# Assignment :- 7

Q Based on your understanding of recent business trend that has influenced the Android platform Explain how this trend impacts android app developers and businesse in the mobile app industry

-> As per my knowledge one notable trend in the mobile app industry that has since the rise of progre the android platform since the rise of Pwa phAs are web applications that often app-like experien ces directly through web browsers

- Impact on android app developers.

1) Cross-platform compatibility : PWAs are desi- gned 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 provides a smoother and more engaging user experience which set higher expectation for android app developers This encouraged them to focus on improving the quality and performance of their app to complete effectivity

3) progressive Enhancement Developers needed to adopt progressive enhancement strategies to ensure that android apps remained competitive by offering progressive and responsive user experiences similar to PWAs.

Impact on business in the mobile app industry

1) cost saving: Business could potentially save on development cost by investing in a single PWA instead of multiple platforms, including Android, rather than building separate native apps

2) Increased reach: PWAs enabled business to reach wider audience, including users with android rather than building separate native apps.

3) Improved engagement: The focus on delivering app like experiences through PWAs encouraged business to prioritize user engagement and retention ultimately benefiting their mobile strategy.

4) Competition and Innovation the rise of PWAs introduced competition driving businesses to innovate their apps to keep up with evolving user expectations. and tech trends.

2) what is the purpose of an Inflater of layout in Android development and how does it fit into The architecture of Android layout

-> In Android development an 'Inflater' is a crucial component that is used to instantiate or inflate the layout xml files into their corresponding view objects

- it's how it fits into the arche techure of Android layouts.

1 Layout XML files: In Android UI elements are defined in xml files within the 'res' 'layout' directory of an Android project. These XML files contain the structure of the user interface specifying things like buttons, fields, images, etc as well their attributes.

→ 2) Inflating layouts: when an Android app runs the layout XML files need to be converted into view objects that can be displayed on the screen. This is where the 'Inflater' comes in. It's used to read the XML files and create the corresponding view objects.

3) Inflating ~~layouts~~: ~~when~~ with context: The 'Inflater' requires a context object provides essential information about the current state of the application about text is typically provided by an Activity fragment.

4) Attaching to parent: The 'Inflater' also has the option to attach the inflated layout to a parent view if specified. This allows the newly created view to be automatically added to the specified container within the parent view.

5) Returning a view: Once the inflation process is complete. The 'INFLATER' returns the root view of the inflated ~~layout~~ which can then be used within the views. such as setting text applying styles, then be used within the applica

6) Manipulating Views Programmatically: After inflation developers can programmatically interact with the views. such as setting text applying styles, adding event listeners and more.

7) Displying in the UI: Finally the manipulated views can be added to the user interface using methods like 'setContentView' for activities

Page No.

or by adding them to the view hierarchy Through their parent viewGroup.

8) Explain the concept of a CustomDialog Box in Android application. provied examples to illustrate its use.

→ In Android application, a customDialogBox is a pop-up window that overlays the current activity and is used to interact with the user & gather input or display information overview of a custom DialogBox or d information overview of a customDialog 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.

- simple example of creating and using a custom dialog in Android

```
Fun showCustomDialog() {
    val customDialog = Dialog(this)
    CustomDialog.setContentView(R.layout.customdialog_layout)
    val messageTextView.text = "This is a customD
    okButton.setOnClickListener {
        CustomDialog.dismiss()
    }
    CustomDialog.show()
}
```

use cases of customDialogBox: login confirmation dialog setting Information pop-up medick playback controls

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 of how they cooperate to design a mobile app ?

→ 1) Activities :

- Role : Activites represent individual screens or UI components in an Android app. They manage the user interface and user interaction.

2) services.

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

3) Android manifest file

- Role : The Android manifest file is like the app's blueprint. It declares the app's components and defines how they interact with the android system and other components.

```
→ class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState
        : Bundle)
    {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    startServiceButton.setOnClickListener {
    val serviceIntent : Intent (this, MyBackgroundService
        ::class.java)
```

```
override fun onCreate(savedInstanceState: Bundle
{
  super.onCreate(savedInstanceState)
  setContentView(R.layout.activity_main)
  startService Button.setOnClickListener {
    val serviceIntent = Intent(this NotificationService::class.java)
    startService(serviceIntent)
  }
}
}
```

→ class NotificationService: IntentService("notification
override fun onHandleIntent(intent: Intents) {
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.createNotificationChannel channel
}
```

val Builder = Notification.CompatBuilder(this, channel
setSmallIcon(R.drawable.ic_launcher_foreground)
setContentText("this is notification from service.
from service")
}
}

5) How does the Android manifest file impact the development of an Android application provide an example 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 impact how the android system interacts with and manages your App. significance of the Android manifest file.

- App configuration
- Component declaration
- permisson
- Intent filter
- App Lifecycle.

→ ex. <?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com"
<uses-permission android:name="android.tool">
<application
android:allowBackup="true"
android:fullBackupContent="@xml/backup_rules"
android:icon="@mipmap/ic_launcher"
android:label="Assignment"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/Theme.Assignment"
tools:targetApi="31">
<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>
</manifest>

6) what is the role of resources in Android development? Discuss the various types of resources and their significance in creating well-structured applications. Provide examples to clarify your points.

→ Resources play a fundamental role in Android development by providing a structured way to manage assets like 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 MainActivity specifying of components like buttons text views and their arrangement

<Button
android:id="@+id/myBtn"

Q How does an Android service contribute to the functionality of a mobile application & describe the process of developing an Android service.

A Contributions of Android services:-

1) Background processing: services allow apps to perform tasks in the background without slowing the user interface.

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

3) Inter-component communication: 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 even when the app is not in the foreground this is useful for features that require ongoing user interaction like music playback, process

Process of developing an Android service:-

1) Define the service class: Create a new java or Kotlin class that extends the Service class that extend the Service class. override methods like onCreate() onStartCommand() to define the behavior of your service

2) Configure service in manifest: Declare your service in 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()

☆ implement 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() me

6) Interact with other components use appropriate mechanism like Intents, broadcast or callbacks to facilitate communication.

7) Foreground service (optional) : If your service needs to run in the foreground, start foreground()'

8) Testing : Thoroughly test your service to ensure it functions as expected including startforground(),

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

android:layout_width="wrap_content"
android:layout_hight="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) Mipmap Resources:
- type: store at multiple resolutions or sizes in the 'res/layout' directory.
- significance: used to provide multiple versions of an image at different resolutions. It ensures that images look crisp and clear on screens with varying pixel densities.
- example ic_launcher -> mipmap-hdpi, mipmap-xxhdpi

4) String Resources.
- Type: strings defined in xml files under 'res/valu
- significance: store text strings making it easier to provide translations and maintain consistency
<string name="app_name">My App</string>
<string name="welcom_message">welcome to my app</string>

5) color resources:
- type: colors defined in xml files under 'res/values'
- significance: store color values ensuring consistency with app's design.
- Example: 'res/values/colors.xml defining color resources
<color name="primary color">#007ACC</color>
6) Style Resources:
- type: styles defined in xml files under 'res/values'
- significance: store color values, ensuring consistency
- example: 'res/values/styles.xml' define style
<style name "My Buttonstyle" >
<item name="android: background">@drawable/my_b
</item>
</style>
7) Dimension Resources.
- type: Dimension define in xml files under 'res/values'
- Example: 'res/values/dimens.xml files, such as JSON data audio, vide.
- example: store a JSON file for app configuration.
8) Raw Resources:-
- type: files stored in the res/raw directory
- significance: store non-XML files such as JSON data, audio, video
- Example: store a JSON file for app configuration

6/10/23