2CET T5PE5: Mobile Application Development

Assignment = 1

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 knowledge, one notable trend in the mobile app industry that has was influencing the android platform was the rise of progressive web apps. PWAs are web-applications that often app-like experienced directly through web-browsers.

Impact on android app developers:

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 performances 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 and responsive user experiences, similar to PWAS.

- 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 users with 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 business the prioritize user 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

with evolving user-expectations

2. What is the purpose of an Inflater of layout in Android development and how does it fit into the architecture of Android layouts?

→

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. The process converts the XML code, which defines the structure and attributes of UI elements, into actual objects that can be manipulated programmatically.
Here is how it fits into the architecture 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, text fields, images, etc. as well as their attributes.

(2) Inflating Layouts: when an android app runs the layout xml files need to be converted into actual

view objects that can be displayed on the screen. this is where the 'Inflater' comes in.

③ Inflating with context:: The 'Inflater' requires a 'context' object to perform the inflation process. the 'context' provides essential information about the current state of the application.

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 application.

⑥ Manipulating views programmatically:: after inflation, developers can programmatically interact with the views, such as setting text, applying styles.

⑦ Displaying in the UI:: finally, the manipulated views can be added to the user interface using methods like 'set content view' or by adding them to the view hierarchy.

3. Explain the concept of a custom Dialog Box in android applications. provides examples to illustrate its use.

→

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 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 typically consists of various UI elements like buttons, textviews, images or input fields tailored to the specific interaction you want to facilitate.

- Customization: Developers can design the dialog's appearance, layout out behaviour 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)
    customDialog.setContentView(R.layout.custom-
                                dialog-layout)
    val messageTextView = customDialog.findViewBy
                          Id<TextView>(R.id.msgTextview)
    val okButton = customDialog.findViewById<button>(
                                R.id.OkBtn)
    messageTextview.text = "This is a customed dialog"
    OkButton.setOnClickListener {
        customDialog.dismiss()
    }
    customDialog.show()
}
```

- use cases of customDialogBox: . login . confirmation Dialog . settings . Information pop-up . Media playback . controls.

4. 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 :- Activities represent individual screens or UI components in an

Android app. they manage the user interface and user interactions.

(2) Services:: 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 not visible.

(3) Android Manifest file:: This is like the app's blueprint. It declares the app's components and defines how they interact with the android system and other components

- example:: In Androidmanifest.xml file is like the app's blueprint. It declares the app's components and defines how they interact with the android system to understand your app's structure and behaviour.

→
```
Class MainActivity: AppCompatActivity() {
    override fun onCreate(SavedInstanceState
                          :Bundle?).
{
Super.onCreate(SavedInstanceState)
SetContentView(R.layout.activity_main)
startServiceButton.setOnClickListener {
val ServiceIntent = Intent(this, NotificationService::class.java)
```

startService(ServiceIntent)

→ class NotificationService: IntentService("Notification-service"){
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.createNotificationChannel(channel)
}
val Builder = Notification.Compact.Builder(this, channelID)
.setSmallIcon(R.drawable.ic_launcher_foreground)
.setContentText("This is notification 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 impacts how the android system interacts with and manages your app. Significance of the Android Manifest file:

- App Configuration • Component Declaration
- Permission • Intent filters • 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.tools"
<application
  android:allowBackup="true"
  android:dataExtractionRules="@xml/data-
                    extraction_rules"
  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" />
</intent-filter>
</activity>
<activity android:name="SecoundActivity" />
....
</activity>
</application>
</manifest>
```

6. What is the role of resource 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, values, layouts and other elements used in your app. 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.

- ex.: 'activity_main.xml' defines the layout of your mainActivity, specifying UI components like buttons, textViews and their arrangement.

```
<Button
    android:id="@+id/myBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click Me" />
```

(2) Drawable Resource:

Type: Images and drawable asset 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 Resource:

Type: Storing images at multiple resolutions or sizes in the 'res/layout' directory.
Significance: Used to provide multiple versions of an image at different resolutions. Ensures that images look crisp and clear on screens with varying pixel densities.
example: ic_launcher - mipmap-hdpi, mipmap-mdpi, mipmap-xdhpi.

④ Colour Resource: type: colours defined in xml files under 'res/values'.
significance: store color values, ensuring consistency in the app's design.
example: 'res/values/colors.xml' defines colours resources.
<color name="primary_color">#007ACC</color>

⑤ Dimension Resource: type: Dimensions difined in xml files under 'res/values'.
significance: store dimension values, ensuring a consistent layout.
example: 'res/values/dimens.xml' defines dimension resources.

⑥ 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.

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

→ Contribution of Android Services:

① Background Processing: Service allow apps to

perform tasks in the background without blocking the user interfaces.

(2) Long-running operations: 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 Service: Android services can run in the foreground. This is useful for features that require ongoing user interaction like music playback.

Process of Developing an android Service:

(1) Define the service class: Create a new Java or Kotlin class that extends the 'Service' class or that extends the 'IntentService' class.

(2) Configure Service in manifest: Declare your service in the Android Manifest.xml file to inform the android system about its existence and configuration.

④ Start or Bind the Service: Decide whether you want to start your service or bind it to other components. Use startService() or bindService().

⑤ Implement Service Logic: In service class, implement the specific logic your service needs to perform its task.

⑥ Handle lifecycle: Release resources when they're no longer needed and consider using stopSelf().

⑦ Foreground Service: If your service needs to run in the foreground,

⑧ Optimization: Optimization your service for performance and resource efficiency to minimize battery usage.

19/10/23