Bheda sagar V. 21012021011 B-2
CEIT-B

MEHSANA DISTRICT EDUCATION FOUNDATION SANCHALIT

# U. V. Patel College of Engineering

GANPAT UNIVERSITY, KHERVA - 382 711 DIST. MEHSANA. (N.G.)

## Assignment - I

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.

→ AI/ML :- Artificial intelligence and Machine Learning.

→ AI & ML are integrated into mobile app development to make it smarter.

Recently apple launched core ML 3 an iOS machine-learning framework designed to help developer incorporate AI technology into their device apps. Not only this, but we also saw Google introduce new features to its Maps app which uses AI to improve user experience.

→ AI / ML have become one of the top mobile app development trends for 2023. In 2023, we can expect AI to reshape how apps are built making them more intelligent, more efficient and capable etc..

-> Augmented Reality and Virtual Reality.

-> An increasing numbers of apps incorporate augmented reality (AR) and virtual reality (VR) features. And this trend is not going away. According to ~~reach~~ research, Mobile AR was estimated at $12.61 billion in 2020, By 2030, it's forecast to grow to $184.61 billion.

-> Mobile IoT Apps

-> The internet of things has become mainstream in many sectors, from healthcare and agriculture to manufacturing and transportation. This resulted in increase development of mobile IoT apps.

-> Mobile Payments:

-> Currently, eCommerce is one of the most rapidly developing markets in the world, and mobile shopping is among the top trends consequently the demand for mobile payment is growing.

-> Cloud-Based Mobile Apps

-> Apps that integrate advanced technology such as AI / ML, IOT, etc.. require a lot of storage in mobile device's internal memory. But a gigabyte-sized app is not a solution of choice for users.

Impacts (~~PAW~~
PWA : Progressive web APP)

-> Cost-Efficiency : Developing native andloid app can be expencive and time-consuming. especially if a bussiness wants to cater to multiple platform. PWA offers a more cost effective solution.

-> Cross-Platform Compatibility : PWA are not limited to andloid devices, they can be accessed on various platforms, including iOS and desktop browsers

-> Improved User Experience : PWAs are designed to provide a samless user experience similar to native app. They can be installed on home screen, run offline, send push notification, etc.

-> Other impacts like

-> Performance and speed
-> APP Distribution
-> APP store independence.

etc..

Q.2 What is the purpose of an Inflator of layout in android development, and how does it ~~fit~~ fit into the architecture of android layots.

—> In android development, an "Inflater" is a cirecial component used to create and initializ. the primary purpose ~~of an~~ inflater is to conv contents of a layout XML file into corresponding view objects ~~ath~~ within your Android app. The primary purpose of an inflater is to convert an ~~XXL~~ XML layout file into a view hierarchy that can be displayed on the device's scree This process is essential for building the user interface of android apps.

—> Here's how the inflater fits into the architecture of android layouts.

1) Layout XML files :- In android, UI components are typically defined in XML layout file. These XML files describe the structure and appearance of the UI elements, specifying things like alignment of widgets, their attributes and their hierarchical relationships.

2) Activity or Fragment :- In the android app arc. an activity represents a single screen with a user interface, while a fragment represents a portion of an activity' UI. The activity or fragment is responsible for inflating and managing the layout.

3) Inflation :- When the activity or fragment needs to display the UI defined in an XML layout file, it uses an inflater to "inflate" the XML layout file. Inflation, in this context, means converting the XML layout into a set of view objects that can be added to the app's UI.

4) View Hierarchy :- The inflater creates a hierarchy of view objects, starting with the root view, ~~often~~ which is typically a viewgroup. This viewgroup contains child views organized as specified in the XML layouts file.

5) Display :- Once the view hierarchy is created it can be added to the app's UI, either by setting it as content view of an Activity or by adding it to a fragment's view hierarchy. The Android framework then handles rendering the UI on the screen.

Q3 Explain the concept of CustomDialogBox in android app. Provide examples to illustrate its use.

→ A custom dialog box in Android a user interface component that allows you to create a customized pop-up style window with your own layout and content. This is usefull when you want to display information, collect

input or provide interaction in a manner that differs from standard AlertDialog or system dialogs.

2) Create XML file

-> <?xml version="1.0" encoding="utf-8"?>

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CUSTOM DIALOG BOX"
    android:textSize="18sp"
    android:textColor="@android:color/black"
    android:layout_gravity="center_horizontal"
    android:layout_marginBottom="8dp" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="OK"
    android:id="@+id/dialogBt" />

</LinearLayout>
```

→ Kt file

```
import android.app.Dialog
import android.content.Context
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.widgem.Button

class CustomDialog (context : Context) :
                Dialog (context) {

 override fun onCreate (SavedInstanceState:
                Bundle ?) {
  super.onCreate (savelInstanceState)

 val inflater = LayoutInflater.from (context)
 val view = inflater.inflate (R.layout.custom
                dialog , null)
 setContentView (view)

 val button = view.findViewById <Button>
                (R.id.dialogbt)

 button.setOnclickListener {

                dismiss()
        }
    }
  }
```

-> Display box.

```
Val customDialog = CustomDialog (this)
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 they cooperate to design a mobile app?

-> Activity :

Role : Activity represent the user interface and serve as the building blocks of the app's UI. Each screen or UI components in your app is typically implemented as an activity. Activities manage the interaction with the user, handling user input and displaying UI element.

Ex -> Suppose you are building a simple Android app with two screen a login screen and home screen. You would create two activities - one for each screen. The login screen activity would handle user authentication, while the home screen activity would display the app's main content.

```
-> class LoginActivity : AppCompatActivity () {
        // --- code --- //
    }
class HomeActivity : AppCompatActivity () {
        // code //
    }
```

2) Service :-

Role :- Service are background components that perform long - running tasks independently of the UI. They are used to execute tasks that should continue running even when the app is not in the foreground. Services are typically used for tasks such as playing music, handling background network request, or updating data in the background

Ex Let's say you want your app to play music in the background even when the user switches to another app or locks the device. You would create a service to handle the music playback

```
class MyService : Service() {
    // Implement music playback logic
    here
}
```

3) Android Manifest :-

Role :- The androidmanifest.xml file is a crucial configuration file that provides essential information about your app to the android system. It specifies the app's components, ~~em~~ permission and how the app should behave when launched or interacts with other parts of the system.

ex In the android manifest file you declare the activities and service you've created and define their properties and relationship. You also specify permission your app needs and declare any intent filter for components that should respond to specific system events.

-> In practical, these components work together to create a seamless user experience. For example, when user launches your app, the android system leads the manifest file to understand which activity to start. The activity, in turn, interacts with the user interface. If your app plays music, you can use a service to ensure that music continues to play even when the user switches to another activity or app.

-> Remember that this is a basic overview, and real world app often involve more complexity and may include additional components such as broadcast, content provider etc...

Q 5 How does the Android Manifest file impact the development of an android app? Provide an example to demonstrate it significance.

-> The androidManifest.xml file plays a crucial role in the development of an Android app. It provides essential info to the android system about your app's components, permission, and behavior. The manifest file is significant for several reasons.

1) Declaring APP components:- The android Manifest file is where you declare all the components of your app, including activities, services, broadcast receivers, and content provider.

The declaration informs the android system about the various parts of your app and how they should interact.

2) Setting Entry Points:- The manifest file specifies which activity is the main entry point of your app.

3) Defining Intent Filter:- You can define intent filter for your app's components in the manifest file. Intent specify types of intents that each component can respond to. This is how other apps or system components can communicate with your app.

4) Requesting Permission :- If your app requires certain permissions to access device features or data like camera, contacts etc, you must declare these permission in the manifest file.

→ Ex, Suppose your app need camera and internet permission so

→ Manifest.xml

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

```
<uses-permission android:name="android.permission.CAMERA"
                 />
<uses-permission android:name="android.permission.INTERNET"/>

<application>
    <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=".ShareActivity">
        <intent-filter>
            <action android:name="android.intent.action.SEND"
                                  />
            <category android:name="android.intent.category.
                                  DEFAULT" />

            <data android:mimeType="image/*" />
        </intent-filter>
    </activity>

</application>
</manifest>
```

Q0 What is the role of resource in android development? Discuss the various types of resources and their significance in creating well-structured app provide example to clarify your point.

→ Layout Resources:-

This is the structure and appearance of user interfaces in your app. They are define in XML files and describe the arrangement and properties of UI elements.

→ Layout resource allow you to create consistent and visually appealing UIs that can adapt different screen sizes and orientation. They help separate the UI design from the code, promoting maintainability and reusability.

<LinearLayout

// Element info like textview, button

</LinearLayout>

→ 2) String Resources:- String resources text and string values that are used in your app's UI and code. They

are typically defined in XML file and can support multiple languages and locales.

-> String resources enable localization and internationalization of your app, making it accessible to global audience. They also centralize text content, making it easier to update and maintain translations.

```
<resources>
    <string name: "name">My Name</string>
</resources>
```

-> 3) Drawable Resources:

-> Drawable resources include image, icons, and graphics in your app's UI. They can be in various formats and are stored different drawable directories based on device screen density.

-> Drawable support high-quality graphics across different devices and screen resolutions.

Ex -> we can place different icon to app with use of this folder

4) Color Resources:-

-> Color resources define colors that are used in your app's UI and can be easily referenced from XML layout files and code.

-> By centralizing color definations, color resources make it simple to apply consistent color schemes across your app.

```
<resources>
    <color name="Primary_color">
        #007ACC</color>
</resources>
```

5) Dimenttsion Resources:

-> Dimension resource store numeric values, such as margins, Padding etc.

-> It is easy to maintain consistent spacing and sizing throughout app

```
-> <resource>
    <dimen name="text">24sp</dimen>
</resource>
```

6) Style and theme resources

-> Style and theme resources define appearance and styling of UI elements. Use for create custom themes for app.

-> It is use to maintain consistant look throught out app.

```
ex <resources>
    <style name="AppTheme" Parent="Theme.AppCompat
        Light.DarK">
    <item name="colorPrimary">@color/Primary</item>
</resources>
```

Q.7 How does an android service contribute to the functionality of an mobile app. Describe the process of developing android service

→ An android service contribute to the functionality of a mobile app.

→ An android service plays a crucial role in the functionality of a mobile app by allocating tasks to run in the background even when the app is not actively in use

→ Contribution of service:

1) Background Processing: service run tasks in the background ensuring the essential functions like music playback, location, tracking, etc without disrupting the user interface.

→ 2) Long Running oper: services are ideal for operations that take a long time to complete such as downloading large files or performing complex calculation without causing the app to freeze.

3) Foreground services: Some services can run in foreground displaying a persistent notification to keep the user aware of ongoing tasks like navigation or chat ~~etc.~~ app.

4) Inter-component comm: Service can comm with other app components through interfaces allocating data exchange and coordination

→ Developing an Android service:

1) Create a service class:
-> Extend the service class or one of its sub-classes like "Intent service" or "JobService"
-> Implement the service's functionality with the "onCreate" & "onStartCommand" method

2) Declare in the manifest

-> Register your service in the android manifest.xml file to make it accessible to the system and other components.

3) Service Lifecycle:
-> Understand the service's lifecycle method [onCreate, onStartCommand, onBind, onDestroy] and override them as needed
-> Service can run in three methods foreground, background, bound.

4) Start and Stop the service
-> Start a service using 'startService' or bind to it using 'bindService (Intent, service, connection, int).
-> Stop a service, when it's no longer needed using 'stopService (Intent)' or stopSelf().

5) Foreground service
-> To create a foreground service, provide a noti- that informs the user about ongoing tasks.

-> Use 'startForeground()' to start a service in the foreground mode

-> 6) Thread Management:
-> When performing time-consuming operations, consider using worker threads or AsyncTask to prevent blocking the main UI thread

-> 7) Communication
-> Use Intent extra, broadcast receiver, or interface to enable communication b/w services and other app components

-> 8) Clean Up and Resource Management.
-> Ensure that you release resources and stop the service when it's no longer needed to prevent unnecessary battery.

-> 9) Testing :-
-> Thoroughly test your service to ensure it works as expects, including scenarios, like app background task interruptions, and restarts

06.10.23