Jaydip B. Chungani   CE IT-B   SB2   21012021012

Mobile Application Development

Assignment - 1

Q-1 Based on your understanding, identify a recent business trend that has influenced the Android platform. Explain how this trend impact Android App developers and businesses in the mobile app industry.

- One recent business trend that has influenced the Android platform is the rise of mobile - Commerce (m-Commerce)
- The growth of m-commerce has led to an increase in demand for mobile Apps.
- This trend has had a significant impact on Android app developers and businesses in the mobile app industry

→ Android App developers

→ Increased demand for m-commerce apps:
- Android app developers now have the opportunity to develop app for a wide range of m-commerce activity.
- This has led to an increase demand for Android app development skills and expertise.

→ New Revenue opportunities
- The growth of m-commerce has created new revenue opportunities for Android app developer.

- For example, developers can earn revenue from ~~on~~ in-app purchase, advertising and subscription fee.

→ Need for new skills and expertise
- Android app developers who want to develop m-commerce apps need to have a good understanding of e-commerce and mobile payment processing.
- They also need to be able to develop secure and reliable app that can handle large volumes of transcations.

⇒ Business in the mobile App industry

→ Increased Competition
- The growth of m-commerce has led to an increase in competition in the mobile app industry.
- Business now need to develop high-quality and innovative m-commerce apps in order to attract and retain customer

→ Need to invest in Mobile App Development
- Businesses need to invest in mobile app development in order to reach their target customers and offer them a convenient and seamless experience.

→ New opportunities for Growth
- The growth of m-commerce has created new opportunities for business in the mobile app industry.

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

→ In Android development, an inflater is a mechanism used to convert an XML layout file into corresponding view objects in your app.

Purpose of Layout Inflater

1) Dynamic UI Generation

- Inflaters enables dynamic UI generation by allowing developers to create views programmatically at runtime based on predefined XML layout.

2) Reuse of layout components

- Inflaters facilitate the reuse of layout components. Instead of duplicating the same layout definition in multiple places within your code, you can define it once and in XML and inflated it whenever needed.

3) Separation of Concerns

- In Android, UI components are typically defined in XML layout files, promoting a seperation of concern between UI and business logic.

How it fits into Android Layout Architecture

1) XML layout definition

- Developers define the structure and appearance of UI Components using XML layout files in the "res/layout" directory.

2) Activity / Fragment Initialization

- In the "onCreate" method of an Activity or fragment, the layout inflater is used to set Content view.

3) Inflating layout

- The "Layout Inflater" class is employed to instantiate XML layouts, Converting them into view objects

Ex
```
val inflater = LayoutInflater.from(context)
val rootView = inflater.inflate(R.layout.my_layout, parent
                                                   , false)
```

4) Accessing Views

- Views within the inflated layout can be accessed programmatic

Ex
```
val myTextView = rootView.findViewById<Textview>(R.id.
                                                textview)
```

5) Setting Content View

- The inflated view hierarchy is set as the Content view of the Activity

ex
```
setContentView(rootView)
```

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

Ans A "CustomDialogBox" in Android is a Pop-up window that developer can design and customize to suit the specific needs and branding of their application.
- It's a way to present information, prompt user input, or confirm actions in a visually customized manner.
- The Android SDK provides the "Dialog" class, and developers often extend it or use the "AlertDialog" class to create custom dialog.

Ex MainActivity.kt

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.widget.Button
import com.example.myapp.CustomDialog
import com.example.myapp.R

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

      val showDialogButton : Button = findViewById
                                 (R.id.showDialogBox)
```

```
    showDialogButton.setOnClickListner
  {
    val customDialog = CustomDialog(this)
    customDialog.show()
  }.
 }
}
```

Q-4 How do activities, services, and the Android Manifest file work togethet 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?

Ans Activities
Roles : Activities are the user interface components of an Android App. They represent individual screen with which ~~users~~ users can interact. Each activity is a self contained unit with its own UI layout.

2) Services

- Roles : Services perform background tasks without a user interface They ~~can~~ are used for tasks that need to run independent of the UI, such as playing musics, fetching data from the internet, or ~~pre~~ performing other long running operations.

3) Android Manifest File
It is a configuration file that provides essential info about the app to the Android system. It declares the app's components, their properties and the permissions they requires.

→ Example of Activity

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

        val playButton : Button = findViewById (R.id.playButton)
        playButton.setOnClickListner
          {
              val intent = Intent (this, MusicService ::
                                            class.java)
              startService (Intent)
          }
    }
}
```

→ Example of Service

```
class MusicService : Service ()
{
   private lateinit var mediaPlayer : MediaPlayer
   override fun onBind (intent : Intent ?) : IBinder ?
   {
        return null
   }
   override fun onStartCommand (intent : Intent ?, flags : Int,
                                   startId : Int) : Int
```

```
  {
   mediaPlayer = MediaPlayer.create(this, R.raw.song)
   mediaPlayer.start()
   return  START_STICKY
 }
override fun onDestroy()
 {
    mediaplayer.release()
    super.onDestroy()
 }
}
```

⇒ AndroidManifest File

```
<manifest xmls:android = "http://schemas.android.com/apk
                                          /res/android"
  package = "com.example.musicplayer" >

  <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/AppTheame" >

   <activity android:name = ".MainActivity" >
     <intent-filter>
        <action android:name = "android.intent.action.MAIN"/>

        <category android:name = "android.intent.category.LAUNCHER"
                                                              />
     </intent-filter>

   </activity>
```

```
    <service android:name=".MusicService" />
  </application>
</manifest>
```

Q-5 How does the Android manifest file impact the development of an Android application? Provide an example to demonstrate its significance.

Ans The Android manifest file ~~impact~~ impact the development:

1) Declaration of App Components

- The manifest file declares all the components of an Android App, including activities, services, broadcast receivers and content providers

2) Setting Main Activity

- The manifest file specifies the main activity, which is the entry point of the app. This is the activity that is launched when the app is started.

3) Permissions and Security

- The manifest file is used to declare the permission that the app requires to access certain device features/data

4) Intent Filters

- Intent filters in the manifest file define how the app responds to implicit intents. They specify the types of actions, categories and data type the app can handles.

Ex
```
<manifest xmlns: android = "http:// schemas. android. com/ apk/ res/
                                                        android"
     package = "Com. example. Cameraapp" >
<uses- permission android : name = "android. permission: CAMERA"

 < 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.
     </intent-filter>                                  LAUNCHER" />
 </activity>

 <activity android : name = ". CameraActivity" >
 </activity>

 <Service android : name = ". CameraService " / >
 </application>

</manifest>
```

Q-8 What is the role of resource in Android development? Discuss the various type of resource and their significance in creating well-structured applications. Provide examples to clarify your points.

Ans In Android development, resources play a crucial role in creating well-structured and adaptable applications.
- Resources in Android are external elements such as images, strings, layout, colours and other assets that ar sepate from application code.
- they are used to provide flexiblility, maintainability an support for various device configurations.

1) string Resource

Role string resources are used to store text strings that are displayed in the user interface
- Storing strings in a seperate resource file make it easier to manage translations and adapt to adapt to different screen size.

ex res/values/strings.xml

```
<resources>
    <string name ="app_name"> My App </string>
    <string name = "welcome_message" > Welcome to My App!
                                              </string>
</resources>
```

GHANSHYAM

2) Drawable Resources

- Drawable resources include image and graphics used in the
- Different versions of images can be provided for different screen densities.

Ex res/drawable/icon.png

```
<ImageView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:src = "@drawable/icon" />
```

3) Color Resources

- It stores color values that can be easily reused ~~aroes~~ across the app. This allows for consistent theming and makes it simple to update the color scheme.

Ex res/values/colors.xml

```
<resources>
    <color name = "red"> #FF0000 </color>
    <color name = "Green"> #00FF00 </color>
</resources>
```

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

Ans Contribution to Functionality

1) Background Processing

- Services are ideal for tasks that should continue runni
even when the app is not activity interacting
with the user.
- Example include playing music, fetching data from the
internet.

2) Inter-Component Communication

- Services can communicate with other app components,
such as activitie or others services, using Android
inter-process communication mechanism like Intent
and Binder.

3) Long-Running Operations

- Services are suitable for executing long-running
operations, such as file download, synchronization
with server, or continuous sensor monitoring.

4) Foreground Services

- They are special type of service that provide a
persistent notification, ensuring that are user is
aware of ongoing tasks.

⟹ Process of Developing an Android Service

1. Create a Service class

Create a new class that extends the 'service' class. This class will contain the logic for your service.

```
class MyService : Service()
{
  override fun onBind (intent : Intent ?) : IBinder?
  {
      return null
  }
  override fun onStartCommand(intent : Intent ?), flags : Int , startId: Int)
                                                                 : Int
  {
      return START_STICKY
  }
  override fun onDestroy()
  {
      super.onDestroy()
  }
}
```

2. Declare the service in the manifest

- Declare your service in the AndroidManifest.xml file to let the Android system know about it.

```
<Service android :name = ".MyService" />
```

3) Start your Service

- you can start the service by creating an 'Intent' and using 'startService()'.

```
val intent = Intent(context, Myservice :: class.java)
context.startService(intent)
```

4) Handle Service Life cycle

- the service lifecycle methods ('oncreate()', ~~onstart~~ (, 'onStartCommand()', 'onBind()' and 'onDestroy()') allow you to manage the behaviour of your service at different points in its life.

[illegible]
6/10/23