Name :- chaudiya Harshil K.

Enr No:- 210120121031

Class:- CEIT-B

BATCH:- SB2

Subject :- MAD Assignment-1

Name:- Ghadiya Harshil K. class: CEIT-B
EN NO:- 21012021031 BATCH:- SB2

Ganpat University | U.V. Patel College of Engineering
|| विद्यया समाजोत्कर्षः ||

U. V. Patel College of Engineering

Ganpat Vidhyanagar, Mehsana-Gandhinagar Highway, Mehsana - 384012

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 business in the mobile app industry.

- one recent business trend that has influenced the Android platform is the rise of mobile-Comm--erce (m-Commerce)
- The growth of m-Commerce has led to an increase in demand for mobile Apps.
- This trend has 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 increased demand for Android app development skill and expertise.

→ New Revenue opportunities:

GHANSHYAM

- The growth of m-Commerce has Created new revenue opportunities for Android app developers.
- For example, developers can earn revenue from in app purchase, advertising and subscription fees.

→ Need for new skill 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 transactions.

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.

- Business 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 layouts?

→ In Android development, an inflator 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

- Inflater 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 separation 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 / Foragment Initialization.

- In the "onCreate" method of an Activity or on 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 inflator = LayoutInflater.from(Context)
val rootview = inflater.inflate(R.layout.my_layout, Parent, false)
```

4] Accessing views

- views within the inflated layout can be accessed programmatically

ex ~~val within the inflated layout can.~~
```
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
```
getContentView(rootview)
```

Q.3 Explain the Concept of a Custom DialogBox in Android applications provide examples to illustrate its use.

⇒ A "CustomDialogBox" in Android is a pop-up window that developer can design and customize to suite 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 "AlterDialog" class to Create Custom dialog.

Ex MainActivity.kt

```
import android.os.Bundle
import androidx.appCompact.app AppCompactActivity
import Com.example.myapp.CustomDialog
import android.Widget.Buttom
import Com.example.myapp.R

class MainActivity : AppCompactActivity()
{
  Override fun onCreate(Saved InstanceState: Bundle?)
  {
  Super.on(reate (Saved InstanceState)
  setContentview (R.layout.activity_main)
```

```
val showDialogButton : Button = findViewById
                                (R.id.showDialogBox)
showDialog Button.setonClickListner
{
  val CustomDialog = CustomDialog (this)
   CustomDialog.show()
  }
}
```

Q.4 How to activities, services, and the Android Maimfest 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: Activities

1) Roles:- Activities are the user interface Components of an Android App. They represent imdividual screen with which users can interact Each activity is a self Contained unit which with its UI Layout.

2] Services

- Roles:- services perform background tasks without a user interface they are used for tasks that need to run independent of the UI such as playing musics, fetching data from the internet, or performming 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 Activity

```
class MainActivity : AppCompactActivity ()
{
  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 late init var mediaPlayer : Mediaplayer
  override fun onBind (intent : Intent?) : Binder?
  {

    return null
  }
```

GHANSHYAM

```
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()
 }
}
```

#> Android Mainfest 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/AppTheme">
<activity android:name = "MainActivity">
  <intent-filter>
   <action android:name = "android.intent.action.MAIN" />
<add 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 activites, services, broadCast receivers and Content provider.

2] Setting Main Activity

- The manifest file specific 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.

GHANSHYAM

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 handles.

ex <manifest xmlns:android="https://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 resource in Android development? Discuss the various types of resources and their significance in creating well structured applications. Provide examples to clarify your points.

→ In Android development, resources play crucial role in creating well-structured and adaptable applications. Resources in Android are external elements such as image, strings, layout, Colors and other assets that are seperate from application code.

- they are used to provide flexibility, maintainbility 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 seperate resource file make it easier to manage translations and 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>
```

2] Drawable Resources

- Drawable resources include image 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

- It stores color value 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="red">#FF0000</color>
    <color name="green">#00FF00</color>
</resources>
```

Q.1 How does an Android service contribute to the functionality of a mobile applications? Describe the process of developing android service.

→ Contribution to Functionality

1] Background Processing

- services are ideal for tasks that should continue running 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 activities 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 servers, or continuous sensor monitoring.

4] Foreground Services

- They are special type of service that provide a persistent notification, ensuring that the user is aware 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 known about it

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

3] start your Service

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

```
val intent = IntentContext.MyService::class.java)
Context.startService(intent)
```

4] Handle Service Life Cycle

- The Service lifecycle method ('onCreate()', 'onStartCommand()', 'onBind()' and 'onDestroy()') allow you to manage the behaviour of your Service at different points in its life cycle.

6-10-23