Lesson 1: Additional note on Mobile Application Development

Basic terminologies

Mobility is the ability to move freely

A mobile app is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers.

Mobile application development is the process to making software for smartphones and digital assistants, most commonly for Android and iOS. The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser. The programming and markup languages used for this kind of software development include Java, Swift, C# and HTML5.

Types of Mobile Applications

Mobile Applications are of types: Native Apps, Web-based Apps, and Hybrid Apps.

1. Native Apps

Native Apps are designed for specific platforms like Android, IOS, and Windows. These apps can run on mobile devices only. Users can quickly access and easily operate the functions.

Technologies used: Java, Kotlin, Python, Swift, Objective-C, C++, and React

Examples of native apps; WhatsApp, Instagram, Twitter, Spotify, Magento 2 POS, Pokemon Go, Waze, SoundCloud etc

Advantages

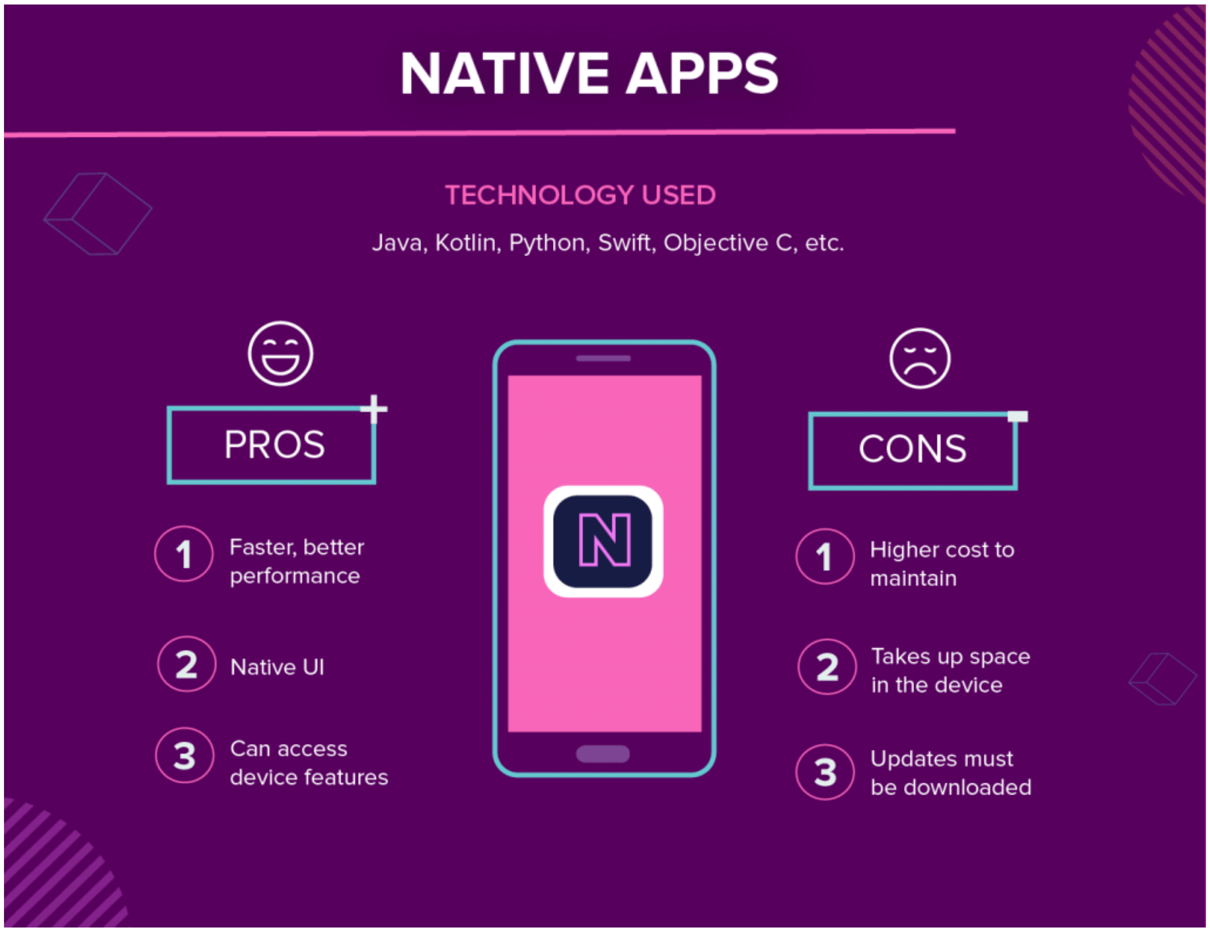
i. The main advantage is high performance and provides an excellent experience to users.
ii. Users can download it from Apps stores, and also updates are available.
iii. More efficient with the device's resources than other types of mobile apps
iv. Native apps utilize the native device UI, giving users a more optimized customer experience.

v. And because native apps connect with the device's hardware directly, they have access to a broad choice of device features like Bluetooth, phonebook contacts, camera roll, NFC, and more.

Disadvantages

However, the problem with native apps lies in the fact that if you start developing them, you have to duplicate efforts for each of the different platforms. The code you create for one platform cannot be reused on another. This drives up costs. Not to mention the effort needed to maintain and update the codebase for each version.

And then, every time there's an update to the app, the user has to download the new file and reinstall it. This also means that native apps do take up precious space in the device's storage.


NATIVE APPS
TECHNOLOGY USED
Java, Kotlin, Python, Swift, Objective C, etc.
PROS
1 Faster, better performance
2 Native UI
3 Can access device features
CONS
1 Higher cost to maintain
2 Takes up space in the device
3 Updates must be downloaded

2. Hybrid Apps

Hybrid apps are combinations of both native and web apps, but wrapped within a native app, giving it the ability to have its own icon or be downloaded from an app store.

The technology used: Hybrid apps use a mixture of web technologies and native APIs. They're developed using: Ionic, Objective C, Swift, HTML5, and others

It contains features of both Native and Web Apps. The apps are wrapped in native app code, and data can be accessed from the server only.

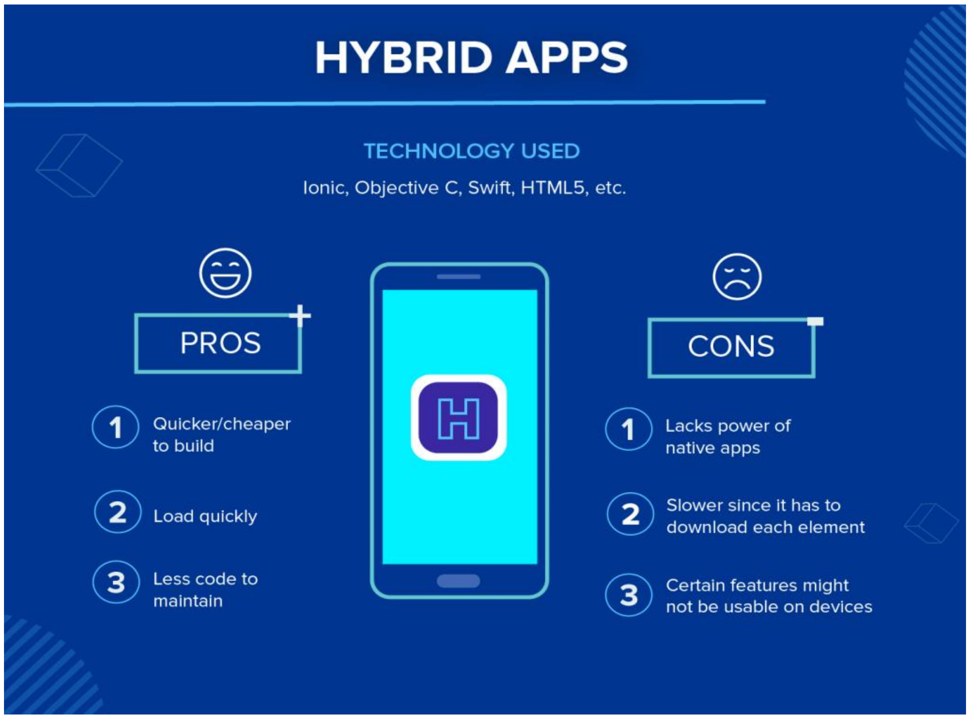
Examples of hybrid apps: Evenote, cryptochange, Justwatch, Uber, Amazon app store, apple app store, etc

Advantages

i. Building a hybrid app is much quicker and more economical than a native app. As such, a hybrid app can be the minimum viable product – a way to prove the viability of building a native app. They also load rapidly, are ideal for usage in countries with slower internet connections, and give users a consistent user experience.
ii. Finally, because they use a single code base, there is much less code to maintain.
iii. Have access to multi-platforms

Disadvantages

i. Compared to native apps, hybrid apps lack performance and speed
ii. They don't provide a user-friendly experience due to cross-platform.



HYBRID APPS

TECHNOLOGY USED

Ionic, Objective C, Swift, HTML5, etc.

PROS

1 Quicker/cheaper to build

2 Load quickly

3 Less code to maintain

CONS

1 Lacks power of native apps

2 Slower since it has to download each element

3 Certain features might not be usable on devices

3. **Web-Based Apps**

Web-Based Apps are not real apps; they are websites that behave like native apps and use browsers to run an app. They're not standalone apps in the sense of having to download and install code into your device. Apps are built using HTML and designed especially for the small screens, and no need to download from the App stores.

Technology used: Web apps are designed using HTML5, CSS, JavaScript, Ruby, and similar programming languages used for web work.

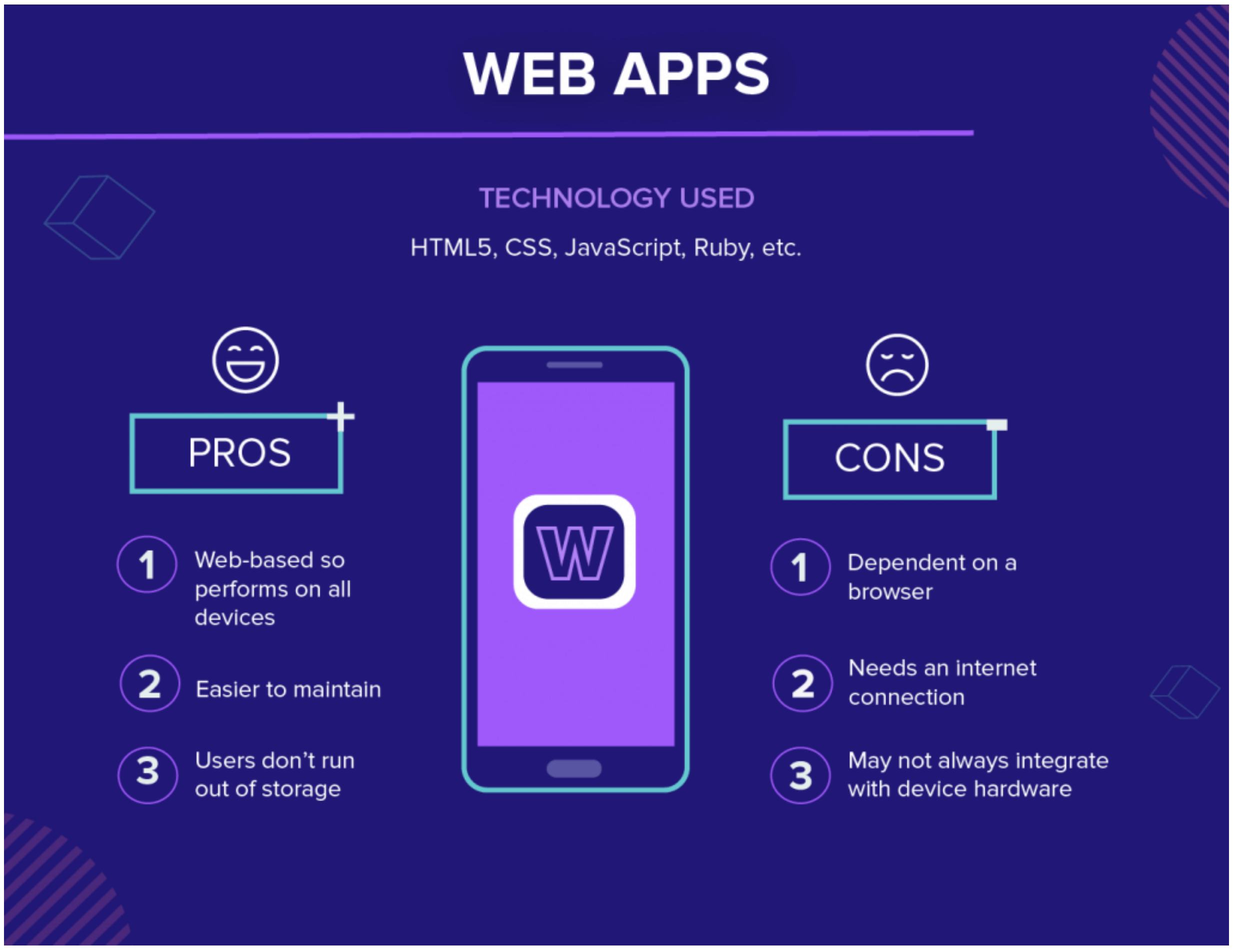
Examples of web apps; google docs, google analytics, Canva, Starbucks etc

Advantages

- **i.** The apps are responsive as when opened on mobile, it will show in mobile view.
- **ii.** It can work on any device with a browser with an internet connection.
- **iii.** There is no need to download hence they don't take up space on your device like native apps
- **iv.** Easier to maintain
- **v.** Users don't need to download the updates at the app store

Disadvantages

i. Poor performance when an internet connection is not available.
ii. Web apps are entirely dependent on the browser used on devices
iii. There will be functionalities available within one browser and not available on another, possibly giving users varying experiences.
iv. And because they're shells for websites, they won't completely work offline. Even if they have an offline mode, the device will still need an internet connection to back up the data on your device, offer up any new data, or refresh what's on screen.


WEB APPS
TECHNOLOGY USED
HTML5, CSS, JavaScript, Ruby, etc.
PROS
1 Web-based so performs on all devices
2 Easier to maintain
3 Users don't run out of storage
CONS
1 Dependent on a browser
2 Needs an internet connection
3 May not always integrate with device hardware

Reasons to build a Mobile App

Consider this statistic: there are 15 billion mobile devices worldwide in 2021. There are millions of apps on the Apple store and Google Play. If one of these is not yours, then maybe it's time to consider getting some "App Appeal"! Here are our top 10 reasons to build a mobile app for your business.

1. **Generate additional sales**. Mobile apps generate additional revenues for a company in three key ways: firstly, because most people carry a mobile device, apps encourage repeat orders from customers on the go. The sheer simplicity and convenience of apps versus going to a website makes it easier for consumers to buy. Secondly a company can benefit from a new advertising revenue stream through apps. And thirdly, the app itself may be so compelling that you can charge for it!
2. **Reduce cost to serve**. Mobile apps allow customers to access basic information and purchase without needing to call a call centre. On some apps, you can even click through a "visual IVR" menu rather than waste time and money listening to prompts in a call queue. Automation obviously reduces servicing costs.
3. **Reduce marketing cost**. A hidden benefit of apps is in the low cost of marketing – compared to traditional advertising and direct marketing, pushing notifications to customers who have downloaded your app is cheap. For example, you may want to announce an event or new product offer.
4. **Enhance the customer experience**. Many websites just don't look great on tiny phone screens, but mobile apps are purpose-designed for small screen sizes so are easier on the eye and simpler to use for the customer. Apps also offer instant access to your contact channels – with one touch, a customer can click to call or initiate a web chat. For users who have their location turned on, the app can also recognise where you are in real time and provide location specific information and directions. The Uber app, which is disrupting the taxi industry, even allows you to visualise your car approaching.

5. **Get the competitive edge.** While most industries have cottoned on to apps, perhaps your business is in a niche where your competitors don't offer an app yet, and if yes, this can be turned to your advantage as a differentiator. You can also custom-build unique features into your app that others don't have – the possibilities are endless.
6. **Broaden your market coverage.** Without an app, you may be missing out on a segment of the market that simply prefers to do business that way. A quick way to understand whether you have a mobile-addicted market is to look at the percentage of visitors who access your website through mobile devices. You can also use common sense, which suggests, for example, that young consumers are more likely to want to transact through an app than mature B2B buyers.
7. **Create stickiness.** What better way to create stickiness with your customers than embedding your brand in your customer's pocket! Unlike mobile websites, apps are always visible on the user's phone home screen. Customers are more likely to repeatedly interact with businesses that are at their fingertips, so apps are a great customer loyalty tool.
8. **Deliver speed.** Mobile apps open faster than a mobile website loads, in fact many aspects of apps are usable even without an Internet connection. In today's digital world, it's a race to deliver services and products to the customer faster, and the winner of the race will win more customers.
9. **Get feedback.** You can easily get actionable insights from your customers, by conducting an "in-app" survey, or by analysing customer reviews in the App Store.
10. **Stay in control** of the customer relationship. Apps are a platform which allow you to have a direct relationship with your customer, without any middleman. You are more likely to retain customers once you know who they are, where they are, and how to reach them.

Mobile apps may not be the right move for every business, but certainly for most, they offer huge appeal and an ROI which can be easily calculated.

Android

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes

History of Android

The history and versions of android are interesting to know. The code names of android ranges from A to T currently. Let's understand the android history in a sequence.

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White** and **Nick Sears**.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

7) In 2008, HTC launched the first android mobile.

Android Versions, Codename and API

Let's see the android versions, codenames and API Level provided by Google.

VERSION	CODE NAME	API LEVEL
1.5	Cupcake	3
1.6	Donut	4
2.1	Éclair	7
2.2	Froyo	8
2.3	Gingerbread	9 and 10
3.1 AND 3.3	Honeycomb	12 and 13
4.0	Ice Cream Sandwitch	15
4.1, 4.2 AND 4.3	Jelly Bean	16, 17 and 18
4.4	KitKat	19
5.0	Lollipop	21
6.0	Marshmallow	23
7.0	Nougat	24-25
8.0	Oreo	26-27

9.0	Pie	28
10.0	Quince Tart	29
11.0	Red Velvet cake	30
12.0	Snow cone	31 and 32
13.0	Tiramisu	33

Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services.

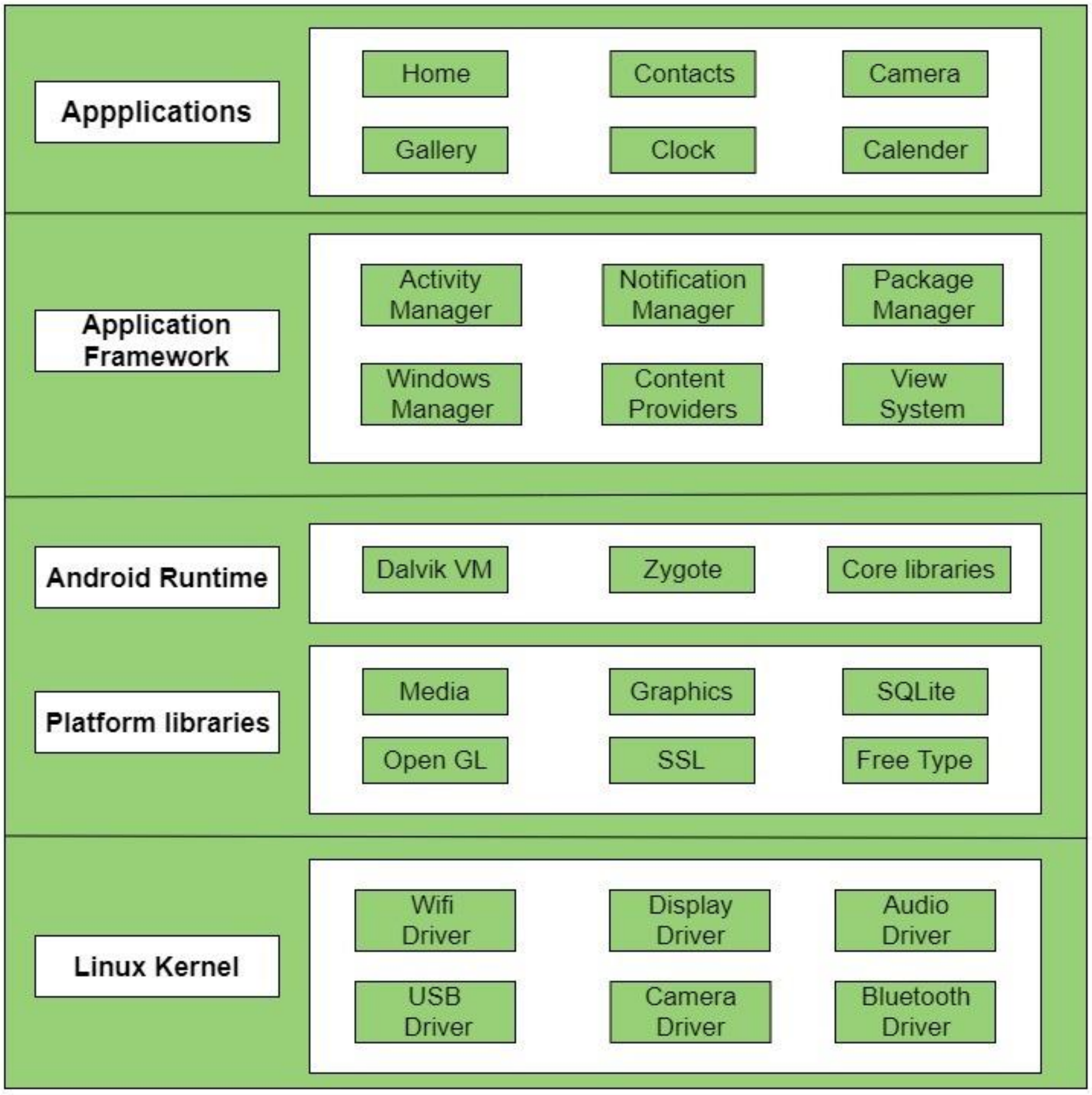
Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

The main components of android architecture are following:-

- Applications
- Application Framework
- Android Runtime
- Platform Libraries
- Linux Kernel

Pictorial representation of android architecture with several main components and their sub components

–



Applications –

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.
It runs within the Android run time with the help of the classes and services provided by the application framework.

Application framework –

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

Application runtime –

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

Platform libraries –

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

- **Media** library provides support to play and record an audio and video formats.
- **Surface manager** responsible for managing access to the display subsystem.
- **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- **SQLite** provides database support and **FreeType** provides font support.
- **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.

Linux Kernel –
Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

The features of Linux kernel are:

- **Security:** The Linux kernel handles the security between the application and the system.
- **Memory Management:** It efficiently handles the memory management thereby providing the freedom to develop our apps.
- **Process Management:** It manages the process well, allocates resources to processes whenever they need them.
- **Network Stack:** It effectively handles the network communication.
- **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

COMPONENTS OF ANDROID APPLICATION

There are some necessary building blocks that an Android application consists of. These loosely coupled components are bound by the application manifest file which contains the description of each component and how they interact. The manifest file also contains the app's metadata, its hardware configuration, and platform requirements, external libraries, and required permissions. There are the following main components of an android app:

1. Activities

An activity is a class that represents a single screen. Activities are said to be the presentation layer of our applications. The UI of our application is built around one or more extensions of the

Activity class. By using Fragments and Views, activities set the layout and display the output and also respond to the user’s actions. An activity is implemented as a subclass of class Activity.

Different States of App (or, the main App Activity)

Starting from a user clicking on the App icon to launch the app, to the user exiting from the App, there are certain defined states that the App is in, let's see what they are.

1. When a user clicks on the App icon, the Main Activity gets started and it creates the App's User Interface using the layout XMLs. And the App or Activity starts running and it is said to be in **ACTIVE** state.

2. When any dialog box appears on the screen, like when you press exit on some apps, it shows a box confirming whether you want to exit or not. At that point of time, we are not able to interact with the App's UI until we deal with that dialog box/popup. In such a situation, the Activity is said to be in **PAUSED** state.

3. When we press the Home button while using the app, our app doesn't closes. It just get minimized. This state of the App is said to be **STOPPED** state.

4. When we finally destroy the App i.e when we completely close it, then it is said to be in **DESTROYED** state.

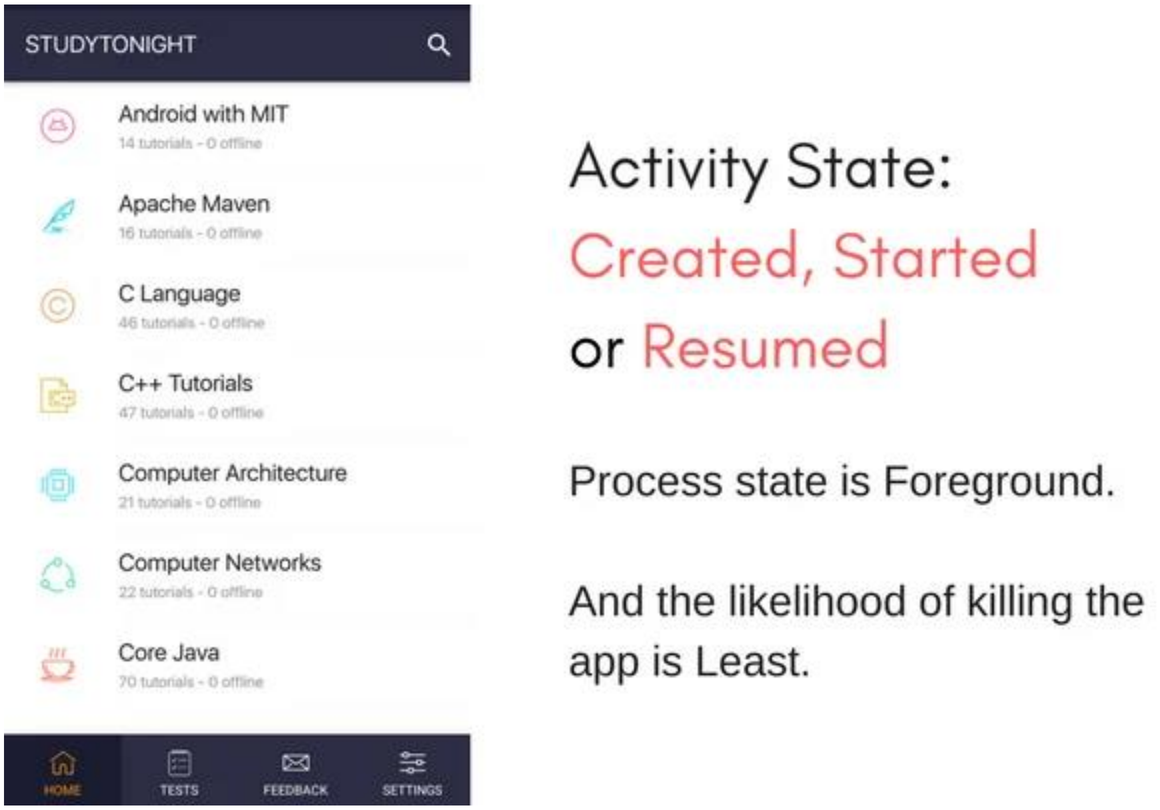
Hence, all in all there are four states of an Activity(App) in Android namely**, Active, Paused, Stopped and Destroyed**.

From the user's perspective, The activity is either visible, partially visible or invisible at a given point of time. Lets discuss these states in detail.

a) Active State

- When an Activity is in active state, it means it is active and running.
- It is visible to the user and the user is able to interact with it.

- Android Runtime treats the Activity in this state with the highest priority and never tries to kill it.



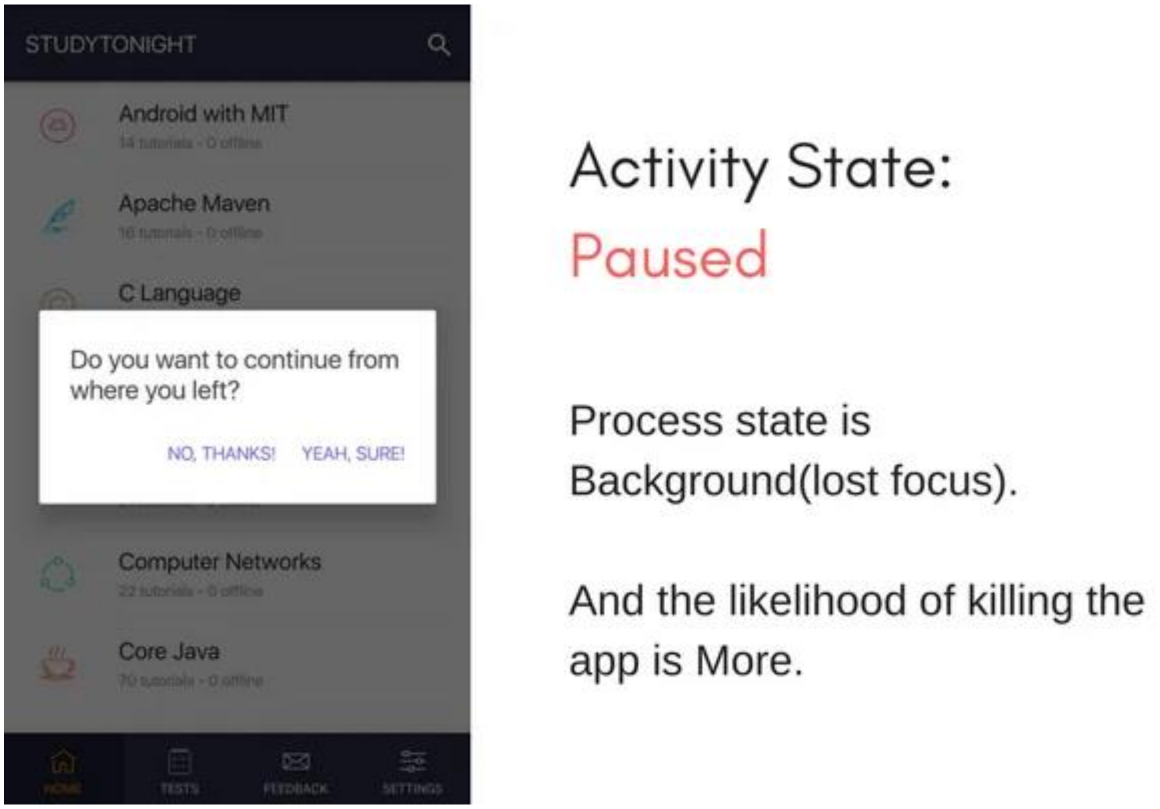
Activity State: Created, Started or Resumed

Process state is Foreground.

And the likelihood of killing the app is Least.

b) Paused State

- An activity being in this state means that the user can still see the Activity in the background such as behind a transparent window or a dialog box i.e it is partially visible.
- The user cannot interact with the Activity until he/she is done with the current view.
- Android Runtime usually does not kill an Activity in this state but may do so in an extreme case of resource crunch.



Do you want to continue from where you left?

NO, THANKS! YEAH, SURE!

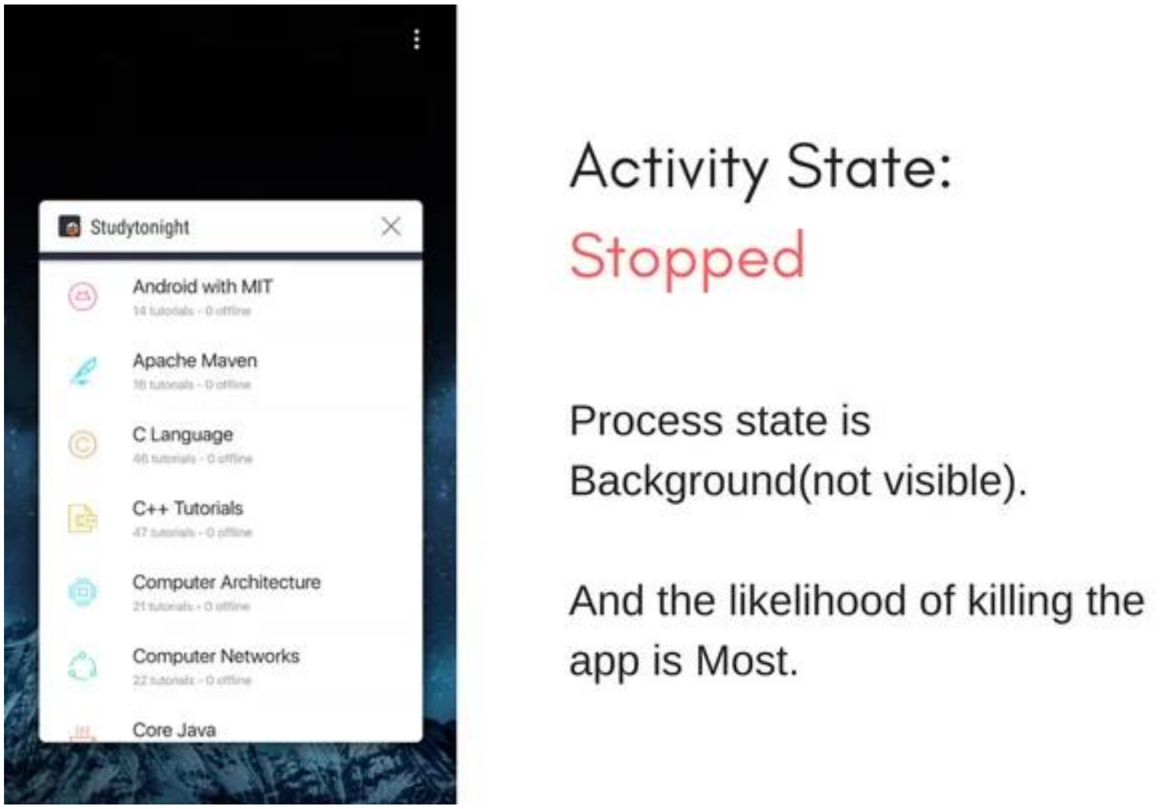
Activity State:
Paused

Process state is Background(lost focus).

And the likelihood of killing the app is More.

c) Stopped State

- When a new Activity is started on top of the current one or when a user hits the Home key, the activity is brought to Stopped state.
- The activity in this state is invisible, but it is not destroyed.
- Android Runtime may kill such an Activity in case of resource crunch.



Activity State:
Stopped

Process state is Background(not visible).

And the likelihood of killing the app is Most.

d) Destroyed State

- When a user hits a Back key or Android Runtime decides to reclaim the memory allocated to an Activity i.e in the paused or stopped state, It goes into the Destroyed state.
- The Activity is out of the memory and it is invisible to the user.

Note: An Activity does not have the control over managing its own state. It just goes through state transitions either due to user interaction or due to system-generated events.

The activity methods will be discussed later in this document.

2. Services

Content Providers are used to share data between the applications. Services are like invisible workers of our app. These components run at the backend, updating your data sources and Activities, triggering Notification, and also broadcast Intents. They also perform some tasks when applications are not active. A service can be used as a subclass of class Service:

3. Content Providers

It is used to manage and persist the application data also typically interacts with the SQL database. They are also responsible for sharing the data beyond the application boundaries. The Content Providers of a particular application can be configured to allow access from other applications, and the Content Providers exposed by other applications can also be configured. A content provider should be a sub-class of the class ContentProvider.

4. Broadcast Receivers

They are known to be intent listeners as they enable your application to listen to the Intents that satisfy the matching criteria specified by us. Broadcast Receivers make our application react to any received Intent thereby making them perfect for creating event-driven applications.

5. Intents

It is a powerful inter-application message-passing framework. Intent are used to invoke components. They are extensively used throughout Android. Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.

6. Widgets

These are the small visual application components that you can find on the home screen of the devices. Widgets are an essential aspect of home screen customization. You can imagine them as "at-a-glance" views of an app's most important data and functionality that is accessible right from the user's home screen. Users can move widgets across their home screen panels, and, if supported, resize them to tailor the amount of information within the widget to their preference.

This page provides an introduction to the different types of widgets you might want to create and some design principles to follow. To start building an app widget, read Create a simple widget.

Widget types

As you begin planning your widget, think about what kind of widget you're trying to build. Widgets typically fall into one of the following categories:

i. **Information widgets**

Information widgets typically display a few crucial information elements and track how that information changes over time. Good examples for information widgets are weather widgets, clock widgets or sports score tracking widgets. Touching information widgets typically launches the associated app and opens a detailed view of the widget information.

ii. **Collection widgets**

Collection widgets specialize in displaying multiple elements of the same type, such as a collection of pictures from a gallery app, a collection of articles from a news app or a collection of emails/messages from a communication app. Collection widgets can scroll vertically.

Collection widgets typically focus on the following use cases:

- Browsing the collection
- Opening an element of the collection to its detail view in the associated app for consumption
- Interacting with elements such as marking them done (with new support for compound buttons in Android 12 (API level 31))

iii. **Control widgets**

The main purpose of a control widget is to display often-used functions, so that the user can trigger right from the home screen without having to open the app first. You can think of them as remote controls for an app. An example of a control widget is a home control widget that lets users turn lights on or off in different rooms of a house.

Interacting with a control widget may or may not open an associated detail view in the app. This depends on whether the control widget's function outputs any data, such as in the case of a search widget

iv. Hybrid widgets

While some widgets tend to gravitate towards one of the types in the preceding sections (information, collection, or control), many widgets in reality are hybrids that combine elements of different types.

A music player widget is primarily a control widget, but also keeps the user informed about what track is currently playing. It essentially combines a control widget with elements of an information widget type. When planning your widget, design around one of the base types and add elements of other types if needed.

7. Notifications

Notifications are the application alerts that are used to draw the user's attention to some particular app event without stealing focus or interrupting the current activity of the user. They are generally used to grab user's attention when the application is not visible or active, particularly from within a Service or Broadcast Receiver. Examples: E-mail popups, Messenger popups, etc

8. Fragment

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

9. AndroidManifest.xml

It contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

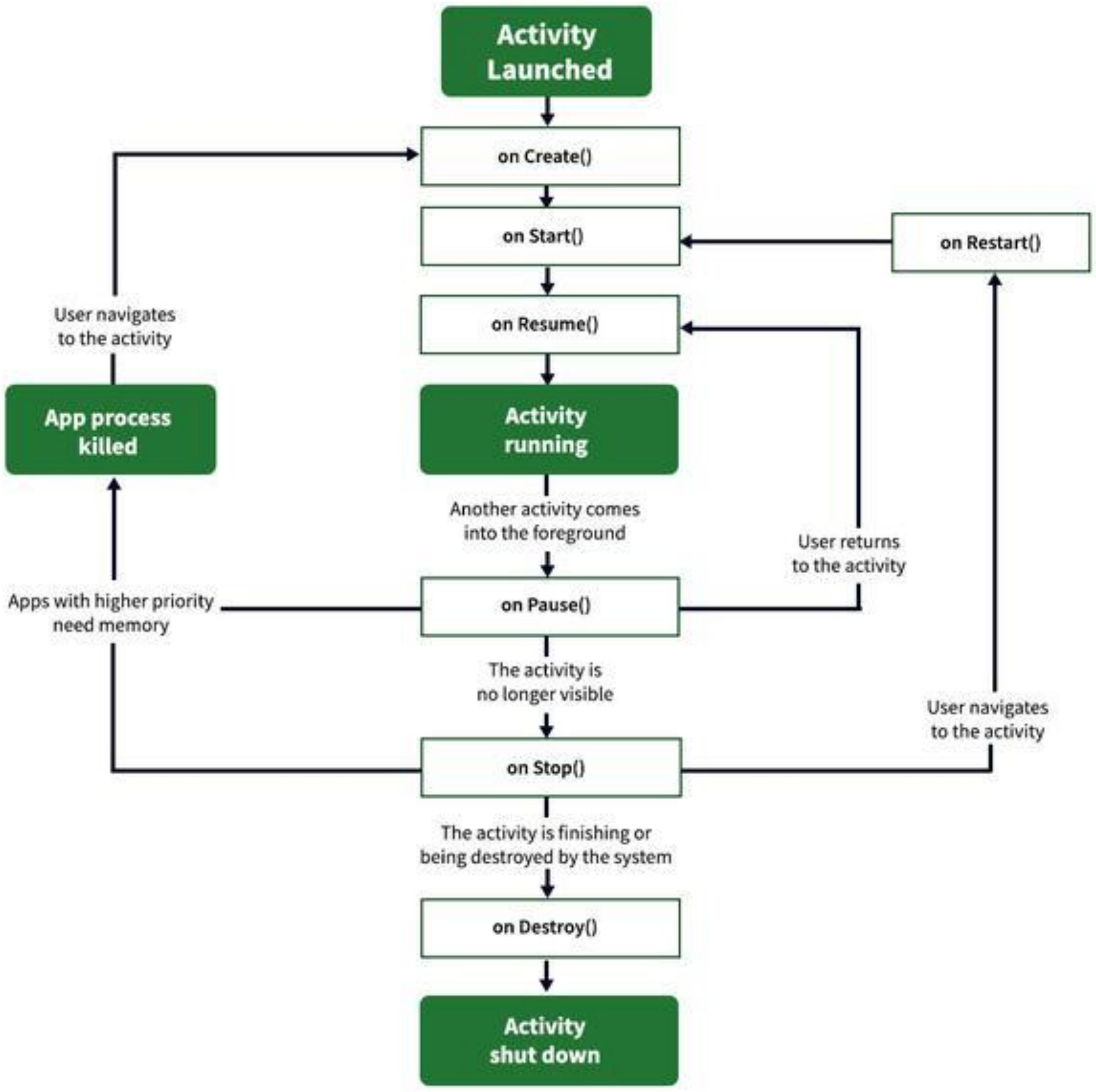
10. Android Virtual Device (AVD)

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

Android Activity Lifecycle methods

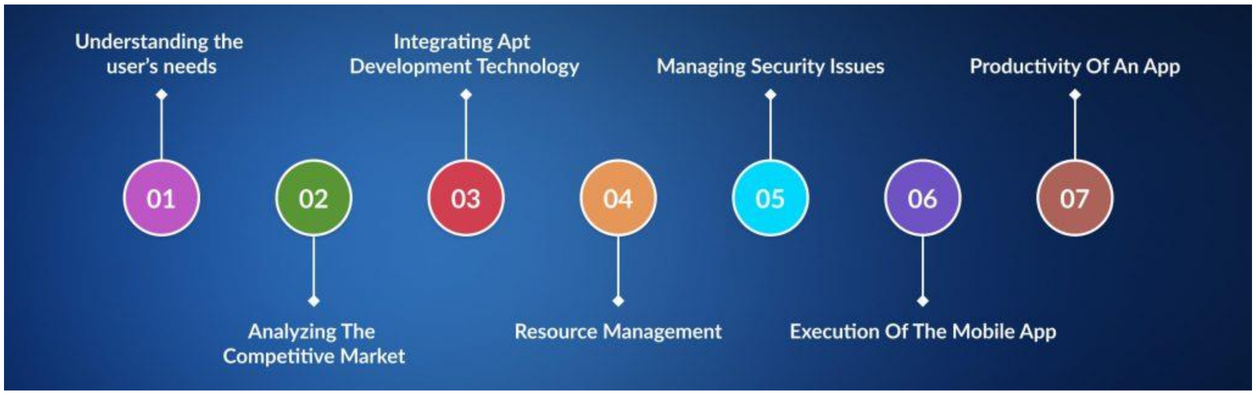
Let's see the 7 lifecycle methods of android activity.

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.


Activity Launched
on Create()
on Start()
on Restart()
on Resume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
on Pause()
The activity is no longer visible
User navigates to the activity
on Stop()
The activity is finishing or being destroyed by the system
on Destroy()
Activity shut down

Activity Lifecycle in Android

COMMON MOBILE APP DEVELOPMENT CHALLENGES


Understanding the user's needs
Integrating Apt Development Technology
Managing Security Issues
Productivity Of An App
01
02
03
04
05
06
07
Analyzing The Competitive Market
Resource Management
Execution Of The Mobile App

Before discussing the key challenges, let's have a brief understanding of what mobile application development is. Mobile application is a sequence of processes, including writing software for multiple wireless computing devices like smartphones and tablets. Nonetheless, it is an intricate procedure in its entirety, and few obstacles can come in the way. Let's have a look:

#Challenge 1 – Understanding the user's needs

Evaluating the target audience's needs is one of the greatest challenges for the developers. Perceiving the vast competition in the mobile app development sector, it becomes daunting to attract eyeballs. The app should be eye-catching as soon as the user lands on it. Being compatible with users' requirements directly influences the quality of a mobile app.

Assess the following factors while developing a mobile application:

1. **Knowing the purpose of the application**
2. **Addressing the functionalities during an app development**
3. **Coming up with the most unique and bespoke idea**
4. **Making a user-friendly app**
5. **Creating an app that adheres to the market needs**

#Challenge 2 – Analyzing the Competitive Market

Mobile app development is not only succumbed to 'how mobile applications are developed; it is way more than that. In this aggressive, competitive market, numerous mobile apps are in an unending hustle to stand out amongst the rest. Reaching out to the users is not at all plain sailing. The urge to create an enchanting customer experience can sometimes distract the developers.

Applying their vision while being concurrent with the current target market situation, in-depth research and diverse business strategies can help them in eliminating this issue.

#Challenge 3 – Integrating Apt Development Technology

If you are beginning mobile application development in the cloud or any other framework, working with a robust, constructive and scalable technology is of utmost importance. Understand your needs and what your target audience wants to make a suitable selection of technology, like hybrid apps, native apps, and cross-platform apps. There are several apps in the technological market, and the coders should be well-versed in selecting the most suitable one. Choosing an obsolete technology can result in a hindrance to building a flexible and all-inclusive mobile app. That's why it is advisable to hire an app development agency, like App Incubator, that follows the latest trends in the industry. Our developers are highly proficient with the latest technology, and they precisely know how to tackle **challenges in website and app development**.

#Challenge 4 – Resource Management

Having a reliable funding source equals having an innovative idea to implement. No matter how pioneering your mobile app idea is, scarcity of resources can pull you back to square one. Several IT companies are facing the challenges of mobile and **web application development**. All you need to do is find a proper channel for your app idea's investment. Consider options like taking a loan or venturing into a partnership to raise funds easily. Cohering to your app idea along with managing funding will be a win-win situation for your business.

#Challenge 5 – Managing security issues

Managing security issues is daunting for developers. No customer would want to engage with apps that are not secure and safe to surf. With a plethora of operating systems and device platforms available out there, safeguarding an app from malware problems can be an arduous task. Issues like hardware or software disintegration will exhaust your resources. It's better not to reach that situation. Incorporate authentic frameworks that prevent virus issues and enforce strong authentication with data encryption to keep cybersecurity problems at bay.

#Challenge 6 – Execution of the Mobile App

Smooth execution of an app is a common challenge of mobile application testing. Ideation, design, and application performance are the three vital aspects to consider while creating an app. However, creativity and design will go in vain without precise execution. A developer should always focus on building a bug-free app that runs on minimum battery. Though, this factor depends on the user's device type too. Some apps can only be working on the latest devices, while some apps are every version friendly. These are the common mobile app testing challenges faced by developers. Variable factors like images, caches, visual effects, and sensors are also responsible for deciding the performance of an application.

#Challenge 7 – Productivity of an App

The common issues observed with users are witnessing quick battery discharge and heating up of phones. This happens due to a lack of thorough checking of an app's productivity, leading to a disappointing end-user experience. It is necessary to overcome this mobile app challenge while testing. Therefore, the developing team is advised to run a testing trial on the app to verify the battery's performance and the draining time. Visioning a bug-free app that delivers the finest performance without consuming excessive power should be one of the primary motives of the developers.

Assignment

State the major difference between *Client based app* and *Web based app* as used in the context of mobile application development.