Name :- Jegadal Jaimaa SUB :- MAD Batch- 5B1
Enroll No :- 210120116033 Class - CEIT-B

Ganpat University | U.V. Patel College of Engineering

U.V. Patel College of Engineering

GANPAT UNIVERSITY, KHERVA-384012, DIST - MEHSANA. (N.G.)

Assignment - 1

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 businesses in the mobile app Industry.

→ As per my knowledge, one notable trend in the mobile App industry that was influencing the Android platform was the rise of progressive web Apps (PWAs). PWAs are web applications that offer app-like experiences directly through web browsers.

Impact on Android App Developers:

1. Cross-Platform Compactibility: PWAs are designed to work seamlessly across various platforms and devices, including Android. developers had to considers 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 expectation for Android app developers. This encouraged them to focus on improving the quality and performance of their apps to Compete effectively.

3. progressive Enhancement: Developers need
adopt progressive enhancement strategies to e
that Android apps remained competitive b
offering progressive and responsive user e
similar to PWAs.

- Impact on Businesses in the mobile App
1. Cost Savings: Bussineses ~~Could Potentially~~
on development costs by investing in a single
that works across multiple platforms, including
rather than ~~than works~~ building Separate. native apps

2. Increased Reach: PWAs enabled buspnesses to
reach a wider audience, including users with An
~~rather than~~. devices. without relying solely o
app store distribution. This broader reach co
lead to Increased uses acquisition.

3. Improved Engagement: The focus on deliveri
app-like experiences through PWAs encourag
businesses to prioritize uses engagement an
retention, ultimately benefiting their ~~of PWAs~~
strategy.

4. Competition and Innovation: The rise of PWAs
introduced competition, driving businesses to in
their Android apps to keep up with evolvin
uses expectations and technology trends.

Q-2 What is the puspose of an Inflator of layout
Android development and how does it fit in
architecture of Android layout?

In Android development, an "Inflater" refers to the Layout Inflater, which plays a crucial role in creating a user interface (UI) 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 how the layout inflater 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 Inflation: When your Android app runs, it needs to turn these XML layout files into actual view objects that can be displayed on the screen. This process is known as "layout inflation".

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 TextViews, Buttons, etc. as specified in the XML.

4. Dynamic UI Creation: layout inflation ps
Valuable when you need to create UI eleme
for example, in response to uses inte
or when working with Recycler Views
display lists of items.

5. Binding Data: Once the view objects are
they can be fusthes customized and data
bound to them. this is typically done
data binding techniques or programmatic
setting propestics and values.

6. Displaying UI: After inflation and cust
the view objets can be added to the app
hierarchy and displayed on the screen.

Q-3 Explain the Concept of a Custom dial
in Android applications. provide examples to
its used ~~to~~ ~~interact~~

→ In Android applications, a custom Dialog
is a pop-up window that overlays the c
activity and is often used to intescact
the uses, gathes Input os display. Infor
overview of a Custom Dialog Box:
puspose: Custom dialog typically consists of
UI elements like buttons, text views, imag
input fields. tailored to the specific int
You want to facilitate.
Customization: A Custom dialog typically
of various UI elements like buttons, text Vi
images or input fields tailored to the

interaction you want to facilitate.

Customization: Developers can design the dialog's appearance, layout, and behaviours according to their app's branding or specific requirements. This customization allows for creativity in design and functionality.

Simple example of creating and using a custom dialog in Android.

```
fun showCustomDialog() {
    val customDialog = Dialog(this)
    Custom.Dialog.setContentView(R.layout.
    Custom_Dialog)
    val messageTextview = CustomDialog.findview
                          ByID, Textview>
                          (R.id.message.Textview)
val okButton = CustomDialog.findviewByID
    <Button>(R.id.okButton)
 messageTextview.text = "This is a CustomDialog"
okButton.setonClicklistnes {
    CustomDialog.dismess()}
    CustomDialog.show()}
```

⇒ use cases of custom dialog box: login. Confirmation Dialog. setting. Informational pop-up. Media playback Controls.

Q-4. How do activities, Services and the Android Manifest file work togethes to make an Android app? Can you describe their

main roles and provide a basic example
they Cooperate to design a mobile app?

→ 1. Activities:
. Role: Activities represent individual
or UI Components in an Android app. They
the user interface and uses interactions.

2. Services:
Role: Services are background Components
perform long running opesations or handle
that don't required uses interface. They can ru
if the apps UI not visible.

3. Android Manifest file:
Role: The Android Manifest.xml file is like
blueprint. It declares the app's com po
defines how they interact with the
system and athes Components.

Examples: In Android Manifest.xml, You speci
Activities ase pust of yous app, theis Launch
pesmissions and service declarations. This file
blueprint for Android System to understand
structuse and behavios.

→ Class Main Activity: AppCompatActivity() {
Override fun onCreate(Saved Instanc
Bundle?){
Supes.onCreate(Saved InstanceStat
Set ContentView(R.layout.activi
Start service Button.setOnClick(
val service Intent = Intent(this, Notification service::cl

```
start Service ( service.Intent) }}}
```

```
class Notification Service : IntentService ("Notification
        service") {
override fun onHandle Intent (intent: Intent?) {
if. (intent!. = null) {
        CreateNotification () } }
private fun Create Notification () {
 Val chappelID = "my_channel"
 if (Build. vession. SDK_Int >= Build. vession_
 Codes. O ) {
  Val name = "my channel"
Val notification Manages = getsystemservice
 (notification Manages :: class. java)
 notification Manages . Create Notification chemal
        (Channel) }
 Val Builder = Notification. Compat. Builder
        (this, channel))
  .Set Small Icon (R. drawable. ic_launches_
        foreground)
  .Set Content Text ("This is notification
  from service.") }}
```

Q-5 How does the Android Manifest file impact the development of an Android application? provide an example to demostrate its Significance.

→ The Android Manifest file is a Crucial Component in the development of an Android application. It Serves several important purpose

and its Content Significantly impacts how the and
System interacts with and manages your App. Sig
of the Android Manifest file:
• App Configuration • Component Declaration • Pes
• Intent filters • App Lifecycle.

Example: <manifest xmlns: android = "http://s
android.com/apk/res/android"
package = "com.example.myapp">

<application
android: allowBackup = "true"
android: icon = "@mipmap/ic_launches"
android: label = "@string/app_name"
android: roundIcon = "@mipmap/ic_launche
android: theme = "@style/AppTheme">

<activity android: name = ".mainActivity"

<intent-filter>

<action android: name = "android.intent.action.
<category android: name = "android.intent.category.

</intent.filter>

</activity>

<activity android: name = ".SecondActivity"

.... Declare additional activities

</activity>

<uses-permission android: name = "and
Permission.Internet" />
</manifest>

.... Declare Required permissions here
</ 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 provide examples to classify your points.

→ Resources play a fundamental role in Android development by providing a structured way to manage assets, values, 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 arrangement.

<Button
Android:id = "@+id/ my Button"
Android: layout_width = "wrap_content"
Android: layout_height = "wrap_content"
Android: text = "click me" />

2. Drawable Resources:

Type: Images and drawable assets in the 'drawable' directory.

-Significance: Store graphics, icons, and images used in your app.

-Example: 'ic_launches.png' is the app's launcher icon.

3. String Resources:

type: strings defined in xml files under 'values'.

-Significance: store text strings, making it easier to provide translations and maintain consistency.

-Example: 'ic_launches.png' is the app's launcher

<string name = "app_name"> my App </string>

<string name = "welcome_message"> welcome to my app </string>

4. Colors Resources:

type: Colors defined in XML files under 'values'.

-Significance: Store Colors values, ensuring consistency in the apps design.

Example: "res/values/Colors.xml" defines style.

<style name = "My Button style">

<item name = "android: background"> @drawable/button </item>

<item name = "android: text Color"> @ color/
primary_color </item>
</style>

6. Dimension Resources:
typ: Dimensions defined in xml files under 'res/values'.
significance: Store dimension values, ensuring a consistent layout.
Example: 'res/values/dimens.xml' defines dimension resources.
<dimen name = "margin_large"> 16dp </dimen>
<dimen name = "padding_medium"> 8dp </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 service.

→ Contributions of Android Services:
1. Background processing: services allow apps to perform tasks in the background without blocking the user interface.
2. Long-running operating: Services are ideal

fos ha... ...quise ...

Complete, Such as playing music.

3. Intes-Componet Communication: Sesv... Components like activities, broadcast ... othes sesvices to Communicate with ... efficiently.

4. Fosegsound Sesvices: Andsoid Sesvices ... Fose gsound, even when the app isn't ... This is useful fos featuse that sequise ... uses inteaction, like music play back.

Pso cess of Developing an Andsoid Sesvice

1. Define the sesvice class: Cseate a new kotlin class that extends the 'Sesvice' ... methods like on Cseate(), on Destsoy(), on ... Commond() to deffine the behavios of Sesvice.

2. Configuse Sesvice in Manifest: Declase ... Sesvice in the Andsoid Mani fest. xml file ... the Andsoid system about its Existence ... Configusation. <Sesvice andsoid: name=".M...

3. Stast os Bind the Sesvice: Decide whe... Components. use StastSesvice() os bindSe...

4. Implement with othes Components: use a... mechanisms like specific logic yous Sesvice ... pesfosm its task

5. Hand life cycle: Release resources when they are no longer needed and consider using 'stopself()' or 'stopservice()'.

6. Interact with other components: use appropriate mechanisms like intents, broadcasts or callbacks to facilitate communication.

7. Foreground services (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 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.

05/10/23