Abhay Hingrajiya-22012012024

Ganpat University | U.V. Patel College of Engineering
॥ विद्यया समाजोत्कर्षः ॥
U. V. Patel College of Engineering
Ganpat Vidhyanagar, Mehsana-Gandhinagar Highway, Mehsana - 384012

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 has was influencing the android platform was the rise of progressive web apps (PWAS). Pwas experinces 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 smother and more engagging users experience which set higher expectations for android app developers. This encouraged them to focus on improving the quality and performance of their apps to complete effectivly.

GHANSHYAM

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 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 android apps to keep up with evolving user expectations and technology trends.

22012024

Q.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 element into actual object that can be manipulated programmatically.

=> Here how it fits into the architechse 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. It is used to read the XML files and create the corresponding view objects.

3) Inflating with context: The 'Inflater' requires a 'context' object to perform the inflation process.

GHANSHYAM

The 'context is typically provides by an Activity or Fragment.

4) Attaching to parent: The 'Inflater' also huss the option to attach the inflated layout to a paret 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, adding event listeneas and more.

7) Displaying in the UI: Finally, The manipulated views can be added to the user interface using methods like setContentView or by adding them to the view hierarchy through their parent view group.

Q.3 Explain the concept of a customDialog Box in Android applications. Provide examples to illustrate its use.

⇒ In Android applications, a customDialogBox is a pop-up window that overlays the current activity and is often used to interact with the user, gather inputs or display information. Overview of a customDialog Box:

22/01/2024

→ 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 and behavior according to the 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.findViewById<TextView>(R.id.msgTextView)
    val okButton = customDialog.findViewById<Button>(R.id.okButton)
    messageTextView.text = "This is a custom dialog"
    okButton.setOnClickListener {
        customDialog.dismiss()
    }
    customDialog.show()
}
```

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 the cooperate to design a mobile app?

⇒ 1) Activites:
→ Role: Activities represent individual screens or UI components in an Android app. They manage the user interface and user interactions.

2) Services: Role of 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: The Android manifest.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 Android Manifest.xml, you specify which activities are part of your app, their Lanch modes, Permissions and Service declarations. This file acts as a blueprint for Android system to understand your App's structure and behavior.

→ Class Main Activity : ~~App CompatActivity()~~
App CompateActivity(){
override fun onCreate(savedInstanceState: Bundle?){
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
startservice Button.setOnClickListener{
val service Intent = Intent(this, Notification service::
Class.java)
startService(serviceIntent)
}
}

2201201202u

→ class NotificationServices: IntentService ("NotificationService") {
override fun onHandleIntent(intent: Intent?) {
if (intent != null) {
creat Notification()
}
}

private fun createNotification() {
val channelID = "my_channel"
if (Build.version.SDK_Int >= Build.Version_codes.O) {
~~val name = "my~~ val name = "my channel"
val notificationmanager.createNotificationChannel(chan
}
val Builder = Notification.Compat.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 impact how the android system interacts with and manages your App. Significance of the Android Manifest file.

GHANSHYAM

→ Example,

```
<?xml version="1.0" encoding="utf-8"?>
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" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name="SecondActivity">
        ...
    </activity>
</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 illustrate your points.

22012012024

=> Resources play a fundamental role in Android development by providing a structured way to manage assets values, layouts and others 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 UI components like buttons, text views and their arrangement:

```
<Button
    android:id="@+id/myBtn"
    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' directo
-> significance: Store graphics, icons and images used in your app.
-> example: "ic_luncher.png' is the app's luncher icon.

3) Mipmap Resources:
-> type: storing images at multiple resolutions or sizes in the 'res/layout' directory.

GHANSHYAM

→ Significance: Used to provide multiple version 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-mdpi, mipmap-xhdpi, mipmap-xxhdpi, mipmap-xxxhdpi

4) String Resources:
→ Type: Strings 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" containing string resources.
<string name="app_name">My App</string>
<string name="welcome">Welcome to my app</string>

5) 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">#00ACC1</color>

6) Style Resources:
→ type: Styles defined in XML files under 'res/values'
→ Significance: Define reusable styles for UI components.
→ example: res/values/styles.xml: define style.
<style name="MyStyle">
<item name="android:background">@drawable/my_button</item>
</style>

7) Dimension Resources:
→ type: Dimensions defined in XML files under 'res/values'.
→ Significance: Store dimension values, ensuring a consistent Layout.
→ Example: <dimen name="large">36 dp</dimen>

22012012024

8) 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 app 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 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 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 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 existance and configuration:
<service android:name=".MyService" />

3) Start or Bind the Service: Decribe whether you want to start your service or bind it to other components. use startService() or bindService()

4) Implement ServiceLogic: In service class, implement the specific logic your service needs to perform its task.

5) Handle Lifecycle: Release resources when they are no longer needed and consider using 'stopSelf()' or 'stopService()'

6) Interact with other components: Use appropriate mechanism like intents, broadcasts or callbacks to facilitate communication.

7) Foreground Service (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 diagnose and address issues that may occur while service is running.

6-10-23