Introduction to Android Programming

Overview of android

- Android is a mobile operating system developed primarily by Google, although it's also been open-sourced and developed by an array of other companies and individuals through the Android Open Source Project (AOSP).
- The purpose of Android was to provide a flexible, customizable, and open-source platform for mobile devices, particularly smartphones and tablets, unlike its major competitor at the time, iOS.
- Android's architecture is built upon the Linux kernel, and it has grown to become the most widely used operating system on mobile devices globally.

History of android

- **2003:** The history of Android began in 2003 when Andy Rubin, Rich Miner, Nick Sears, and Chris White founded Android Inc. in Palo Alto, California.
- Their vision was to create an advanced operating system for digital cameras.

History of android ….

- **2005:** Google acquired Android Inc., and this acquisition marked the beginning of Android's evolution into the powerhouse it is today.
- The very first release of Android, in **2007**, already included the open-source aspect through the Android Open Source Project (AOSP).

Google

History of android ….

- Google launched first commercial android, HTC Dream (T-Mobile G1) in **2008** with Android 1.0

First android phone

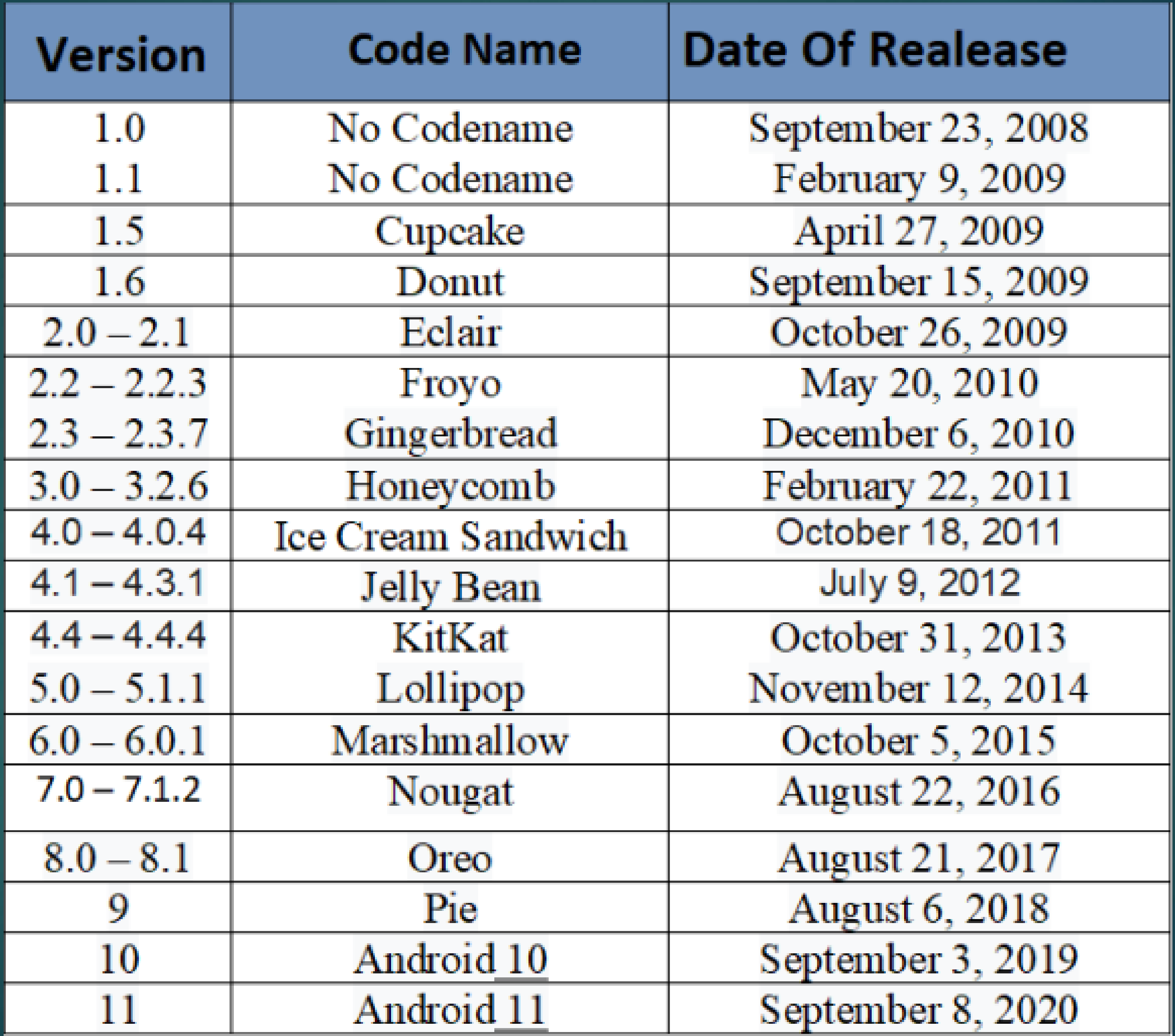
First Iphone

History of android ....

- Android 1.5 Cupcake (April **2009**) brought on-screen keyboards and support for third-party widgets.

History of android ….

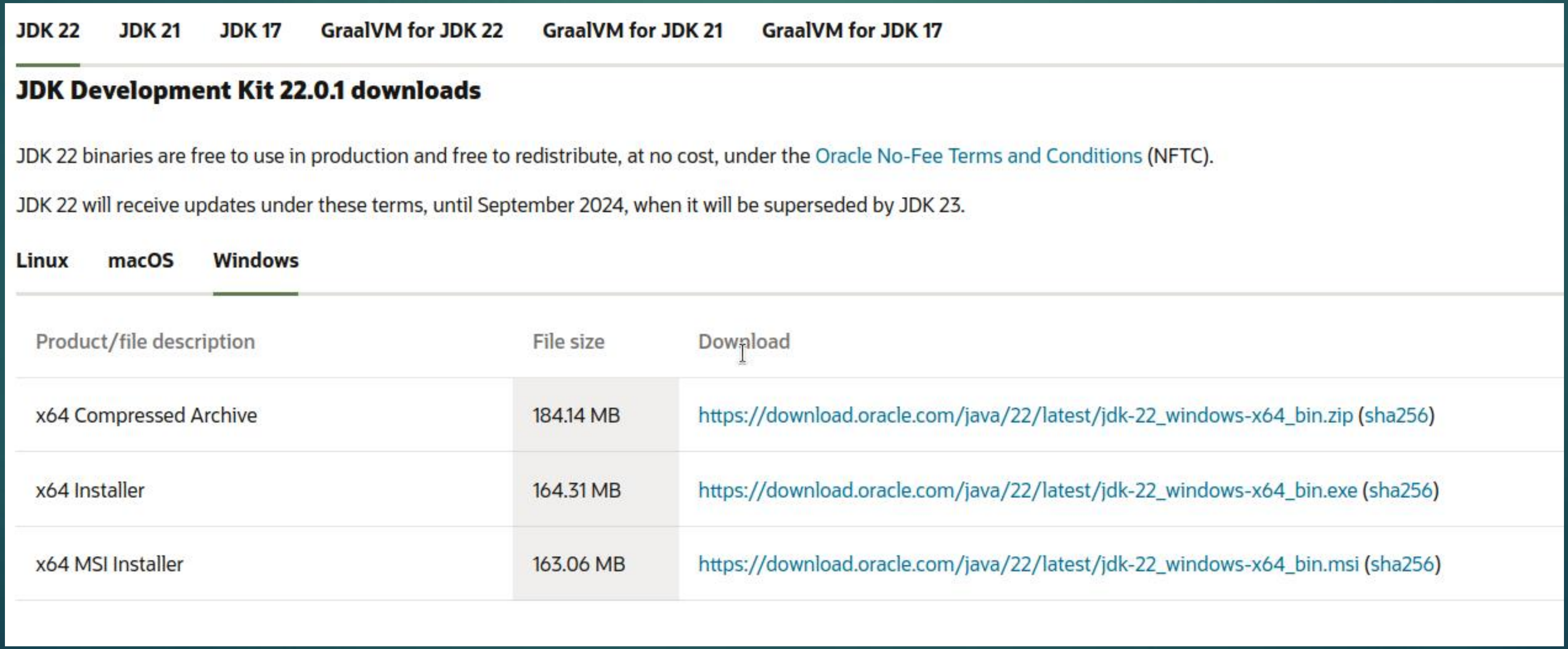
- Android's evolution has not been limited to smartphones. It has expanded to power a wide range of devices, including tablets, smartwatches, smart TVs, and automotive infotainment systems.



Version	Code Name	Date Of Realease
1.0	No Codename	September 23, 2008
1.1	No Codename	February 9, 2009
1.5	Cupcake	April 27, 2009
1.6	Donut	September 15, 2009
2.0 – 2.1	Eclair	October 26, 2009
2.2 – 2.2.3	Froyo	May 20, 2010
2.3 – 2.3.7	Gingerbread	December 6, 2010
3.0 – 3.2.6	Honeycomb	February 22, 2011
4.0 – 4.0.4	Ice Cream Sandwich	October 18, 2011
4.1 – 4.3.1	Jelly Bean	July 9, 2012
4.4 – 4.4.4	KitKat	October 31, 2013
5.0 – 5.1.1	Lollipop	November 12, 2014
6.0 – 6.0.1	Marshmallow	October 5, 2015
7.0 – 7.1.2	Nougat	August 22, 2016
8.0 – 8.1	Oreo	August 21, 2017
9	Pie	August 6, 2018
10	Android 10	September 3, 2019
11	Android 11	September 8, 2020

Setting Up Android Environment

- JDK is required to setup android development environment.
- JDK includes all the Java tools, executables, and binaries that are needed to run Java programs. This includes JRE, a compiler, a debugger, an archiver, and other tools that are used in Java development.
- To download JDK visit: https://www.oracle.com/java/technologies/downloads/#jdk22-windows



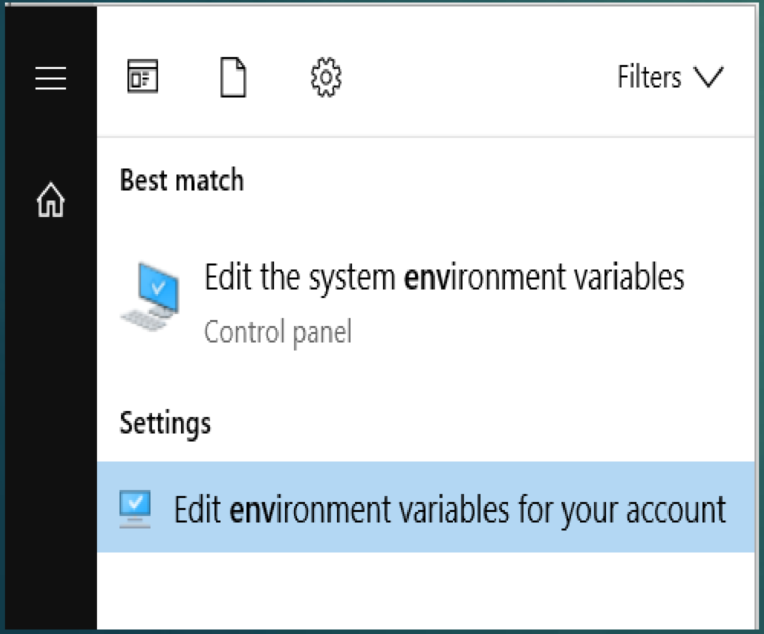
JDK Development Kit 22.0.1 downloads

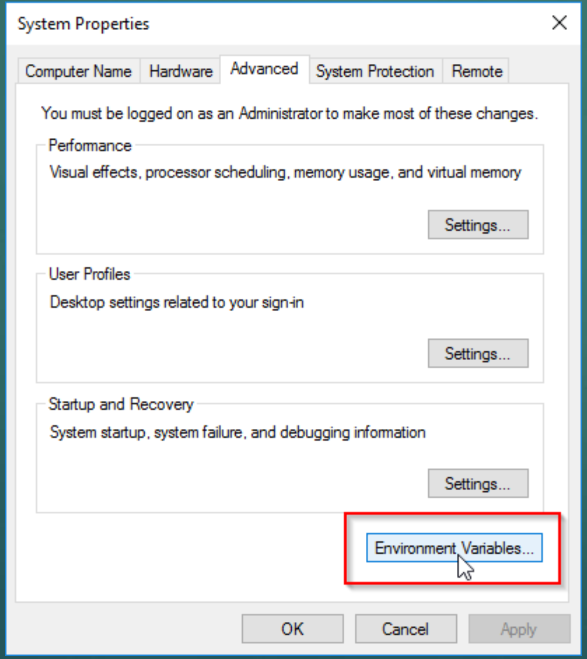
JDK 22 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

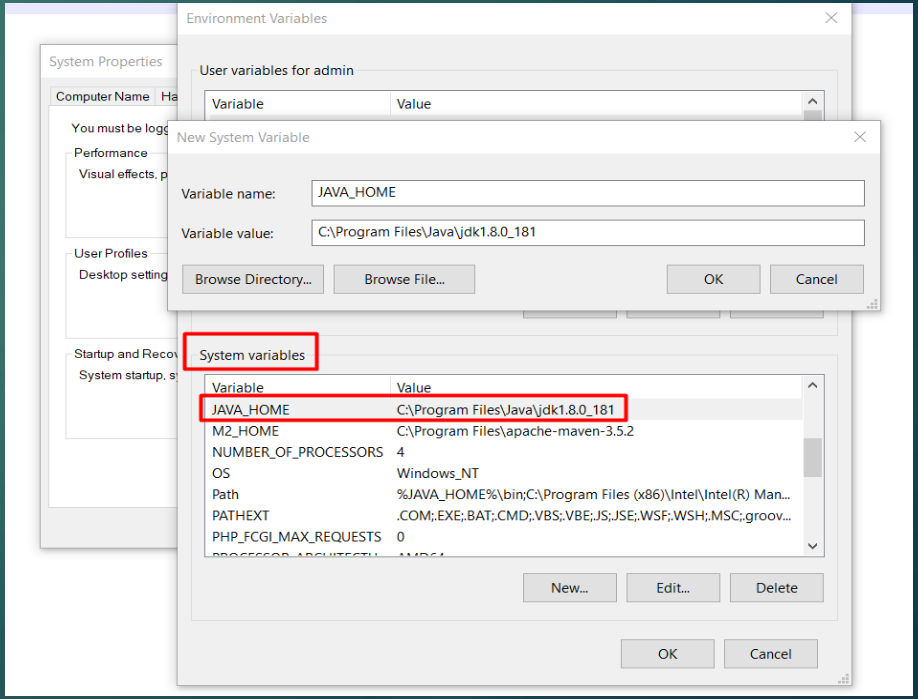
JDK 22 will receive updates under these terms, until September 2024, when it will be superseded by JDK 23.

Product/file description	File size	Download
x64 Compressed Archive	184.14 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.zip (sha256)
x64 Installer	164.31 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.06 MB	https://download.oracle.com/java/22/latest/jdk-22_windows-x64_bin.msi (sha256)

- Search "*edit environment variable*" in search bar and click on "*Environment variable*".
- Click Environment Variables ->Add new variable as "*JAVA_HOME*" => "*JDK Path (C:\Program Files\Java\jdk-22)" excluding bin.*
- *Then go on system variables ->click on Path -> click Edit ->click New and paste paste the path where JDK is installed (C:\Program Files\Java\jdk-22\bin) .Then click ok -> ok.*


Filters
Best match
Edit the system environment variables
Control panel
Settings
Edit environment variables for your account

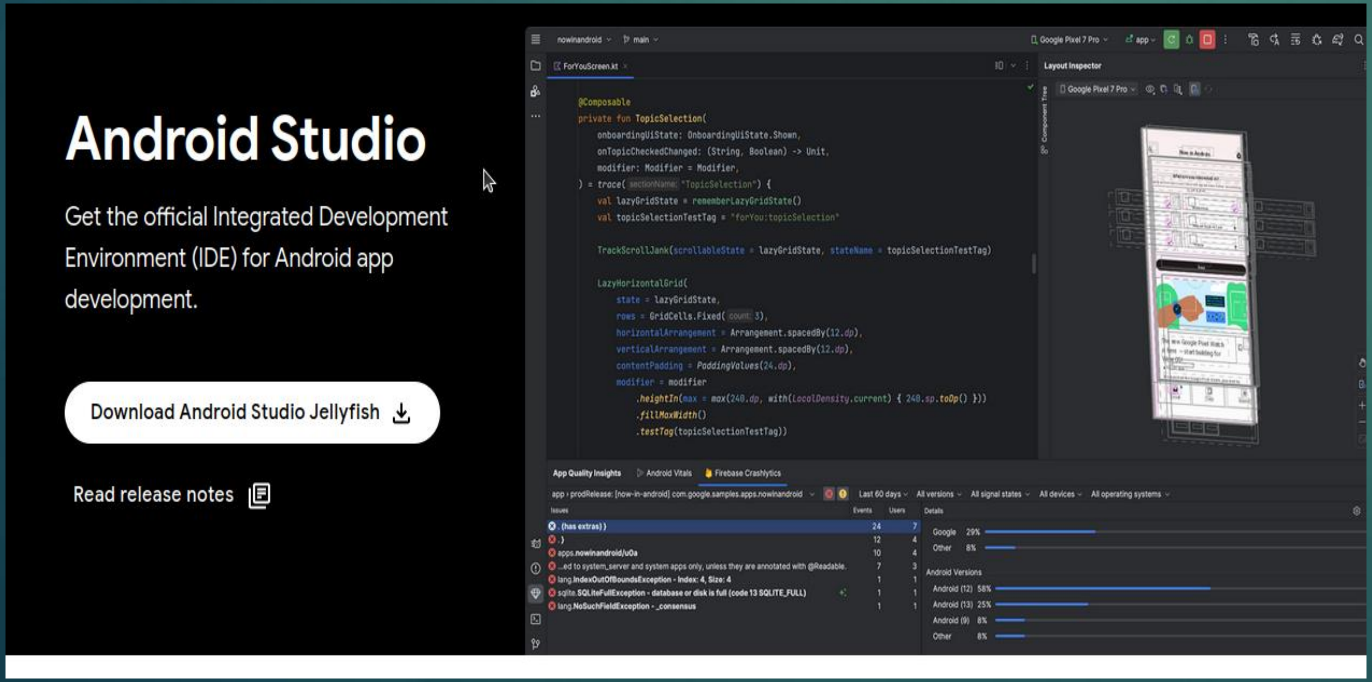

System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply



Variable name: JAVA_HOME
Variable value: C:\Program Files\Java\jdk1.8.0_181

Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_181
M2_HOME	C:\Program Files\apache-maven-3.5.2
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	%JAVA_HOME%\bin;C:\Program Files (x86)\Intel\Intel(R) Man...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.groov...
PHP_FCGI_MAX_REQUESTS	0

- Download latest version of android studio from https://developer.android.com/studio
- Run the downloaded installer and follow the on-screen instructions. It's recommended to keep the default installation options.
- Follow this video tutorial for further details https://www.youtube.com/watch?v=zIN2v3n5-Cs



Android Studio

Get the official Integrated Development Environment (IDE) for Android app development.

```
@Composable
private fun TopicSelection(
    onboardingUiState: OnboardingUiState.Shown,
    onTopicCheckedChanged: (String, Boolean) -> Unit,
    modifier: Modifier = Modifier,
) = trace( sectionName: "TopicSelection") {
    val lazyGridState = rememberLazyGridState()
    val topicSelectionTestTag = "forYou:topicSelection"

    TrackScrollJank(scrollableState = lazyGridState, stateName = topicSelectionTestTag)

    LazyHorizontalGrid(
        state = lazyGridState,
        rows = GridCells.Fixed( count: 3),
        horizontalArrangement = Arrangement.spacedBy(12.dp),
        verticalArrangement = Arrangement.spacedBy(12.dp),
        contentPadding = PaddingValues(24.dp),
        modifier = modifier
            .heightIn(max = max(240.dp, with(LocalDensity.current) { 240.sp.toDp() }))
            .fillMaxWidth()
            .testTag(topicSelectionTestTag))
```

app › prodRelease: [now-in-android] com.google.samples.apps.nowinandroid · Last 60 days · All versions · All signal states · All devices · All operating systems

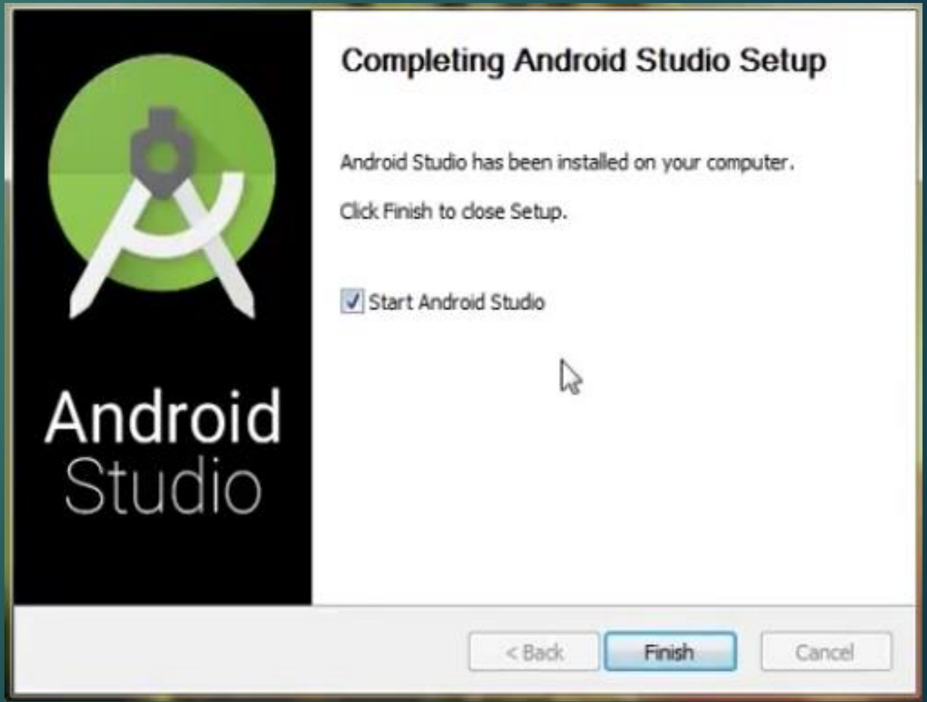
Issues	Events	Users
. (has extras))	24	7
.)	12	4
apps.nowinandroid/u0a	10	4
...ed to system_server and system apps only, unless they are annotated with @Readable.	7	3
lang.IndexOutOfBoundsException - Index: 4, Size: 4	1	1
sqlite.SQLiteFullException - database or disk is full (code 13 SQLITE_FULL)	1	1
lang.NoSuchFieldException - _consensus	1	1

Details

Google	29%
Other	8%

Android Versions

Android (12)	58%
Android (13)	25%
Android (9)	8%
Other	8%



Completing Android Studio Setup

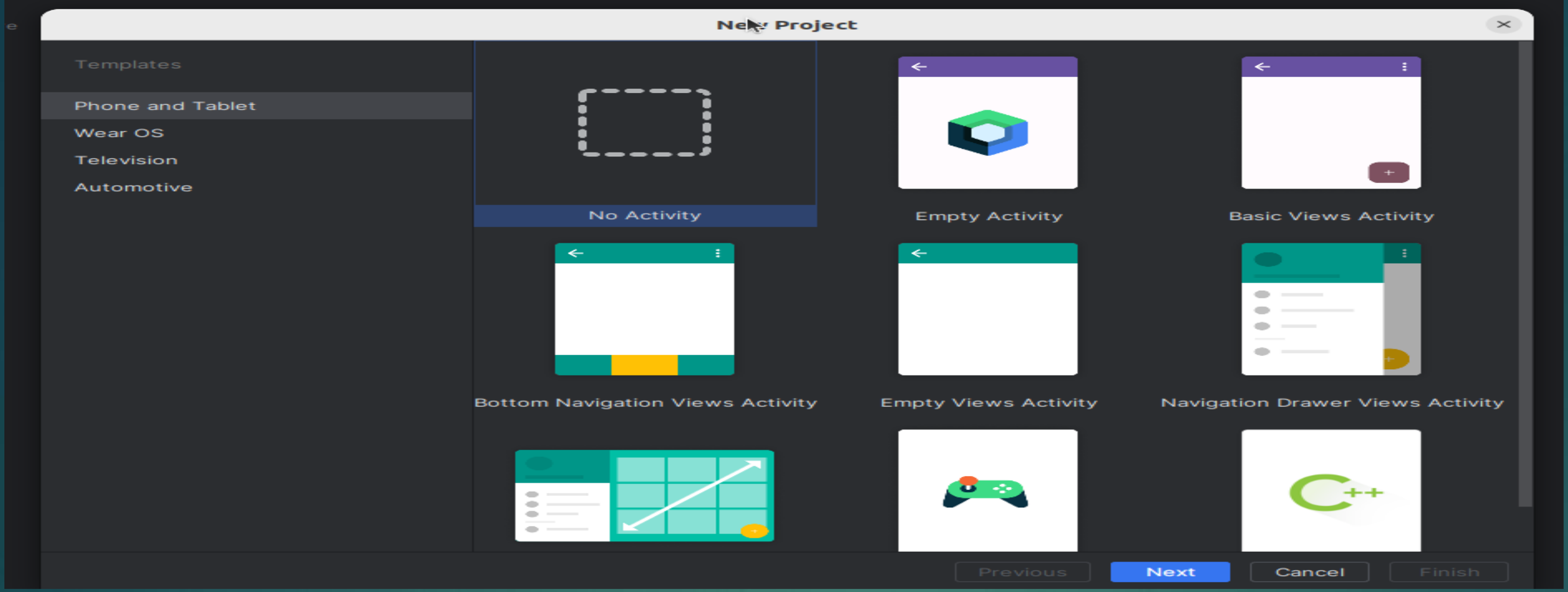
Android Studio has been installed on your computer.

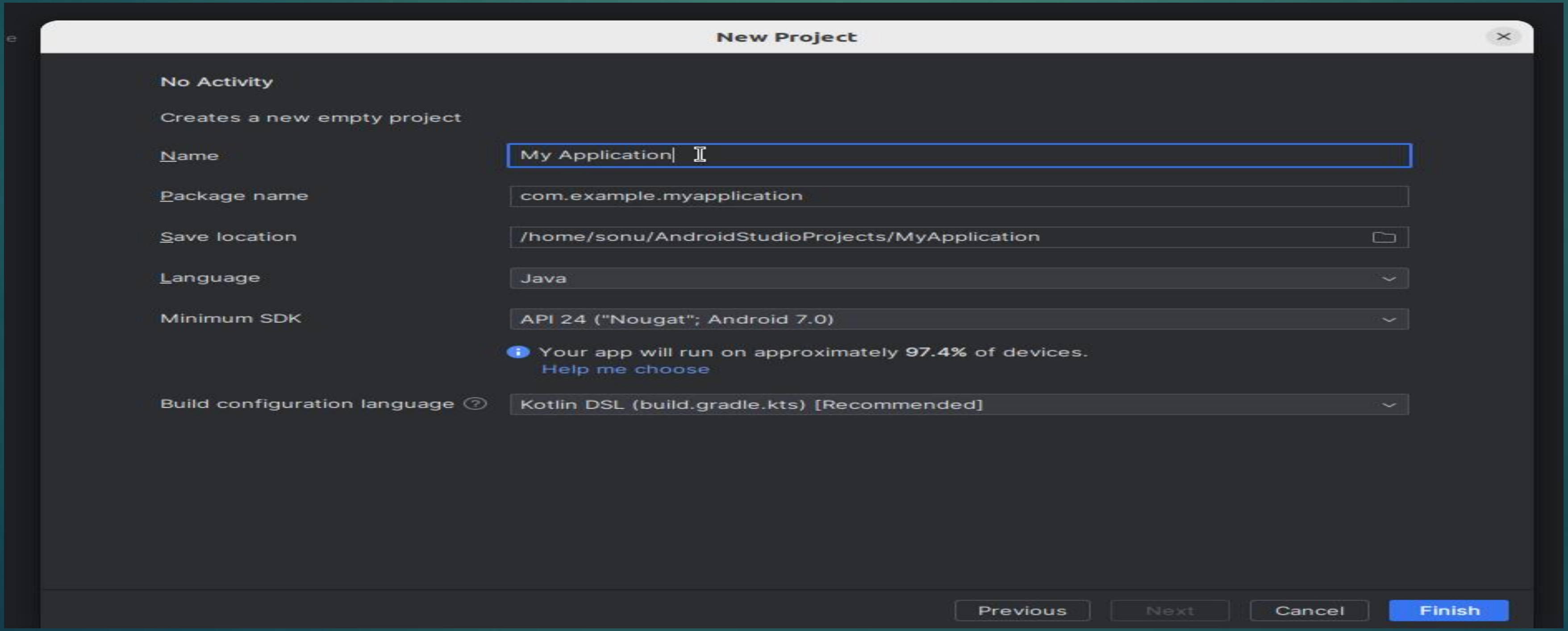
Click Finish to close Setup.

☑ Start Android Studio

Creating android project

- Start a New Project: Click on NEW PROJECT from the welcome screen or goto File -> New ->New Project from current project.
- Select a Project Template: Choose a template, like No Activity for easy setup
- Configure Your Project:
  - Name: Enter your app's name.
  - Package Name: Enter a unique package name (e.g., com.example.myfirstapp).
  - Save Location: Choose the location to save your project.
  - Language: Select Java.
  - Minimum SDK: Choose the lowest version of Android your app will support.
- Finish: Click Finish to create your project and wait till gradle is building.




New Project

No Activity

Creates a new empty project

Name: My Application

Package name: com.example.myapplication

Save location: /home/sonu/AndroidStudioProjects/MyApplication

Language: Java

Minimum SDK: API 24 ("Nougat"; Android 7.0)

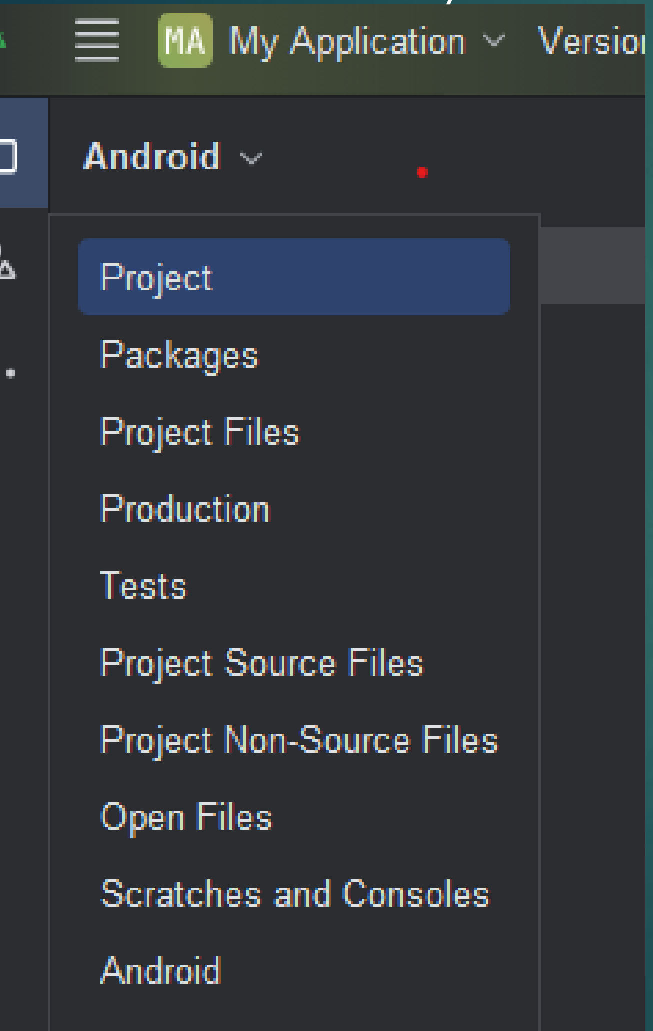
Your app will run on approximately **97.4%** of devices.
Help me choose

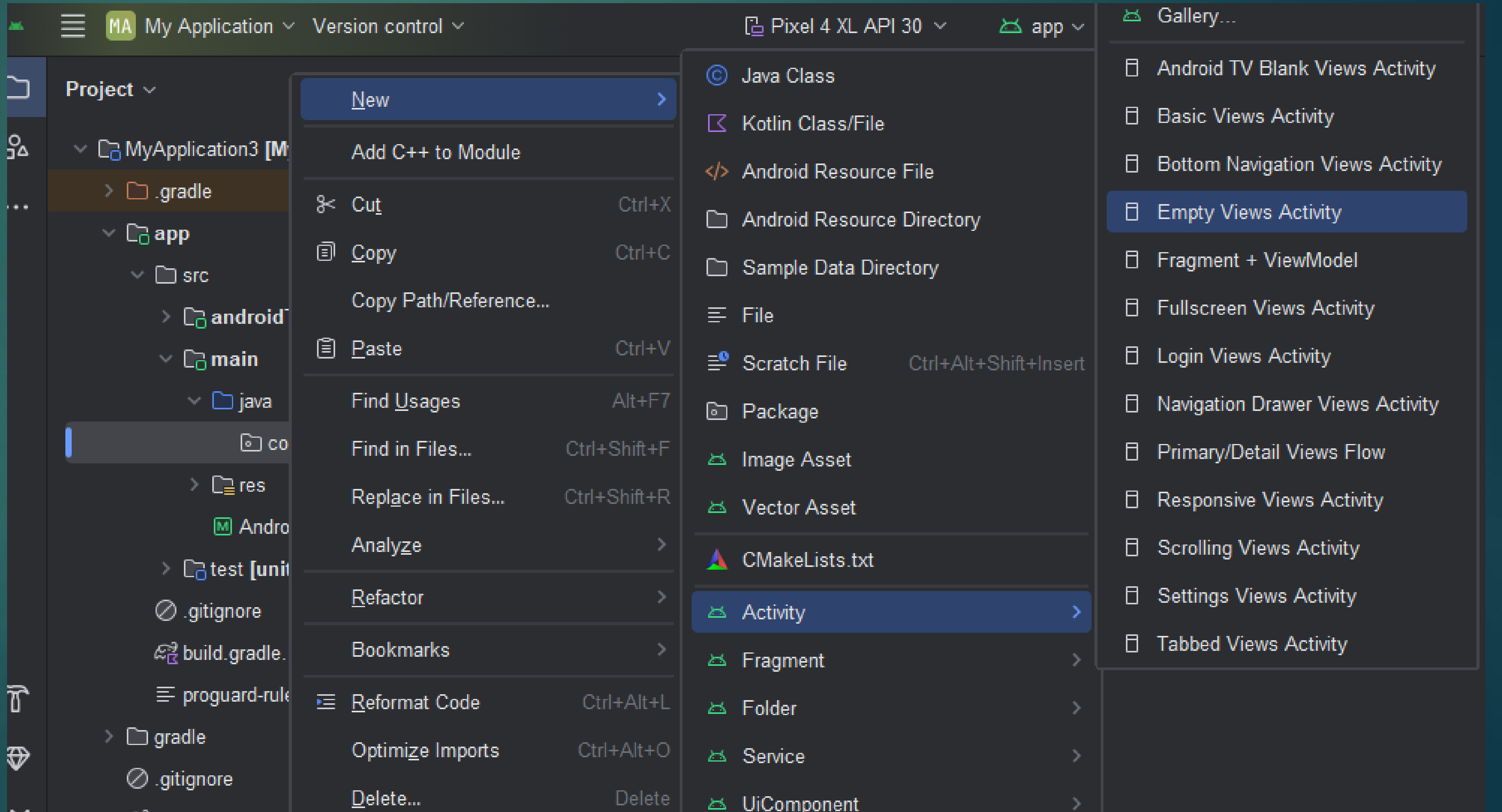
Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

Setting up the new project

- Go to project if it is in Android as shown in figure.
- Create MainActivity. Goto ProjectName ->app ->src ->main ->

  java ->com.example.myapplication and right click there and goto ->New ->Activity ->Empty Views Activity and click on FINISH





Setting up the new project

- Delete Unnecessary file from MainActivity.

```
package com.example.myapplication;

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

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle b) {
        super.onCreate(b);

        setContentView(R.layout.activity_main);

    }
}
```

Setting up the new project

- Add <intent-filter> element in AndroidManifest.xml.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication“>
        <activity
            android:name=".MainActivity"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

.

What is an Intent Filter?

An intent-filter in Android is a way to declare what kinds of intents your app can handle. Intents are messaging objects you can use to request an action from another app component. When an intent is sent, the Android system matches it with the intent filters of the available activities (or other components like services or broadcast receivers) and launches the appropriate one.

<intent-filter>: Declares that activity can respond to specific intents & makes MainActivity the launcher activity, which is the first screen the user sees.

Main Components of intent-filter:

1. **Action (<action>)**: Specifies the action that your activity can handle, like android.intent.action.MAIN, which indicates the main entry point.

2. **Category (<category>)**: Defines additional information about the intent. For example, android.intent.category.LAUNCHER means the activity can be started from the launcher (app drawer).

Note : With the intent-filter in the AndroidManifest.xml, we define that MainActivity should be launched when the user taps the app icon.