Ganpat University | U.V. Patel College of Engineering

U. V. Patel College of Engineering

Ganpat Vidhyanagar, Mehsana-Gandhinagar Highway, Mehsana - 384012

Name : Prajapati Rutvi Sanjaybhai

Class : 5CEIT-B

Batch : 5B-1

En No : 210120111123

Subject: Mobile cApplication Development

[cAssignment : 1]

GHANSHYAM

Page No.

2CEIT5PES : Mobile Application Development

Assignment : 1

Q : 1] Based on your understanding, identify a recent business trend that has influenced the Android app developers and business in the mobile app industry.

Ans Based on my understanding, some recent business trend that has influenced the Android platform is the "rise of the metaverse." The metaverse is a shared virtual world where people can interact with each other and with digital objects.

→ It is still in its early stages of development, but it has the potential to revolutionize the way we use our smartphones and other devices.

→ Google is one of the companies that is heavily investing in the metaverse. In 2021, Google announced that it would be creating a new team to focus on developing augmented reality (AR) and virtual reality (VR) products.

→ The rise of the Metaverse is impacting Android app developers and business in the mobile app industry in a number of ways.

- It is creating new opportunities for developers to create innovative new apps and experiences.

GHANSHYAM

For example, developers could create apps that allow users to explore the metaverse, play games in the metaverse, or even create their own virtual world.

- The rise of the metaverse is changing the way that businesses interact with their customers.

For example, businesses could use the metaverse to create virtual showrooms or customer support center. also use the metaverse to host events or to launch new products.

→ Some examples how the metaverse is impacting Android app developers and businesses in the mobile app industry:

- New app Categories: Opportunities to create innovative new apps and experiences.
- New ways to monetize app: To monetize their apps in new ways.
- New ways to engage with Customers: To engage with their Customers in new ways. e.x: to provide Customer support.

→ Developers and buisnesses who are able to adapt to this trend will be well-positioned to succed in the future.

Q:27 What is the purpose of an Inflater or layout in Android development, and how does it fit into the architecture of Android layouts?

→ In Android Development, an "Inflater" refers to the LayoutInflater 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 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 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 files 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 is particularly valuable when you need to create UI elements dynamically, for example in response to user interactions or when working with RecyclerViews to display lists of items.

5) Binding Data: Once the View objects are created, they can be bond to them. This is typically done using data binding technique 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 screen.

Q:3] Explain the concept of a CustomDialogBox in Android applications. provide examples to illustrate its use.

→ A custom dialog box in Android is a dialog box that is created by the developes, rather them using one of the built-in dialogs. This allows the developer to create a dialog box with a custom layout and functionality.

→ To create a custom dialog box, the developer first needs to create an XML layout file for the dialog box. The layout file can contain any view objects that the developer wants to display in the 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.

- Customization: Developes can design the dialog's appearance, layout, and behaviour according to their app's branding or specific requirements. Their 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)
    val MessageTextView = CustomDialog.findViewById<
                          TextView>(R.id.messageTextVie
    val OkButton = CustomDialog.findViewById<Button>
                          (R.id.OkButton)
    messageTextView.text = "This is a Custom dialog!"
    OkButton.setonClickListener {
                          CustomDialog.dismiss() }
    CustomDialog.show() }
```

⇒ Use case of Custom display dialog box: . login . Confirmation Dialog . Settings . Informational pop-up . Media playback controls.

Q:4] How do activities, services and the Android Manifest file cook 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?

→ • Activities:

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

GHANSHYAM

- 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 apps UI not visible.

- Android Manifest file:

The AndroidManifest.xml 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.

Example: In AndroidManifest.xml, you specify which activities are part of your app, their launch modes, permissions and Service understand your App's structure and behavior.

→ Class MainActivity: AppCompatActivity () {
override fun onCreate (Saved InstanceState: Bundle?) {
Super.onCreate (Saved InstanceState)
Set ContentView (R.layout.activity_main)
StartServiceButton.SetonclickListener {
val ServiceIntent = Intent (this, NotificationService ::
Class.java) . StartService (ServiceIntent)
}
}
}

→ Class NotificationService : IntentService ("Notification Sevice") {
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(NotificationManager::class.java)
        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 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
- Intent Filters
- Permissions
- App LifeCycle

- Example:

```
<Manifest xmlns: android = "http://schemas.android.com/apk/res/
                                                android"
    package = "com.example.myapp">

    <application
        android : allowBackup = "true"
        android : icon = "@mipmap/ic_launcher"
        android : label = "@string/app_name"
        android : roundIcon = "@mipmap/ic_launcher_round"
        android : SupportSRt1 = "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>
    <uses-permission android : name = "android.permission.Intend" />

            ... 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 clarify your point.

→ 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 independant 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.
- 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/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 launches icon.

3) String Resources:
- Type: String defined in XML files under 'res/values'.
- Significance: Store text strings, making it easier to provide translations and maintain consistency.
- 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>

<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 consistent layout.
- Example: 'res/values/dimens.xml' defines dimensions 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.

-> Contribution of Android Service:

1] Background processing: Services allow 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 music.

GHANSHYAM

3] Inter-Component Communication: Services enable components like activities, broad-cast receivers and services to communicate with each other efficiently.

4] Foreground Service: Can run in the background, even when the app isn't in the foreground. This is useful for features that app isn't requires 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(), onDestory(), onStartCommand() to define the behaviours of your service.

- Configure Service in Manifest: Declare your service in the Android Manifest.xml file to inform the Android System about its existence and configuration.
  Ex:
  <Service android:name=".MyService" />

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

- Implement Service Logic: In Service class, implement the specific logic your services need to perform its task.

- Handle LifeCycle: Release resources when they are no longer needed and consider using 'StopSelf()' or 'StopService()'.

- Intract with Othes Components: Use cippsopsicute mechanisms like intents, broadcusts as Callbacks to facilitate Communications.

- Foreground Services (Opticonal): If yous Sesvice needs to sun in the foreground, 'StustForeground()'.

- Testing: Throughly test yous Sesvice to ensuse it functions as expected, including handling vasious Scenaricas like netcoak failuses.

- Optimaization: Optimize yous Sesvice fos pesfosmcnce cnd sesouse efficiency to minimize battesy usage.

- Essos Handling cnd logging: Implement psopes essos handling cnd logging mechanisms to diagnose cnd address issues that cnay occus cshile Sesvice is sunning.

5/10/23

GHANSHYAM