Name : Patel Nisarg A.

Class : CEIT-B

Batch : CSB-1)

En no : 210120111094

Sub : MAD

Assignment : 1

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 developer and business in the mobile app industry

→ One significant trend in the android app industry was the increasing emphasis on user privacy and data security

→ impact on android App developers

1. Enhanced permissions and consent:

→ developers had to be more transparent about the data their apps called and request explicit user content This meant redesigning permission dialogs and ensuring that users understood any certain data was being collected

2. Limitations on Advertising

→ For apps relying on advertising revenue, changed in ad tracking ~~and~~. Developers needed to adapt to those changes. Possibly exploring monetization models

→ impacts on Businesses

1. Compliance costs:

→ Businesses operating in the android app industry needed to allocate resources for compliance with stricter

data privacy regulations.

2. Reputation management:
→ Privacy breaches or mishandling of users data could result in severe reputational damage. Building and maintaining trust with users become even more critical

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

→ In android app development, Inflater helps turn your design plans into actual buttons, text boxes, and other things you see on your phones screen.

→ Architecture of LayoutInflator

1. XML Layout files: Developers design the layout & structure of UI elements in XML layout resource file.

2. Activity / Fragment: In the java or kotlin code of an android activity or fragment. Developers use the LayoutInflator to "inflate". This is typically done within the 'onCreate' method.

3 View Hierarchy: The result of inflating the layout XML is a hierarchy of view objects, with the root view being the top-level layout.

4. Data Binding & Event Handling: Developers often bind data to these views using data binding libraries or

handle user interactions by attaching event listeners

5. Rendering on the screen: The android system is responsible for rendering this hierarchy of views on the device screen according to the layout specifications defined in the xml file

Q-3 Explain the concept of customDialogBox in Android application. Provide examples to illustrate its use

→ A custom dialogBox in Android applications is a pop-up window that developers can design and customize to show specific information, receive input from users or perform actions without navigating to a new screen or activity. Custom dialog Boxes are helpful for displaying messages, alerts, forms, or any custom content in controlled and visually appealing manner.

→ 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, to the specific interaction you want to facilitate.

→ Customization: Developers can design the dialogs appearance layout and behavior according to their apps branding

ex: 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.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 . setting. Informational pop-up . Media playback controls

Q-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. Service:

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 apps UI not visible

3. Andsoid Manifest file:

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

ex: in androidmanifest.xml. You specify which activities are part of your app. their Launch models. permissions and service declarations. This file acts a blueprint for android system to understand your apps structure and behaviour.

→ Class MainActivity : AppCompatActivity() {
override fun OnCreate (Saved InstanceState: Bundle?) {
SuperOnCreate (Saved InstanceState)
SetContentView(R.layout.activity_main)
StartServiceButton.SetOnClickListener {
val ServiceIntent = 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(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 of

→ 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 android system interacts with and manages your App.

Significance of the Android Manifest file:

- App configuration
- Intent filters
- App Lifecycle
- Component Declaration
- Permissions

Ex: <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:supportsRtl="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.
                                    INTERNET" />
... 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 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 helps create flexible, maintainable and device independent

applications. The various types of resources and these significance with ex:

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 main activity, specifying UI components like buttons, textviews 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 'res/drawable' directory.
- Significance: Store graphics, icons, and images used in your app.
- Ex: 'ic_launcher.png' is the app's launcher icon.

3. String Resources:

- type: Strings defined in xml fields under 'res/values'.

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

- Ex: '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.

Ex: '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
- Ex: '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.
- Ex: 'res/values/dimens.xml' defines dimension resources
<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
- Ex: 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 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 receives and other services to communicate with eachother 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:

1. 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 AndroidManifest.xml file to inform the Android system about its existence and configuration. <service android:name=".myService" />

3. Start the servid: Decide whether you want to start service or bind it to other components. use startService() or bindService().

4. Implement Service logic: In service class, 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()'

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 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 address issues that may occur while service is running.