

Q.1) Based on your understanding, identify a recent business trend that has influenced the android platform. Explain how this trend impacts android app developers and business in the mobile app industry.

→ As per my understanding, one notable trend in the mobile application industry that was influencing the android platform was the rise of progressive web apps. PWAs are web applications that offer app-like experiences directly through web browsers.

1. Cross-platform compatibility : PWAs are designed to work seamlessly across various platforms and devices, including android. Developers had to consider creating PWAs alongside traditional Android Apps to ensure broad accessibility.

2. Enhanced user Experience :- PWAs, aimed to provide a smoother and more engaging user experience which set higher expectations for android app developers. This encouraged them to focus on improving the quality and performance of their apps to complete effectively.

3. **Progressive Enhancement**:- Developers needed to adopt progressive enhancement strategies to ensure that android apps remained competitive by offering progressive experiences, similar to PWAs.
4. **Impact on business in the mobile app Industry.**

  1. **cost savings** :- Business could potentially save on development costs by investing in a single PWA that works across multiple platforms, including, android, rather than building separate native apps.
  2. **Increased Reach** :- PWAs enabled business to reach a wider audience, including android devices, without relying solely on app store distribution. This broader reach could lead to increased user acquisition.
  3. **Improved Engagement**:- The focus on delivering app-like experiences through PWAs encouraged engagement and retention, ultimately benefiting their mobile strategy.

4. competition and Innovation : The rise of PWAs introduced competition, driving business to innovate their android apps to keep up with evolving user expectations and technology trends.

Q.2) What is the purpose of Inflater layout in android development, how does it fit into architecture of Android Layout?

→ In android development, an 'Inflater' refers to layoutInflater, which plays a crucial role in creating a user interface from XML layout files. Its primary purpose is to take an XML layout resource and convert it into a corresponding view object in memory. Here's the layoutInflater fits into the architecture of android layouts:-

1. XML Layout files:- In ~~Android~~ UI components are often defined using XML layout files. These files describe the structure and appearance of the UI, specifying things like widgets, their properties and their placement within the UI.

2. Layout Inflater When your android app runs, it needs to turn these XML layout files into actual view objects that

that can be displayed on the screen. This process is known as 'Layout Inflater'.

3. Layout Inflater:- It is responsible for reading the XML layout files and instantiating the corresponding view objects in memory. It takes the XML file as input, parses it, and creates the view objects such as text view, buttons etc. specified in the XML.

4. Dynamic UI creation:- Layout inflation is valuable when you need to create UI elements dynamically for example in response to user interactions or when working with Recycler view to display lists of items.

5. Binding Data:- Once the view objects are created, they can be further customized and data can be bound to them. This is typically done using data binding techniques or programmatically setting properties and values.

6. Displaying UI:- After inflation and customization, the view objects can be added to the app's layout hierarchy and displayed on the screen.

Q.3

Explain the concept of a CustomDialogBox in android applications. Provide examples to illustrate it?

→ In android applications, a custom dialog box is a pop-up window that overlays the current activity and is often used to interact with the current activity and as often used to interact with the user, gather input or display information overview of a custom Dialog box.

Purpose :- custom - dialogs, are used when you want to present information, receive user input or perform actions within a self-contained, isolated UI element that temporarily interrupts

components:- A custom dialog box typically consists of various UI elements like bottom, TextViews, images or inputs fields, tailored, to the specific requirements. This customization allows for creativity in design and functionality.

example:-

```
fun showCustomDialog () {
    val customDialog = Dialog (this)
    customDialog .setContentView (R.layout.custom-
    val messageTextView = customDialog .findview<
        R.Id.TextView>
    = customDialog .findview<
        R.Id.TextView>
```

(R.id.messageTextView)

val okbutton = customDialog.findViewById  
<button> @R.id.okbutton).

messageTextView.text = "This is a custom dialog!"

```
okbutton.setOnClickListener {  
    customDialog.dismiss()  
    customDialog.show()  
}
```

use cases of custom dialog box: login confirmation Dialog . settings . Informational Pop-up Media playback controls.

Q4. How do activities, services, and the android Manifest file work together to make an android app? can you describe their main roles and provide a basic example on that.

→ 1. Activities

- Role : Activities represent individual screens or UI components in an android app. They manage the user interface and user interactions.

2. Services

- Role : services are background components that perform long running operations or handle tasks that don't require a user interface. They can run even if the app's UI is not visible.

3. Android Manifest file :-  
 • Role :- The android manifest .xml files is like the app's blueprint. It decides defines the app's components and how they interact with the android system and other components.

Example :- In android Manifest .xml you specify which activities are part of your app, their launch nodes, permissions and service declarations. This file acts as a blueprint for android system to understand your app's structure and behavior.

→ class MainActivity : AppCompatActivity () {  
 override fun oncreate (savedInstanceState: Bundle?) {  
 super.oncreate (savedInstanceState)  
 setContentView (R.layout.activity\_main)  
 startServiceButton.setOnClicklistener {  
 val service Intent = Intent (this,  
 notificationService::class.java)  
 startservice (serviceIntent) } }

→ class notificationService : IntentService ("  
 notificationService") {

```
override fun onHandleIntent (intent : Intent){  
    if (intent != null) {  
        createNotification() } }
```

```
private fun createNotification () {  
    val channelID = "my-channel":  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
        val name = "my channel".  
        val notificationManager = getSystemService  
            ("notification manager"  
            classOf<NotificationManager>).
```

```
        notificationManager.createNotificationChannel  
            (channel) }
```

```
    val builder = NotificationCompat.Builder  
        (this, channelID)  
        .setSmallIcon (R.drawable.ic_launcher_foreground).  
        .setContentText ("this is notification  
            from service").
```

```
}
```

Q.5. How does the android manifest file impact the development of an android app? provide eg to demonstrate its significance?

→ The android manifest file is a crucial component in the development of an android application. It serves several important purposes and its content significantly impacts how the android system interacts with and manages your application. Significance of the android manifest file:

- App configuration
- permissions
- Component declaration
- Intent filters
- App Lifecycle

Example: <manifest xmlns:android = 'https://schemas.android.com/apk/res/android'  
 package = 'com.example.myapp'>  
 <application  
 android:allowBackup = 'true'  
 android:icon = '@imppmap/ic-launcher'.  
 android:label = '@string/app-name'.  
 android:roundIcon = '@imppmap/ic-launcher-round'.  
 android:supportRtl = 'true'  
 android:theme = '@style/AppTheme">>  
 <activity android:name = '.MainActivity'.  
 <intent-filter>

```
<action android:name = 'android.intent.action  
· MAIN' />  
<category android:name = 'android.intent.category  
· launcher' />  
</intent-filter>  
</activity>  
<activity android:name = 'secondActivity'>  
... Declare additional activities here ...  
</activity>  
<user-permission android:name = 'android.  
permission. INTERNET'>  
</application>  
</manifest>
```

Q-6). What is the role of resources in android development? Discuss the various types of resources and their significance in creating well structured applications.

→ Resources play a fundamental role in android development by providing a structured way to manage assets, value ; layouts and other elements used in your app. They help create flexible, maintainable and device independent application . The various types of resources and their significance with examples

## 1. Layout Resources:

- type: XML files in the 'res/layout' directory.
- significance: Define the structure and appearance of the app's user interface.
- Example: 'activity\_main.xml' defines the layout of your main activity, specifying UI components like buttons, text views and their arrangements.

<Button

```
    android:id = '@+id/myButton'
    android:layout_width = 'wrap-content'
    android:layout_height = 'wrap-content'
    android:text = 'click-me!' />
```

## 2. Drawable Resources:

- type: Images and drawable assets in the 'res/drawable' directory.
- significance: store graphics, icons and images used in your app.
- Example: 'ic\_launcher.png' is the app's launcher icon.

## 3. String Resources:

- type: strings defined in XML files under 'res/values'.
- significance: store text strings, making it easier to provide consistency and translations and

Example: 'res/values / strings.xml' contains string resources.

```
<string name = 'app-name'> My APP </string>
<string name = 'welcome-message'> welcome to my APP
</string>.
```

#### 4. color Resources:

-type: colors defined in XML files under 'res/values'.

-significance: store color values, ensuring consistency in the app's design.

Example: 'res/values / colors.xml' defines color resources.

```
<color name = 'primary-color'> #007ACC </color>
<color name = 'accent-color'> #FFA500
</color>.
```

#### 5. style Resources:

-type: styles defined in XML files under 'res/values'.

-significance: Define reusable styles for UI components.

Example: 'res/values / styles.xml' defines style.

```
<style name = 'MyButtonStyle'>
<item name = 'android:background'>
    @drawable/my-button </item>
</style>.
```

```
<item name = 'android:textcolor'> @color/primary-color </item>
</style>
```

## 6. Dimension Resources

-type:- Dimensions defined in XML files under 'res/values'.

-significance: store dimension 'values', ensuring a consistency layout.

Example: 'res/values/dimens.xml'. defines dimension resource.

```
<dimen name = 'margin-large'> 16 dp </dimen>
<dimen name = 'padding-medium'> 8 dp </dimen>
```

## 7. Raw resources

-type:- files stored in the 'res/raw' directory.

-significance: store non-XML files, such as JSON data, audio.

Example: store a JSON file for app configuration.

Q.7. How does an Android service contribute to the functionality of a mobile application? Describe the process of developing an android device.

Contributions of Android services:-

1. Background processing: services allows apps to perform tasks in the background without blocking the user interface.

2. Long-running operations :- services are ideal for handling operations that require more time to complete, such as playing ~~big~~ music.

3. Inter-component communications :- services enable components like activities, broadcast receivers and other services to communicate with each other efficiently.

4. Foreground services: Android services can run in the foreground, even when the app isn't in the foreground. This is useful for features that require ongoing user interaction, like music playback.

Process of developing an android service:

- Define the service class: create a new Java or Kotlin class that extends the 'service class'. override methods like oncreate(), onDestroy(), onStartCommand() to define the behavior of your service.

2. Configure service in Manifest:- Declare your service in the android manifest.xml file to inform the android system about its existence and configuration.  
`<service android:name = "myservice"/>`
3. Start or Bind the service : Decide whether you want to start your service or bind it to other components . use `startService()` or `bindService()`.
4. Implement Service Logic: In service class, implements the specific logic your service needs to perform its task.
5. Handle Lifecycle:- Release resources when they're no longer needed and consider using '`stopself()`' or '`stopService()`'.
6. Interact with other components:- use appropriate mechanisms like intents, broadcast or callbacks to facilitate communication.
7. Foreground services (optional) :- If your service needs to run in the foreground, 'startForeground'.

8. Testing: Thoroughly test your service to ensure if functions as expected, including handling various scenarios, like network failures.
9. optimization : optimize your service for performance and resource efficiency to minimize battery usage.
10. Error handling and Logging : Implement proper error handling and logging mechanisms to diagnose and address issues that may occur while service is running.

