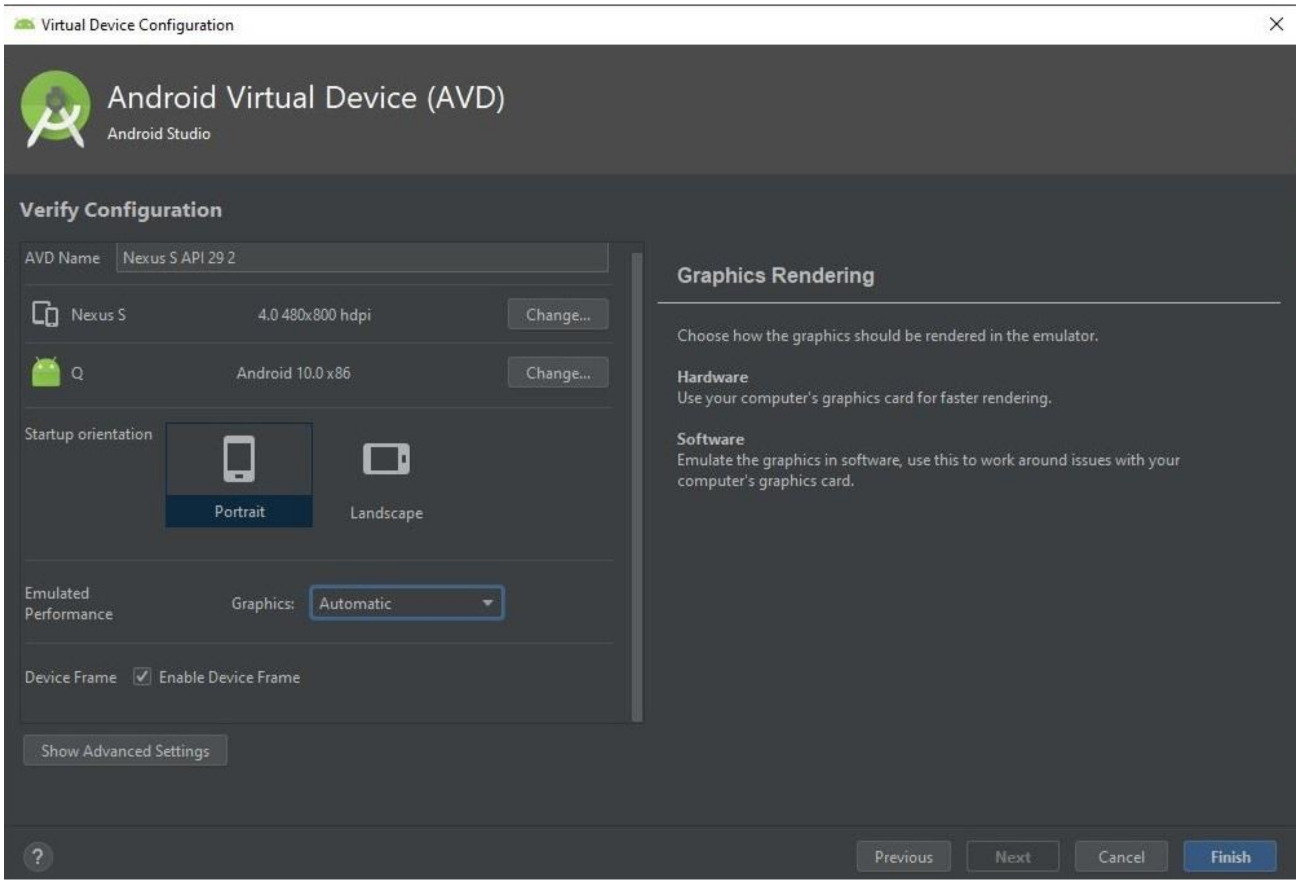
Q

API Level: 29
Android: 10.0 Google Inc.
System Image: x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

Step 5: Click the finish button to complete the installation.



Verify Configuration

AVD Name: Nexus S API 29 2
Nexus S 4.0 480x800 hdpi
Q Android 10.0 x86
Startup orientation: Portrait
Emulated Performance Graphics: Automatic
Device Frame ✓ Enable Device Frame

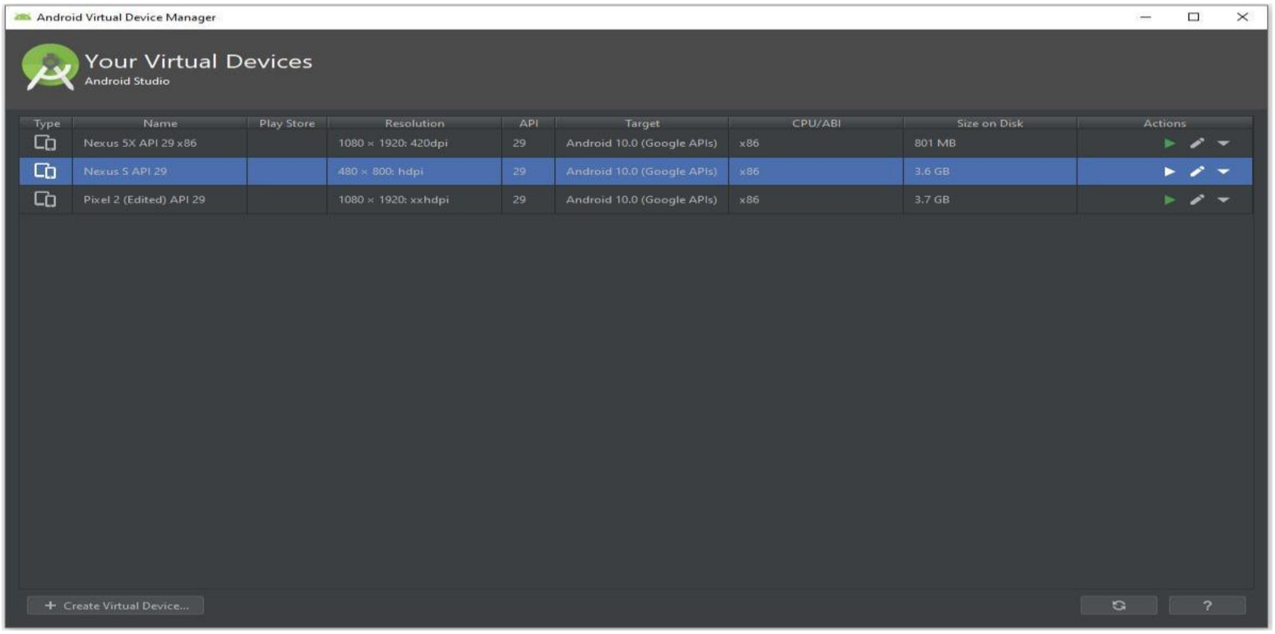
Graphics Rendering

Choose how the graphics should be rendered in the emulator.

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

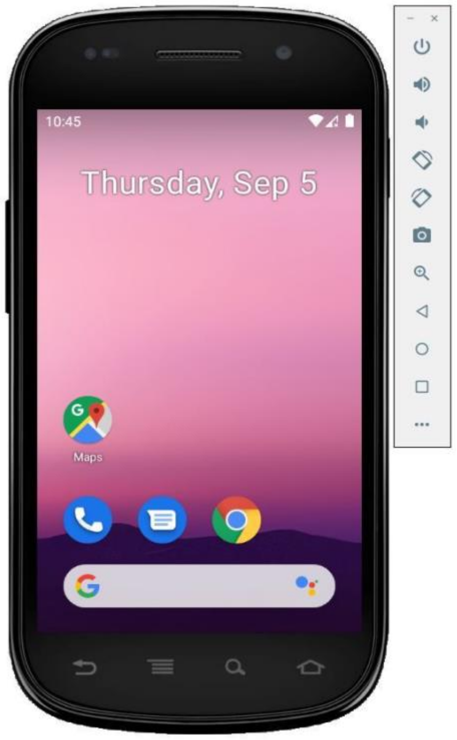
Software
Emulate the graphics in software, use this to work around issues with your computer's graphics card.

Step 6: Now we can select the virtual device we want to run as emulator can click on the run icon.



| Type | Name | Play Store | Resolution | API | Target | CPU/ABI | Size on Disk | Actions |
|---|---|---|---|---|---|---|---|---|
| | Nexus 5X API 29 x86 | | 1080 × 1920: 420dpi | 29 | Android 10.0 (Google APIs) | x86 | 801 MB | |
| | Nexus S API 29 | | 480 × 800: hdpi | 29 | Android 10.0 (Google APIs) | x86 | 3.6 GB | |
| | Pixel 2 (Edited) API 29 | | 1080 × 1920: xxhdpi | 29 | Android 10.0 (Google APIs) | x86 | 3.7 GB | |

Step 7: Finally our virtual device is ready to run our android app.



**EXPERIMENT 3**

**AIM:** Build and Run Hello World Application on the virtual Device and also test the app on your mobile phone

**Program:**

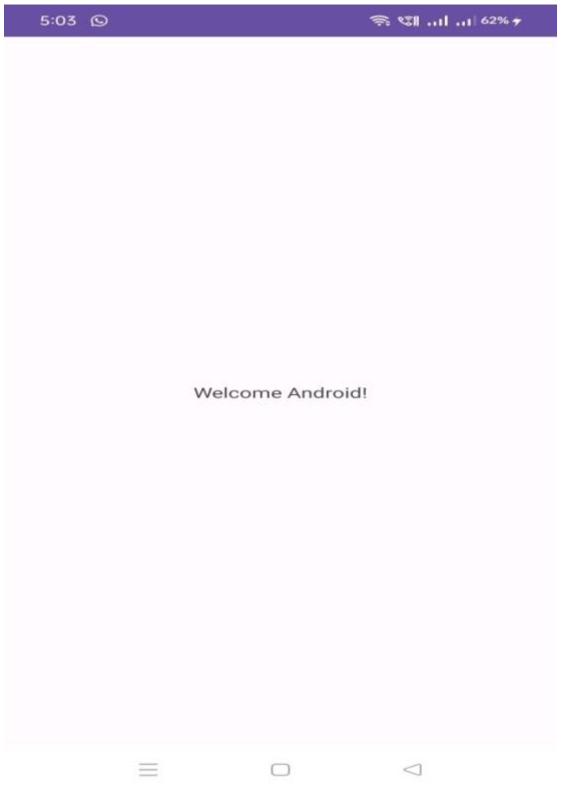
**Xml file:**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="Vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.helloworld.MainActivity">
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Android!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
  </ LinearLayout>
```

Java Code:

```
package  com.example.helloworld;
import android.support.v7.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:**



Welcome Android!

## EXPERIMENT 4

AIM: Explore all the UI Controls and design a Student Registration Activity

Procedure:

XML Code:

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

<TextView
android:layout_width="102dp"
android:layout_height="21dp"
android:layout_x="137dp"
android:layout_y="50dp"
android:text="Registration Form"
tools:layout_editor_absoluteX="125dp"
tools:layout_editor_absoluteY="39dp" />

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="56dp"
android:layout_y="90dp"
android:text="First Name"
tools:layout_editor_absoluteX="43dp"
tools:layout_editor_absoluteY="120dp" />

<EditText
android:id="@+id/fname"
android:layout_width="211dp"
android:layout_height="wrap_content"
android:layout_x="148dp"
android:layout_y="74dp"
tools:layout_editor_absoluteX="151dp"
tools:layout_editor_absoluteY="176dp" />

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="61dp"
android:layout_y="135dp"
android:text="Last Name"
tools:layout_editor_absoluteX="43dp"
tools:layout_editor_absoluteY="179dp" />
```

```
<EditText
android:id="@+id/lname"
android:layout_width="216dp"
android:layout_height="wrap_content"
android:layout_x="145dp"
android:layout_y="128dp"
tools:layout_editor_absoluteX="151dp"
tools:layout_editor_absoluteY="107dp" />

<TextView
android:layout_width="62dp"
android:layout_height="29dp"
android:layout_x="57dp"
android:layout_y="176dp"
android:text="Gender"
tools:layout_editor_absoluteX="44dp"
tools:layout_editor_absoluteY="250dp" />

<RadioGroup
android:layout_width="163dp"
android:layout_height="wrap_content"
android:layout_x="80dp"
android:layout_y="205dp"
tools:layout_editor_absoluteX="161dp"
tools:layout_editor_absoluteY="234dp">

<RadioButton
android:id="@+id/male"
android:layout_width="154dp"
android:layout_height="wrap_content"
android:text="male" />

<RadioButton
android:id="@+id/female"
android:layout_width="154dp"
android:layout_height="wrap_content"
android:text="female" />

</RadioGroup>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="58dp"
android:layout_y="311dp"
android:text="Intrested Sports"
tools:layout_editor_absoluteX="47dp"
tools:layout_editor_absoluteY="385dp" />

<CheckBox
android:id="@+id/Cricket"
```

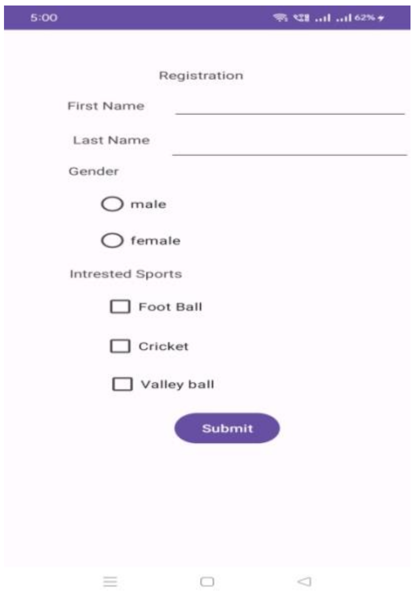
```
android:layout_width="127dp"
android:layout_height="44dp"
android:layout_x="87dp"
android:layout_y="394dp"
android:text="Cricket"
tools:layout_editor_absoluteX="167dp"
tools:layout_editor_absoluteY="385dp" />

<CheckBox
android:id="@+id/football"
android:layout_width="127dp"
android:layout_height="44dp"
android:layout_x="87dp"
android:layout_y="342dp"
android:text="Foot Ball"
tools:layout_editor_absoluteX="167dp"
tools:layout_editor_absoluteY="429dp" />

<CheckBox
android:id="@+id/valleyball"
android:layout_width="127dp"
android:layout_height="44dp"
android:layout_x="89dp"
android:layout_y="444dp"
android:text="Valley ball"
tools:layout_editor_absoluteX="167dp"
tools:layout_editor_absoluteY="473dp" />

<Button
android:id="@+id/submit"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="151dp"
android:layout_y="500dp"
android:text="Submit"
tools:layout_editor_absoluteX="180dp"
tools:layout_editor_absoluteY="555dp" />

<TextView
android:id="@+id/resp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="84dp"
android:layout_y="576dp"
tools:layout_editor_absoluteX="47dp"
tools:layout_editor_absoluteY="385dp" />

</AbsoluteLayout>
```

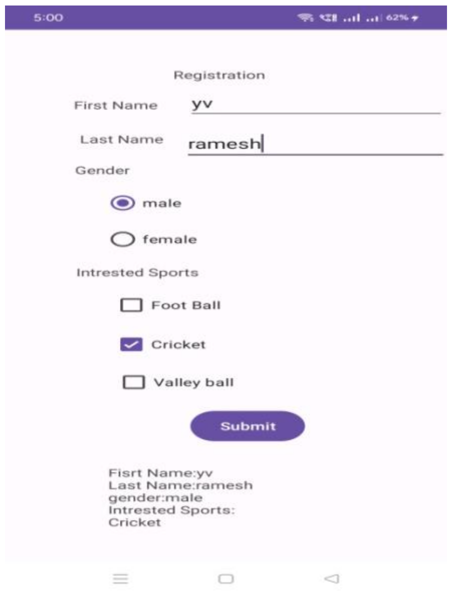
Java Code:

```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;

public class MainActivityextends AppCompatActivityimplements View.OnClickListener {
private Button b;
private TextViewresp;
private EditTextfname,lname;
private RadioButtonr1,r2;
private CheckBoxc1,c2,c3;
   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
b = findViewById(R.id.submit);
r1=(RadioButton) this.findViewById(R.id.male);
r2=(RadioButton) this.findViewById(R.id.female);
fname=(EditText) this.findViewById(R.id.fname);
lname=(EditText) this.findViewById(R.id.lname);
resp=(TextView)this.findViewById(R.id.resp);
c1=(CheckBox)this.findViewById(R.id.Cricket);
c2=(CheckBox)this.findViewById(R.id.football);
c3=(CheckBox)this.findViewById(R.id.valleyball);
b.setOnClickListener(this);
   }
public void onClick(View view)
      {
         String Data1="";
         String Data2="";
if(r1.isChecked()){
            Data1=r1.getText().toString();
         }
if(r2.isChecked()) {
            Data1 = Data1 + "\n" + r2.getText().toString();
         }

if(c1.isChecked()){
```

```
                Data2=c1.getText().toString();
            }
if(c2.isChecked()){
                Data2=Data2+"\n"+c2.getText().toString();
            }
if(c3.isChecked()){
                Data2=Data2+"\n"+c3.getText().toString();
            }
resp.setText("Fisrt Name:"+fname.getText().toString()+"\n"+
"Last Name:"+lname.getText().toString()+"\n"+
"gender:"+Data1+"\n"+
"Intrested Sports:"+"\n"+Data2);
        }
}
```

**Output:**





First Name yv
Last Name ramesh
(● male) female
[✓ Cricket]

```
Fisrt Name:yv
Last Name:ramesh
gender:male
Intrested Sports:
Cricket
```

**EXPERIMENT 5**

Task: Design the Student Registration Activity using Material Design for Android Components

Procedure:

Xml Code:

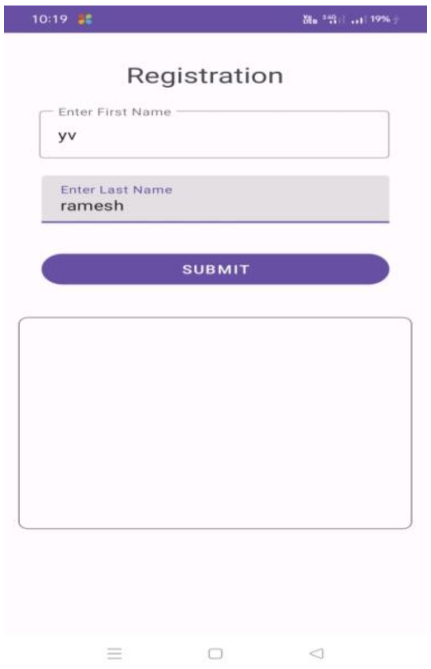
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity"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

    <com.google.android.material.textview.MaterialTextView
        android:id="@+id/text"
        android:layout_width="152dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="32dp"
        android:text="Registration"
        android:textSize="25sp">
    </com.google.android.material.textview.MaterialTextView>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/filledTextField"
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="32dp"
        android:hint="Enter First Name">
        <!--this is the actual edit text which takes the input-->
        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/fname"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/filledTextField1"
        style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="32dp"
        android:hint="Enter Last Name">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/lname"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.button.MaterialButton
        style="@style/Widget.MaterialComponents.Button.UnelevatedButton"
        android:id="@+id/submit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="32dp"
        android:text="Submit">
    </com.google.android.material.button.MaterialButton>

    <com.google.android.material.card.MaterialCardView
        android:id="@+id/text_preview"
        android:layout_width="335dp"
        android:layout_height="248dp"
        android:layout_gravity="center"
        android:layout_marginTop="32dp">

        <TextView
            android:id="@+id/resp"
            android:layout_width="298dp"
            android:layout_height="209dp"
            android:layout_gravity="center"
            android:textSize="18sp" />

    </com.google.android.material.card.MaterialCardView>
</LinearLayout>
```

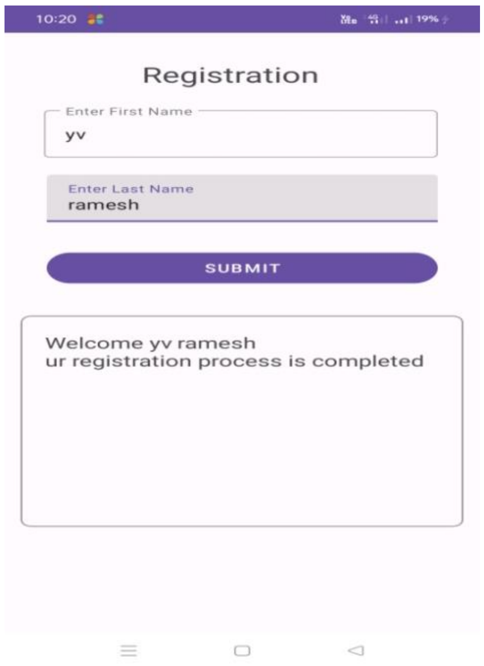
Java Code:

```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
```

```
private Button b;
private TextView resp;
private EditText fname,lname;
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        b = (Button) this.findViewById(R.id.submit);
        fname=(EditText) this.findViewById(R.id.fname);
        lname=(EditText) this.findViewById(R.id.lname);
        resp=(TextView)this.findViewById(R.id.resp);
        b.setOnClickListener(this);
        }
        public void onClick(View view)
        {
                resp.setText("Welcome "+fname.getText().toString()+"\t"+
        lname.getText().toString()+"\n"+"ur registration process is completed");

        }
}
```

Output:





```
Welcome yv ramesh
ur registration process is completed
```

## EXPERIMENT 6

Task: Design a complete Student Management Application using Android and provide effective navigation between various Activities

Procedure: (Bottom Navigation)

Xml Code:

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

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="0sp"
        android:id="@+id/frameLayout"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/bottomNav"/>

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/bottomNav"
        app:layout_constraintTop_toBottomOf="@id/frameLayout"
        app:layout_constraintBottom_toBottomOf="parent"
        app:menu="@menu/bottom_nav_menu"
        app:labelVisibilityMode="labeled"/>

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

Java Code:

```
package com.example.myapplication;
import java.io.*;
import androidx.appcompat.app.AppCompatActivity;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.android.material.navigation.NavigationBarView;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.MenuItem;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       BottomNavigationView bottomNav = findViewById(R.id.bottomNav);
       openFragment(new FirstFragment());
       bottomNav.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
           @Override
           public boolean onNavigationItemSelected(@NonNull MenuItem item) {
               if(item.getItemId()==R.id.home) {
                   openFragment(new FirstFragment());
                   return true;
               }else if(item.getItemId()==R.id.trend){
                       openFragment(new SecondFragment());
                       return true;
               }else if(item.getItemId()==R.id.setting){
                       openFragment(new ThirdFragment());
                       return true;
               }
               return false;
           }
       });
   }
   void openFragment(Fragment fragment){
       FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();
       fragmentTransaction.replace(R.id.frameLayout, fragment);
       fragmentTransaction.addToBackStack(null);
       fragmentTransaction.commit();
   }
}
```

menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/home"
        android:title="COLLEGE NAME"/>
    <item android:id="@+id/trend"
        android:title="BRANCH"/>
    <item android:id="@+id/setting"
        android:title="YEAR"/>
</menu>
```

FirstFragment xml code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
 <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="430dp"
        android:gravity="center"
        android:text="GEETHANJALI INSTITUTE OF SCIENCE AND TECHNOLOGY"
        android:textColor="#43a047"
        android:textSize="40sp"
        android:textStyle="italic|bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

FirstFragment java code:

```
package com.example.myapplication;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.io.*;
import androidx.fragment.app.Fragment;
import com.example.myapplication.R;

public class FirstFragment extends Fragment {
    public FirstFragment(){
        // require a empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_first, container, false);
    }
}
```

SecondFragment xml code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/secondFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="COMPUTER SCIENCE AND ENGINEERING "
        android:textColor="#43a047"
        android:textSize="40sp"
        android:textStyle="italic|bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

SecondFragment java code:

```
package com.example.myapplication;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import androidx.fragment.app.Fragment;
import java.io.*;

public class SecondFragment extends Fragment {
    public SecondFragment(){
        // require a empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_second, container, false);
    }
}
```

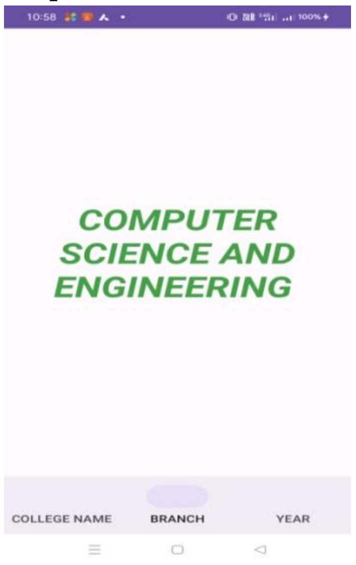
ThirdFragment xml code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/thirdFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="FINAL B.Tech"
        android:textColor="#43a047"
        android:textSize="40sp"
        android:textStyle="italic|bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

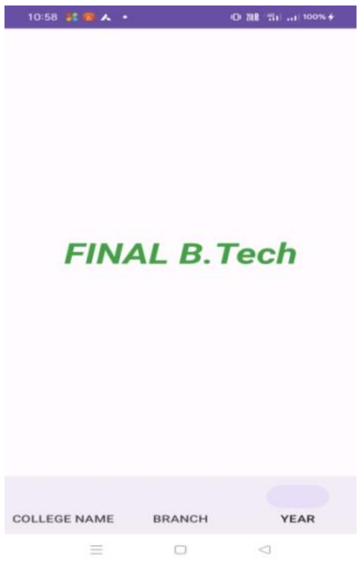
ThirdFragment java code:

```java
package com.example.myapplication;
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import androidx.fragment.app.Fragment;
import java.io.*;

public class ThirdFragment extends Fragment {
    public ThirdFragment(){
        // require a empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_third, container, false);
    }
}
```

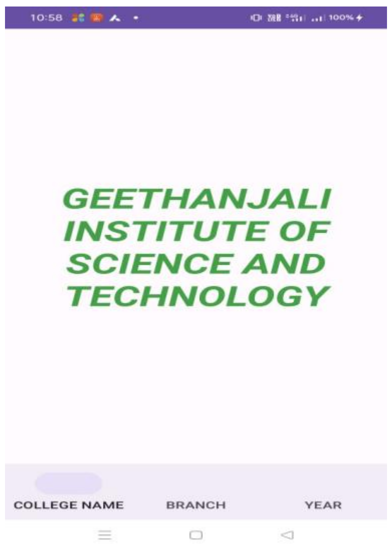
**Output:**



COMPUTER SCIENCE AND ENGINEERING



FINAL B.Tech



GEETHANJALI INSTITUTE OF SCIENCE AND TECHNOLOGY

## EXPERIMENT 7

Task: Develop an Android Application that stores Student Details into the hosting server and retrieve student details from the server

Procedure:

Xml Code:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <TextView
        android:id="@+id/lt1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:gravity="center_horizontal"
        android:text="LOGIN"
        android:textSize="25dp"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="25dp"
        android:text="User Name"
        android:layout_below="@id/lt1"
        android:textStyle="bold"
        android:textSize="45px"
        android:padding="15dp"
        android:layout_marginTop="25dp"
        android:layout_marginLeft="15dp"/>
    <EditText
        android:id="@+id/user_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/username"
        android:padding="15dp"
        android:layout_marginRight="25dp"
        android:layout_marginLeft="25dp"
        android:singleLine="true"/>
    <TextView
        android:id="@+id/password1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="25dp"
        android:text="Password"
        android:layout_below="@id/user_name"
        android:textStyle="bold"
        android:textSize="45px"
        android:padding="15dp"
        android:layout_marginTop="25dp"
        android:layout_marginLeft="15dp" />
    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/pass"
        android:layout_below="@+id/password1"
        app:passwordToggleEnabled="true">
        <com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/password"
            android:padding="15dp"
            android:inputType="textPassword"
            android:singleLine="true"
            android:layout_marginRight="25dp"
            android:layout_marginLeft="25dp">
        </com.google.android.material.textfield.TextInputEditText>
    </com.google.android.material.textfield.TextInputLayout>

    <Button
        android:id="@+id/b1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/pass"
        android:layout_marginStart="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:text="SAVE"/>

    <Button
        android:id="@+id/b2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/pass"
        android:layout_toRightOf="@+id/b1"
        android:layout_marginStart="60dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:text="DISPLAY" />

    <TextView
        android:id="@+id/resp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_below="@+id/b1"
```

```
        android:layout_centerHorizontal="true"
        />
</RelativeLayout>
```

Java Code:

```
package com.example.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONArray;
import org.json.JSONException;
public class MainActivity extends AppCompatActivity {
    public EditText e1, e2;
    public Button bl, bn;
    public TextView resp;
    String user, url, password;
      String responseJSON2;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        resp = findViewById(R.id.resp);
        e1 = findViewById(R.id.user_name);
        e2 = findViewById(R.id.password);
        bl = findViewById(R.id.b1);
        bn = findViewById(R.id.b2);
        RequestQueue queue = Volley.newRequestQueue(this);
        bl.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                user = e1.getText().toString();
                password = e2.getText().toString();
                url = "http://192.168.252.88/sample/registration.php?user=" + user + "&password=" + password;
                //resp.setText("Welcome "+user+"\t"+password+"\n"+"ur registration process is completed");

                StringRequest stringRequest = new StringRequest(Request.Method.POST, url, new
Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        if (response.contains("successful")) {
                            Toast.makeText(MainActivity.this, "login successful", Toast.LENGTH_LONG).show();
                        } else {
                            Log.d("valuek", response);
                            Toast.makeText(MainActivity.this, response, Toast.LENGTH_LONG).show();
                        }
                    }
                }, new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        Toast.makeText(MainActivity.this, error.toString(), Toast.LENGTH_LONG).show();
                        Log.d("value1", String.valueOf(error));
                    }
                });
                queue.add(stringRequest);
            }
        });

        bn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                user = e1.getText().toString();
                password = e2.getText().toString();
                url = "http://192.168.252.88/sample/retrieve.php";
                //resp.setText("Welcome "+user+"\t"+password+"\n"+"ur registration process is completed");
                StringRequest stringRequest = new StringRequest(Request.Method.POST, url, new
Response.Listener<String>() {
                    @Override

                    public void onResponse(String response) {
                        try {
                            JSONArray jsonArr = new JSONArray(response);
                            responseJSON2 = response.toString();
                            resp.setText(responseJSON2);
                            }
                        catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                }, new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        Toast.makeText(MainActivity.this, error.toString(), Toast.LENGTH_LONG).show();
                        Log.d("value1", String.valueOf(error));
                    }
                });
                queue.add(stringRequest);
            }
        });
    }
}
```

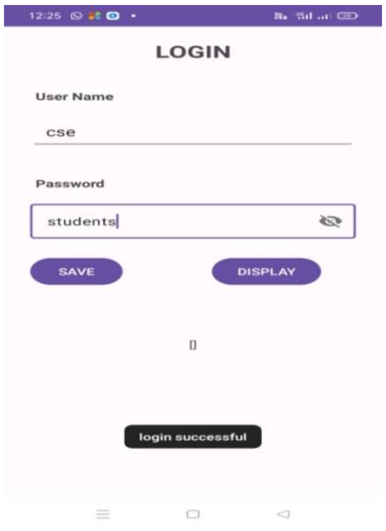
login.php page:

```
<?php
 $con=mysqli_connect("localhost","root","admin","db");
if(!$con){
die("Connection failed");
}
$users =$_REQUEST['user'];
$password =$_REQUEST['password'];
$sql = "insert into login(username,password) values('$users','$password')";
 if (mysqli_query($con,$sql)) {
      echo "Values have been inserted successfully";
   }
else{
echo "not";}

?>
```

Retrieve.php page:

```
<?php
 $con=mysqli_connect("localhost","root","admin","db");
if(!$con){
die("Connection failed");
}

$sql="SELECT username,password FROM login";
$data=array();
$smt=$con->prepare($sql);
$smt->execute();
$smt->bind_result($username,$password);
while($smt->fetch()){
$temp=[$username,$password];
array_push($data,$temp);
}
echo json_encode($data);

?>
```

Output:



LOGIN

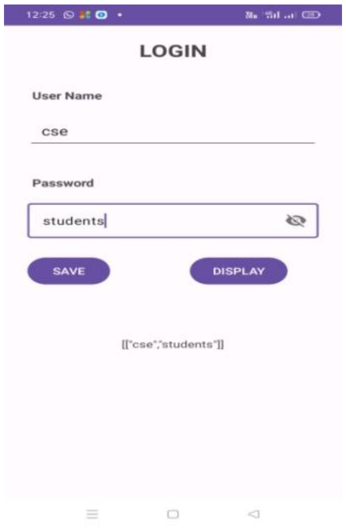
User Name: cse

Password: students

SAVE DISPLAY

```
[]
```

login successful



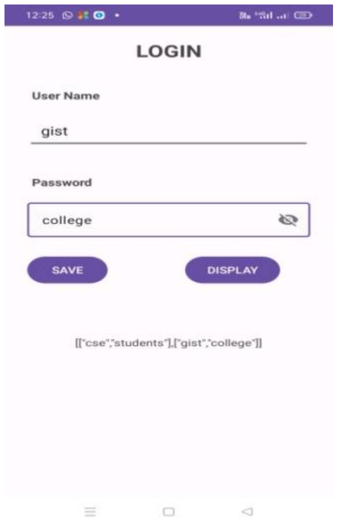
LOGIN

User Name: cse

Password: students

SAVE DISPLAY

```
[["cse","students"]]
```



LOGIN

User Name: gist

Password: college

SAVE DISPLAY

```
[["cse","students"],["gist","college"]]
```

## EXPERIMENT 8

Task: Design and implement an effective student Login System using firebase

Procedure:

Xml Code:

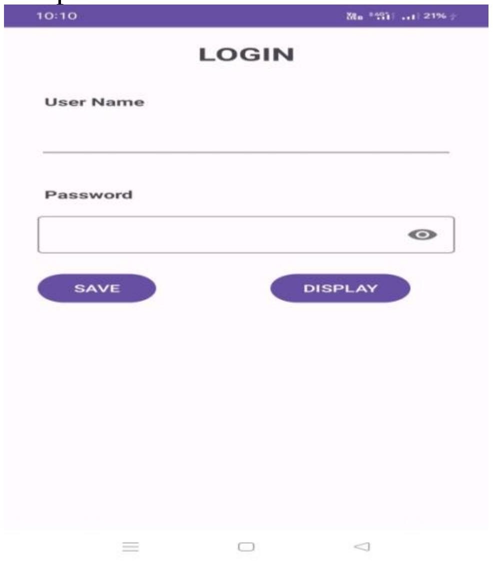
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <TextView
        android:id="@+id/lt1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:gravity="center_horizontal"
        android:text="LOGIN"
        android:textSize="25dp"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="25dp"
        android:text="User Name"
        android:layout_below="@id/lt1"
        android:textStyle="bold"
        android:textSize="45px"
        android:padding="15dp"
        android:layout_marginTop="25dp"
        android:layout_marginLeft="15dp"/>
    <EditText
        android:id="@+id/user_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/username"
        android:padding="15dp"
        android:layout_marginRight="25dp"
        android:layout_marginLeft="25dp"
        android:singleLine="true"/>
    <TextView
        android:id="@+id/password1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="25dp"
        android:text="Password"
        android:layout_below="@id/user_name"
        android:textStyle="bold"
        android:textSize="45px"
        android:padding="15dp"
        android:layout_marginTop="25dp"
        android:layout_marginLeft="15dp" />
    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/pass"
        android:layout_below="@+id/password1"
        app:passwordToggleEnabled="true">
        <com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/password"
            android:padding="15dp"
            android:inputType="textPassword"
            android:singleLine="true"
            android:layout_marginRight="25dp"
            android:layout_marginLeft="25dp">
        </com.google.android.material.textfield.TextInputEditText>
    </com.google.android.material.textfield.TextInputLayout>

    <Button
        android:id="@+id/b1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/pass"
        android:layout_marginStart="25dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:text="SAVE"/>

    <Button
        android:id="@+id/b2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/pass"
        android:layout_toRightOf="@+id/b1"
        android:layout_marginStart="60dp"
        android:layout_marginTop="25dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="25dp"
        android:text="DISPLAY" />

    <TextView
        android:id="@+id/resp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_below="@+id/b1"
```

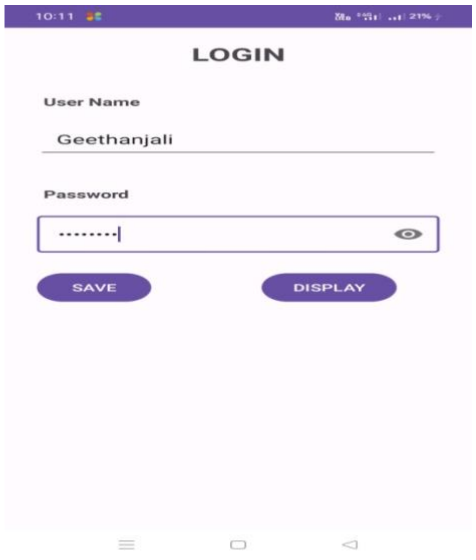
```
        android:layout_centerHorizontal="true"
        />
</RelativeLayout>
```

Java Code:

```
package com.example.myapplication;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.HashMap;
import java.util.Map;
public class MainActivity extends AppCompatActivity {
    public EditText e1, e2;
    public Button bl,bn;
    public TextView resp;
    String ip;
    String user, url, password;

    private DatabaseReference rootdatabase;
    //sessionManager sessionManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //sessionManager = new sessionManager(this);
        resp = findViewById(R.id.resp);
        e1 = findViewById(R.id.user_name);
        e2 = findViewById(R.id.password);
        bl = findViewById(R.id.b1);
        bn = findViewById(R.id.b2);
        rootdatabase = FirebaseDatabase.getInstance().getReference();
        bl.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view){
                user = e1.getText().toString();
                password = e2.getText().toString();
                rootdatabase.setValue(user);
                HashMap hashmap=new HashMap();
                hashmap.put("pass",password);
            hashmap.put("name",user);
            rootdatabase.child(user).setValue(hashmap);
        }
    });
    bn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            rootdatabase.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot snapshot) {
                    if(snapshot.exists()) {
                    String data= snapshot.getValue().toString();
                    resp.setText(data);
                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {

                }
            });
        }
    });
}}
```

Output:





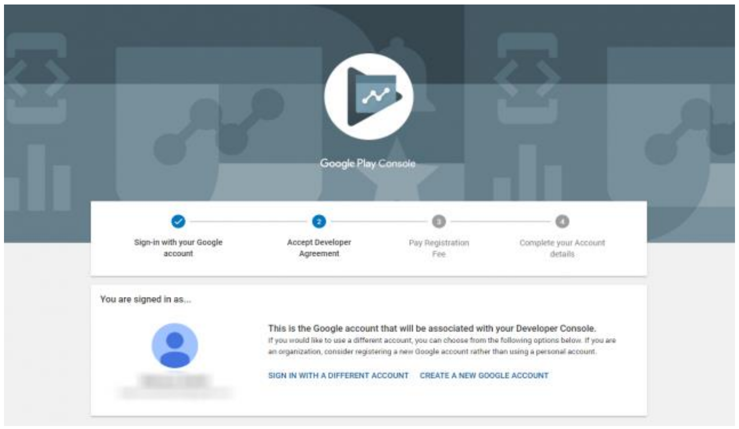
## EXPERIMENT 9

**AIM:** Install Android Studio and Configure Latest Android SDKs and Android Virtual Devices

**Explanation:**

**Step 1: Create a Developer Account**

Before you can publish any app on Google Play, you need to create a Developer Account. You can easily sign up for one using your existing Google Account.


Google Play Console
Sign-in with your Google account
Accept Developer Agreement
Pay Registration Fee
Complete your Account details
SIGN IN WITH A DIFFERENT ACCOUNT
CREATE A NEW GOOGLE ACCOUNT

You are signed in as...

This is the Google account that will be associated with your Developer Console.
If you would like to use a different account, you can choose from the following options below. If you are an organization, consider registering a new Google account rather than using a personal account.

The sign up process is fairly straightforward, and you'll need to pay a one-time registration fee of $25. After you've reviewed and accepted the Developer Distribution Agreement, you can proceed to make the payment using your credit or debit card.

To finish the sign up process, fill out all your necessary account details, including your Developer Name, which will be visible to your customers on Google Play. You can always add more details later.

Also, do remember that it can take up to 48 hours for your registration to be fully processed.

**Step 2: Plan to Sell? Link Your Merchant Account**

If you want to publish a paid app or plan to sell in-app purchases, you need to create a payments center profile, i.e. a merchant account. Here's how you can do that:

1. Sign in to your Play Console
2. Click on Download Reports — Financial
3. Select 'Set up a merchant account now'
4. Fill out your business information

Once you create the profile, it will be automatically linked to your developer account.

A merchant account will let you you manage your app sales and monthly payouts, as well as analyze your sales reports right in your Play Console.

**Step 3: Create an App**

Now that you have set up your Play Console, you can finally add your app. Here's how to do that:

1. Navigate to the 'All applications' tab in the menu
2. Click on 'Create Application'
3. Select your app's default language from the drop-down menu
4. Type in a title for your app
5. Click on "Create"

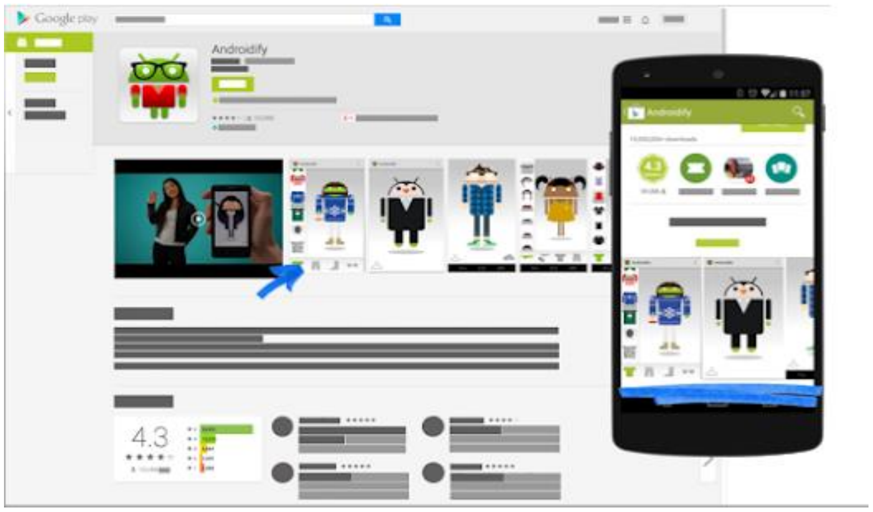
The title of your app will show on Google Play after you've published. Don't worry too much about it at this stage; you can always change the name later.
After you've created your app, you'll be taken to the store entry page. Here, you will need to fill out all the details for your app's store listing.

**Step 4: Prepare Store Listing**
Before you can publish your app, you need to prepare its store listing. These are all the details that will show up to customers on your app's listing on Google Play.
Note: You don't necessarily have to complete this step before moving on to the next one. You can always save a draft and revisit it later when you're ready to publish.
The information required for your store listing is divided into several categories:
**Product Details**
There are three fields here that you need to fill out:

| Field | Description | Character Limit | Notes |
|---|---|---|---|
| Title | Your app's name on Google Play. | 50 character limit | You can add one localized title per language. |
| Short description | The first text users see when looking at your app's detail page on the Play Store app. | 80 character limit | Users can expand this text to view your app's full description. |
| Full description | Your app's description on Google Play. | 4000 character limit | |

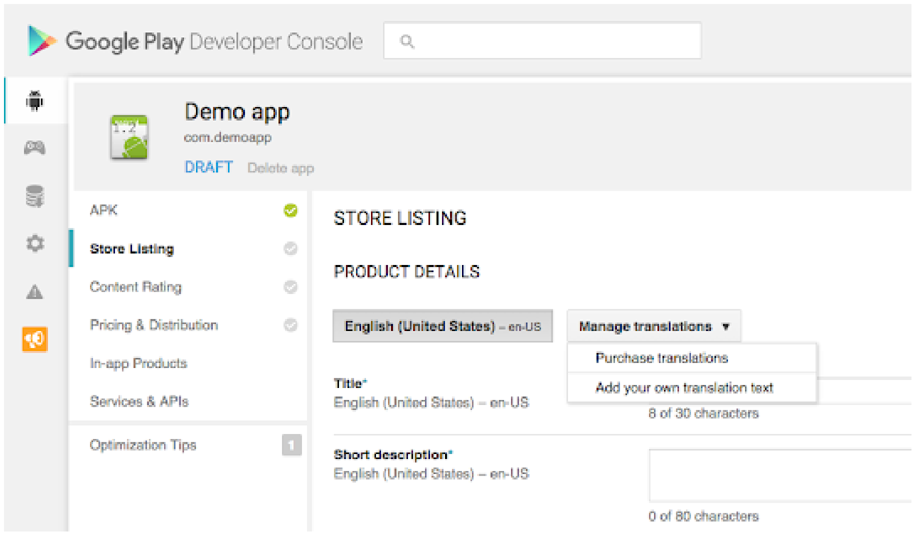
Your app's title and description should be written with a great user experience in mind.
Use the right keywords, but don't overdo it. Make sure your app doesn't come across as spam-y or promotional, or it will risk getting suspended on the Play Store.

**Graphic Assets**
Under graphic assets, you can add screenshots, images, videos, promotional graphics, and icons that showcase your app's features and functionality.



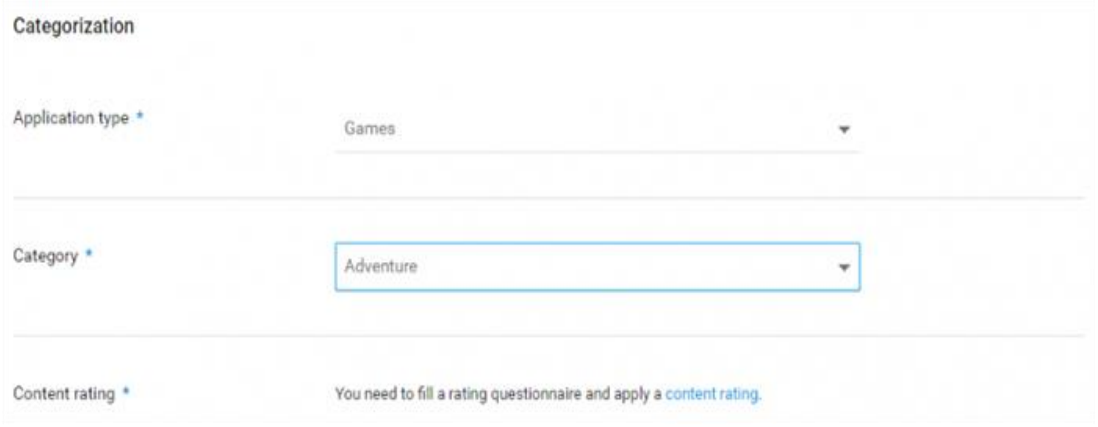
Some parts under graphic assets are mandatory, like screenshots, a feature graphic, and a high resolution icon. Others are optional, but you can add them to make your app look more attractive to users.
There are specific requirements for each graphic asset that you upload, such as the file format and dimensions. You can read more about each requirement here.

**Languages & Translations**
You can also add translations of your app's information in the store listing details, along with in-language screenshots and other localized images.



There's also an option for users to view automated translations of your app's information using Google Translate (with the exception of Armenian, Raeto-romance, Tagalog, and Zulu), in case you don't add your own translations.

**Categorization**
This part requires you to select the appropriate type and category your app belongs to. From the drop-down menu, you can pick either app or game for the application type.

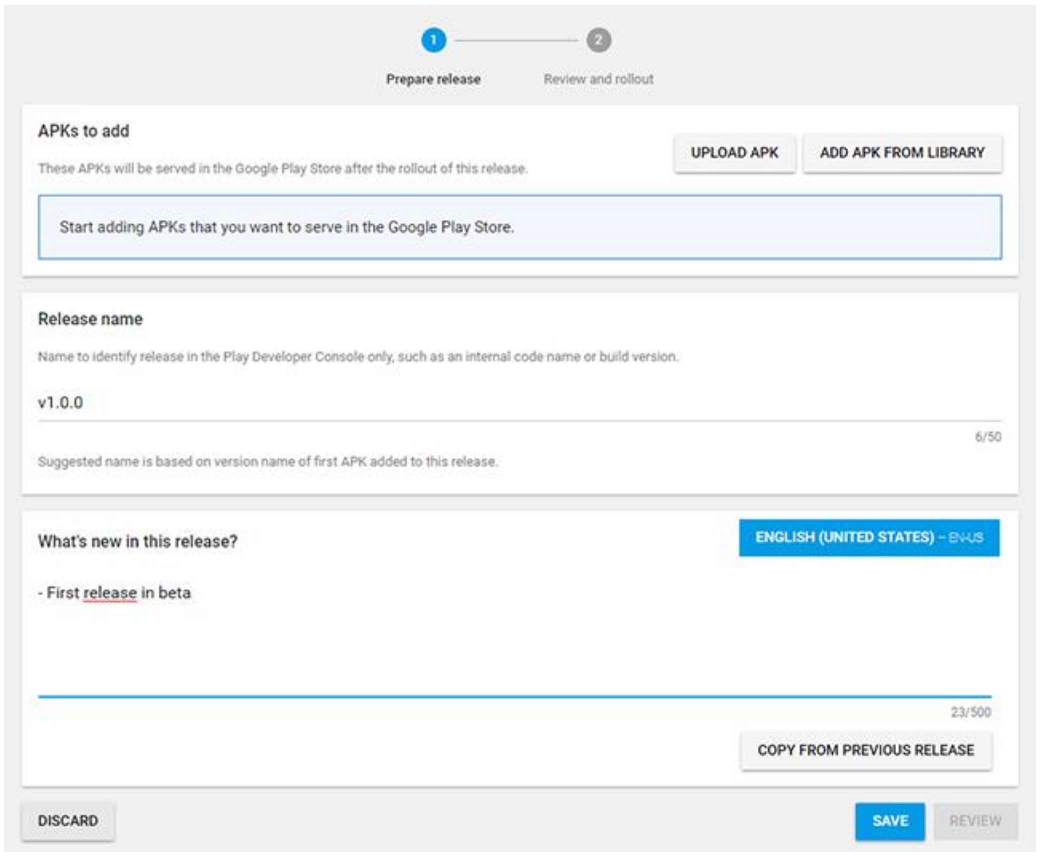


There are various categories for each type of app available on the Play Store. Pick the one your app fits into best.In order to rate your content, you'll need to upload an APK first. You can skip this step for later.
**Contact Details**
This part requires you to enter contact details to offer your customers access to support regarding your app.

You can add multiple contact channels here, like an email, website, and phone number, but providing a contact email is mandatory for publishing an app.

**Privacy Policy**
For apps that request access to sensitive user data or permissions, you need to enter a comprehensive privacy policy that effectively discloses how your app collects, uses, and shares that data.
You must add a URL linking to your privacy policy in your store listing and within your app. Make sure the link is active and relevant to your app.
You're now done with the store listing. Go ahead and click on 'Save Draft' to save your details. You can always skip some steps and come back to them later before you publish your app.

**Step 5: Upload APK to an App Release**
Now that you have prepared the ground to finally upload your app, it's time to dig out your APK file.
The Android Package Kit (or APK, for short) is the file format used by the Android operating system to distribute and install apps. Simply put, your APK file contains all the elements needed for your app to actually work on a device.
Google offers you multiple ways to upload and release your APK. Before you upload the file, however, you need to create an app release.
To create a release, select the app you created in Step 3. Then, from the menu on the left side, navigate to 'Release management' -> 'App releases.'
Here, you need to select the type of release you want to upload your first app version to. You can choose between an internal test, a closed test, an open test, and a production release.
The first three releases allow you to test out your app among a select group of users before you make it go live for everyone to access.
This is a safer option because you can analyze the test results and optimize or fix your app accordingly if you need to before rolling it out to all users.
However, if you create a production release, your uploaded app version will become accessible to everyone in the countries you choose to distribute it in.
Once you've picked an option, click on 'Create release.'
Next, follow the on-screen instructions to add your APK files, and name and describe your release.


Prepare release

Review and rollout

APKs to add

These APKs will be served in the Google Play Store after the rollout of this release.

Start adding APKs that you want to serve in the Google Play Store.

Release name

Name to identify release in the Play Developer Console only, such as an internal code name or build version.

v1.0.0

6/50

Suggested name is based on version name of first APK added to this release.

What's new in this release?

- First release in beta

23/500

After you're done, press Save.

**Step 6: Provide an Appropriate Content Rating**

If you don't assign a rating to your app, it will be listed as 'Unrated'. Apps that are 'Unrated' may get removed from Google Play.

To rate your app, you need to fill out a content rating questionnaire. You can access it when you select your app in the Play Console, and navigate to 'Store presence' — 'Content rating' on the left menu.

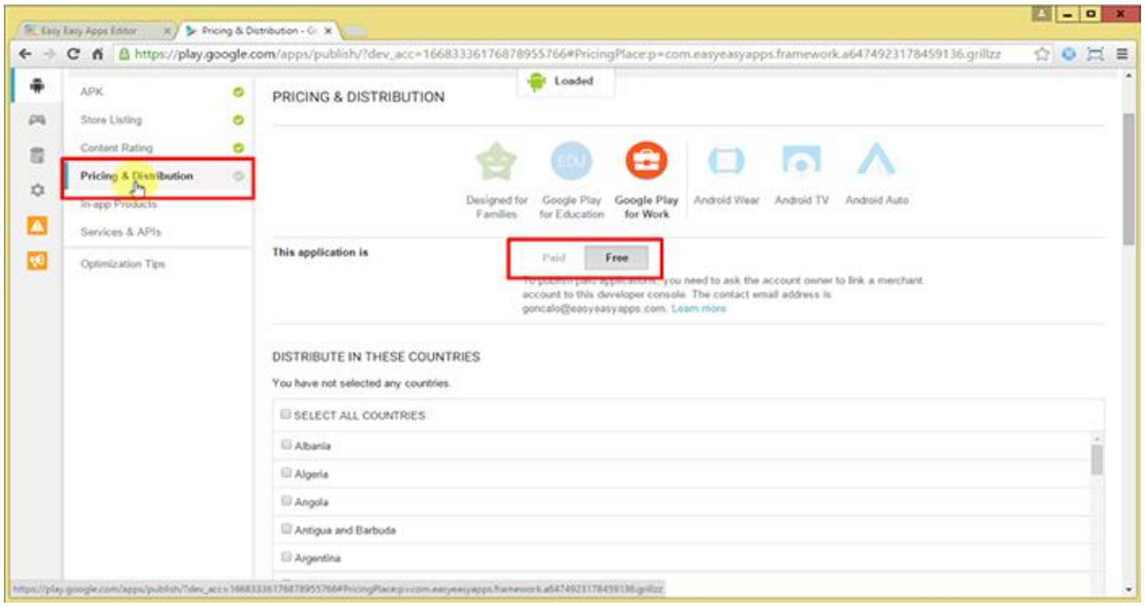
Make sure you enter accurate information. Misrepresentation of your app's content can lead to suspension or removal from the Play Store.

An appropriate content rating will also help you get to the right audience, which will eventually improve your engagement rates.

**Step 7: Set Up Pricing & Distribution**

Before you can fill out the details required in this step, you need to determine your app's monetization strategy.

Once you know how your app is going to make money, you can go ahead and set up your app as free or paid.



PRICING & DISTRIBUTION

This application is

To publish paid applications, you need to ask the account owner to link a merchant account to this developer console. The contact email address is goncalo@easyeasyapps.com. Learn more

DISTRIBUTE IN THESE COUNTRIES

You have not selected any countries.

Remember, you can always change your app from paid to free later, but you cannot change a free app to paid. For that, you'll need to create a new app and set its price.
You can also choose the countries you wish to distribute your app in, and opt-in to distribute to specific Android devices and programs too.

**Step 8: Rollout Release to Publish Your App**
You're almost done. The final step involves reviewing and rolling out your release after making sure you've taken care of everything else.
Before you review and rollout your release, make sure the store listing, content rating, and pricing and distribution sections of your app each have a green check mark next to them.
Once you're sure you've filled out those details, select your app and navigate to 'Release management' — 'App releases.' Press 'Edit release' next to your desired release, and review it.
Next, click on 'Review' to be taken to the 'Review and rollout release' screen. Here, you can see if there are any issues or warnings you might have missed out on.
Finally, select 'Confirm rollout.' This will also publish your app to all users in your target countries on Google Play.

## Additional Experiment 1

**AIM:** Create the Application to play the Audio clip

**Procedure**:

Xml Code:

```
<?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">
    <TextView
        android:id="@+id/response"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textColor="#700"
        android:layout_marginTop="100dp"
        />
    <ToggleButton
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textOn="stop"
        android:textOff="play"
        />
   </LinearLayout>
```

**Java Code:**

we open MainActivity and add the code to initiate the video view and create an object of MediaController to control the video playback.

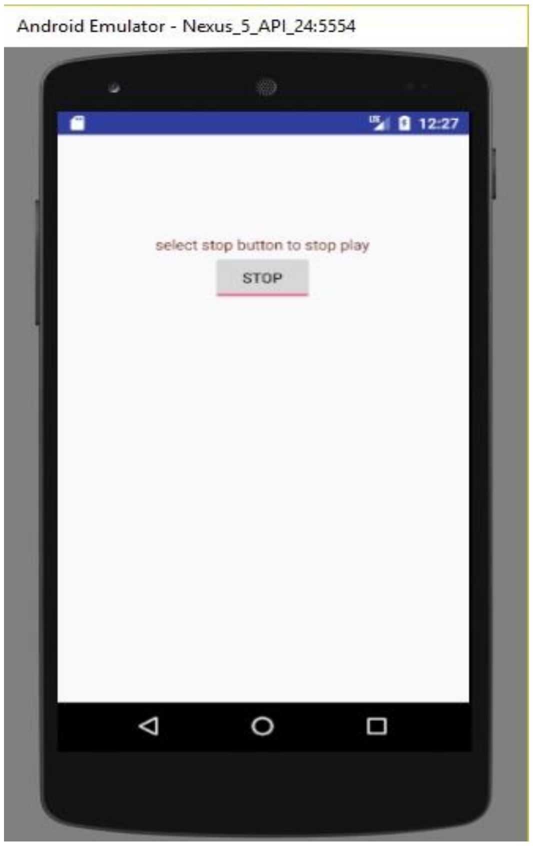
Add raw directory under resource and paste audio under raw directory

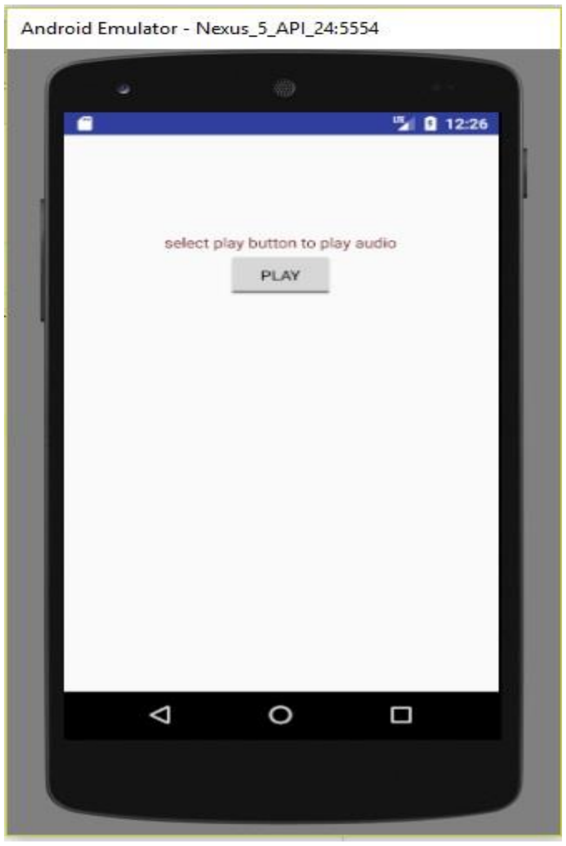
In this class we also set the uri for the video and perform set on error and completion listener events and display Toast message when video is completed or an error is occur while playing thee video.

Also make sure to create a new directory in res folder and name it raw. Save audio  name song.mp3 in raw folder. We will be setting path to this audio clip

```
package com.sumanthsvs.audioclip;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ToggleButton;
import android.view.View;
import android.widget.TextView;
import android.media.MediaPlayer;
import android.view.View.OnClickListener;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Find your VideoView in your video_main.xml layout
        final TextView response = (TextView) findViewById(R.id.response);
        response.setText("select play button to play audio");
        final MediaPlayer mp = MediaPlayer.create(MainActivity.this, R.raw.abc);
        final ToggleButton button = (ToggleButton) findViewById(R.id.btn);

        button.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                if (button.isChecked()) {
                    response.setText("select stop button to stop play");
                    mp.start();
                } else {
                    response.setText("select play button to play audio");
                    mp.pause();
                }
            }



        } );
    }
}
```

**OUTPUT:**


Android Emulator - Nexus_5_API_24:5554
12:27
select stop button to stop play
STOP


Android Emulator - Nexus_5_API_24:5554
12:26
select play button to play audio
PLAY

## Additional Experiment 2

**AIM:** Design the Application for Menus

Procedure:

Xml code:

```
<?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="com.sumanthsvs.menuactiobar.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

**Java Code:**

```
package com.sumanthsvs.menuactiobar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu,menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()){
            case R.id.settings:
                Toast.makeText(this,"Settings Selected" , Toast.LENGTH_LONG).show();
                return true;
            case R.id.share:
                Toast.makeText(this,"Share Selected" , Toast.LENGTH_LONG).show();
                return true;
            case R.id.refresh:
                Toast.makeText(this,"Refresh Selected" , Toast.LENGTH_LONG).show();
                return true;
            case R.id.search:
                Toast.makeText(this,"Search Selected" , Toast.LENGTH_LONG).show();
                return true;
            case R.id.help:
                Toast.makeText(this,"Help Selected" , Toast.LENGTH_LONG).show();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }

    }
}
```

In this step we show string file which is used to store string data of an app.

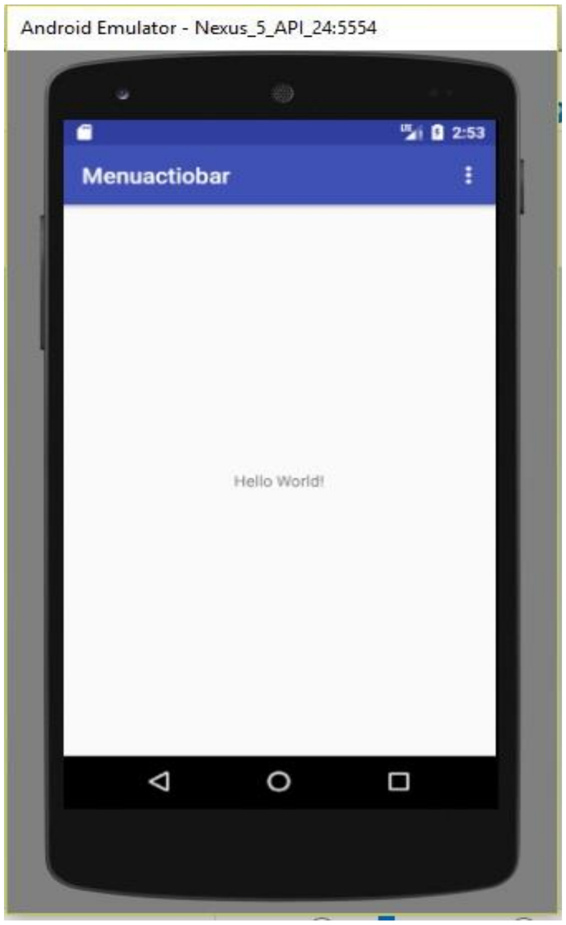
```
<resources>
    <string name="app_name">Menuactiobar</string>
    <string name="settings">Settings</string>
    <string name="share">Share</string>
    <string name="search">Search</string>
    <string name="refresh">Refresh</string>
    <string name="help">Help</string>
</resources>
```

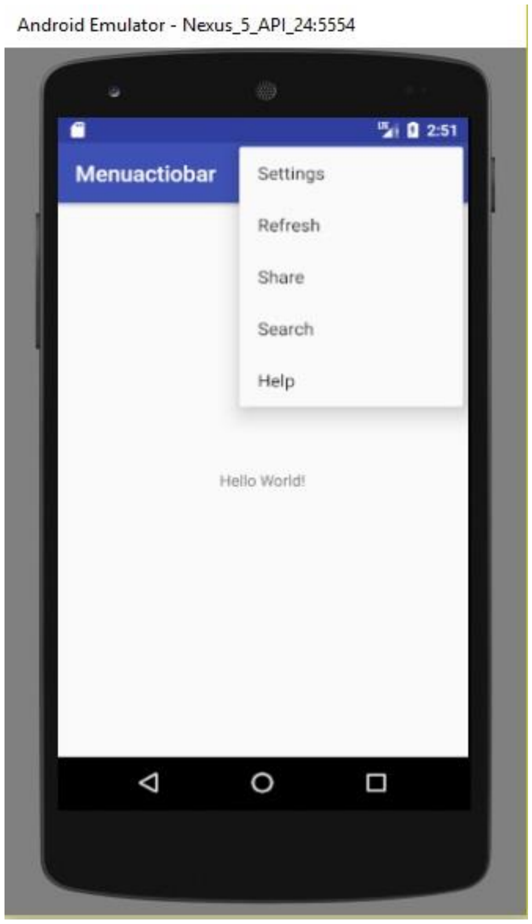
Add menu resource directory then add menu resource file
Open res ->menu-> Main_menu.xml
In this step we show string file which is used to store string data of an app.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/settings"
        app:showAsAction="never"
        android:title="@string/settings" />
    <item
        android:id="@+id/refresh"
        app:showAsAction="never"
        android:title="@string/refresh" />
    <item
```

```
        android:id="@+id/share"
        app:showAsAction="never"
        android:title="@string/share" />
    <item
        android:id="@+id/search"
        app:showAsAction="never"
        android:title="@string/search" />
    <item
        android:id="@+id/help"
        app:showAsAction="never"
        android:title="@string/help" />
</menu>
```

**OUTPUT:**


Android Emulator - Nexus_5_API_24:5554
2:53
Menuactiobar
Hello World!


Android Emulator - Nexus_5_API_24:5554
2:51
Menuactiobar
Settings
Refresh
Share
Search
Help
Hello World!