MAD Assignment-1

Name : Raj Patel

Subject : Mobile Application Development

Enrollment No: 210120111126

Batch : 5B-4

Class : CEIT-B

Semester : 5

Branch : CE

Assignment - 1

Q1 Based on your understanding, identify a 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 was influencing the Android platform was the rise of progressive web apps (PWAs). PWAs are web browsers in

→ Impacts on Android App Developers :

1. Cross-platform Compatibility : PWAs alongside traditional android apps, 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, ~~This encouraged~~ and performance of their apps to compete 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 Businesses in the Mobile App Industry:

1. Cost Savings: Businesses could potentially save on app 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 businesses 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 businesses to 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 ~~for~~ android apps to keep up with evolving tech user expectation and technology tech.

Q2 What is the purpose of an Inflator of Layout in Android development and how does it fit into the architecture of Android Layouts?

⇒

→ "Inflater" refers to the Layout Inflater, which plays a crucial role in creating a user interface (UI) from XML 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 Inflator fits into the architecture of android layout:

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 with the UI.

2. Layout Inflation : When 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 Inflator : 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 create view objects in memory, such as TextView, Button, etc.

4. Dynamic UI Creation : Layout inflation is particularly valuable when you need to create UI elements dynamically.

5. Binding Data : Once the view objects are created, they can be further customized and data can be bound to them. This is typically done using data binding techniques 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 the screen.

Q3 Explain the concept of a custom Dialog Box in Android applications. Provide ex. to illustrate its use.

⇒ 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: These 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 button, text views, 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 the their app's branding or specific requirements. This customization allows dialog creativity

→ Simple example of creating and using a custom dialog:

```
fun showCustomDialog() {
    var customDialog = Dialog(this)
    customDialog.setContentView(R.layout.custom_Dialog)
    Var messageTextView = customDialog.findViewById<Textview>
                          (R.id.messageTextView)
    var okButton = customDialog.findViewById<Button>(R.id.OkButton)

    messgTextView.text = "This is a custom dialog!"
    ok.Button.setonClickListner {
        customDialog.dismess()
    }
    custonDialog.show() }
```

→ Use cases of custom Dialog box: login, confirmation Display, settings, Informational pop-up, media playback control.

Q4 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 :
   - Role : Servios are background components that perform long-running operations or handle tasks Tasks that dont required user interface. They can run even if the apps UI not visible.

3. Android Manifest File :

- Role : It is file is like the app's blue print. It declared the app's components and defines how they interact with the android system.

eg :

→
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState : Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        startServiceButton.setOnClickListener {

        val serviceIntent = Intent(this, NotificationServic::class.java)

        startService(serviceIntent)
    }
}
```

→ class NotificationService : IntentService("NotficationService"){

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 notificationManager = getSystemService(NotificationManager::class.java)

notificationManager.createNotificationChannel(channel)
}

Val Builder = Notification.Compat.Builder(this, channelID)

~~Small~~.
setSmallIcon(R.drawable.IC_launcher_foreground)
setContentText("This is notification from service")

}
}

Q5 How does the Android Manifest file impact the development of an android app? Provide an example to demostrate its significance.

=>

→ It is a crucial component in the development of an android app. It serves several important purposes, and its content significantly impacts how the android system interacts with and manages your app. Ex.

Significance of the Android Manifest File:

- App Configuration
- Permissions
- App lifecycle.
- Component declaration
- Intent Filters

→ 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>

<user-permission android:name="android.permission.Inlet"/>
   ... Declare required permission here ...
</application>
</manifest>
```

Q7 How does and 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: 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 activity, broadcast receivers and other services to communicate with each other

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 and 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 behaviour of your service.

2. Configure Service in Manifest : Declare your service in the android manifest.xml file, on information the Android system about its existence configuration
   eg : <service android : name = ".myService" />

3. Start or Bind the service : Implement the specific logic your service needs to perform its tasks.

4. Implement Service Logic : In service class, Implement the specific logic your service needs to perform its tasks

5. Handle 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 intent, 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 performs and resource optimization to minimize battery use.

10. Error Handling and Logging : Implement proper error handeling and logging mechanisms to diagnose and address issues that may occure while service is running

20/10/23