U. V. Patel College of Engineering

Ganpat Vidhyanagar, Mehsana-Gandhinagar Highway, Mehsana - 384012

Name :- Patel Divy Mineshkumar

Er. No. :- 21012811072

Class :- CEIT - B

Batch :- 5B - 1

Subject :- MAD Assignment - 1

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 impacts Android App developers and businesses in the mobile app industry.

Ans. One significant business trend that influenced the Android platform was the growing focus on privacy and data security. Google, the company behind Android, has been implementing stricter privacy measure and encouraging developers to follow suit.

- Here's how this trend impacts Android App developers and businesses in the mobile app industry:

1. **Privacy Regulations and Compliance :-**

- With the introduction of privacy regulations like GDPR and CCPA, Android App developers have had to ensure that their apps are compliant with these laws. This often involves obtaining explicit user's consent for data collection and implementing mechanisms for users to control their data.

2. **User's trust :-**

GHANSHYAM

- Building and ~~maintaining~~ maintaining user trust has become crucial. Apps that handle user data must be transparent about their data practices. Developers need to provide clear privacy policies and minimize data collection to gain user confidence.

3. Advertising & Monetization :- Changes in Android's advertising and tracking policies affect how businesses monetize their apps. Developers may need to find new ways to target users with ads or rely on alternative revenue streams, such as subscription models, to sustain their businesses.

4. App Store Guidelines :- Google Play Store has been tightening its guildlines related to privacy and data security. Developers must adhere to these guildlines to get their apps published, which can impact app development timelines and strategies.

5. Data Minimization :- Business have to be more selective about the data they collect and how they use it. This can lead to more efficient app development and potentially reduce legal and security risks.

6. Security Investments :- As data breaches becomes more costly and damaging to businesses, there's a growing need for increased security measures within apps. Developers need to invest in robust encryption, authentication, and data protection mechanisms.

7. User Experience :- Stricter privacy measures can sometimes lead to a less personalized user experience. Developers need to find a balance b/w respecting user privacy and providing valuable and service

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?

Ans. 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 enable dynamic UI generation by allowing developers to create views programmatically at runtime based on predefined XML layouts.

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 in XML and inflate it whenever needed.

③ Separation of Concerns :-

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

GHANSHYAM

- 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 typically used to set the content view.

③ Inflating Layouts :-

- The "LayoutInflater" 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,
                                parentView, false)
```

④ Accessing Views :-

- Views within the inflated layout can be accessed programmatically

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 OR 2 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 developers 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 dialogs.

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.showDialogButton)
```

```
showDialogButton.setOnClickListener
{
    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?

Ans. 1. Activities :-

- Role :- Activities are the user interface components of an AndRoid app. They represent individual screens with which users can interact. Each activity is a self-contained unit with its own UI layout.

2. Services :-

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

3. AndRoid Manifest File :-

- Role :- The AndRoid Manifest.xml file is a configuration file that Provides essential information about the app to the AndRoid

system. It declares the app's components, their properties, and the permissions they require.

- 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.setOnClickListener
        {
            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()
}
}
```

- Example AndroidManifest file :-

```
<manifest xmlns: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/AppTheme" >
```

```
<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 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 permissions that the app requires to access certain device features or 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 types the app can handle.

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

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

<service android:name = ".CameraService" />
</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.

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, layouts, colors and other assets that are separate from the application code. They are used to provide flexibility, maintainability and support for various device configurations.

1. String Resources :-

- Role :- String Resources are used to store text strings that are displayed in the user interface. Storing strings in a separate resource file makes it easier to manage translations and adapt to different screen sizes.

Ex :- res/values/strings.xml :-

GHANSHYAM

```
<resources>
    <string name = "app_name" > MyApp </string>
    <string name = "Welcome_message" > Welcome to MyApp! </string>
</resources>
```

2. Drawable Resources :

- Role :- Drawable resources include images and graphics used in the UI. 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 :-

- Role :- It store color values that can be easily reused 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 = "primary_color" > #3498db </color>
    <color name = "accent_color" > #ff4081 </color>
</resources>
```

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

Ans. Contribution to Functionality :-

1. Background Processing :-

- Services are ideal for tasks that should continue running even when the app is not actively interacting with the user. Examples include playing music, fetching data from the internet, or processing updates in the background.

2. Inter-Component Communication :-

- Services can communicate with other app components, such as activities or other services, using Android's inter-process communication mechanisms like Intent and Binder.

3. Long-Running Operations :-

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

4. Foreground Service :-

- Foreground services are a special type of service that provides a ~~Plase~~ persistent notification, ensuring that the 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?): ~~Bundle~~ IBinder?
    {
        return null
    }
    override fun onStartCommand(intent: Intent?, flags:
                                Int, startId : Int) : In
    {
        return START-STICKY
    }
    override fun onDestroy()
    {
        super.onDestroy()
    }
}
```

2. Declare the Service in the manifest :-

- Declare your service in the Android Manifest.xml file to let the Android System know about it.

= <service android:name = ".MyService" />

3. Start the 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()', 'onStartCommand()', 'onBind()' and 'onDestroy()') allow you to manage the behavior of your service at different points in its life.