Name:- Vedant S Pansuriya

Enrollment No - 21012011059

class - CEIT-B

Batch - 5B1

Subject: mobile Application Development.

Ganpat University | U.V. Patel College of Engineering

U.V. Patel College of Engineering
GANPAT UNIVERSITY, KHERVA-384012, DIST - MEHSANA. (N.G.)

Assignment : 1

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 business in the mobile app industry.

→ A recent business trend that has influenced the Android Platform is the rise of artificial intelligence (AI). AI is increasingly being used to develop new and innovative Android apps, and to improve the performance and functionality of existing app. For Android app developers, AI presents a new opportunity to develop innovative and useful apps. However, it also presents some challenges.

→ Here are some specific examples of how AI has impacted Android app developers and businesses in the mobile app industry:

→ Google has developed a number of AI-powered features for Android, such as Google Lens, Smart Reply, and Smart compose. These features make Android devices more intelligent and user-friendly.

→ A number of third-party app developers have developed AI-powered Android apps, such as chatbots, gaming apps, and image and video editing apps. These apps have become increasingly popular with users.

→ Some businesses in the mobile app industry are using AI to improve the performance and functionality of their apps. For example, some businesses are using AI to develop new battery optimizing techniques and to improve the accuracy of speech recognition and translation features.

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

→ The purpose of an inflator of layout in Android development is to convert an XML layout file into a view hierarchy. The inflator takes an XML layout file as a input and creates a view object for each element in the file. The inflator also sets the attributes of each view object based on the XML file.

→ The inflator is a key component of the Android layout architecture. It is responsible for creating the view hierarchy that is displayed on the screen. The view hierarchy is a tree of view objects, with each view object representing a different element on the screen.

3) Explain the concept of a custom DialogBox in Android applications. Provide examples to its illustrate it.

→ A custom DialogBox also known as a custom Dialog or custom Alert Dialog, is a user interface element that allows developers to create a custom pop up dialog with their own layout, content, and functionality. This is useful when you need to display information. gather input. or perform actions in a way that dosen't fit the standard Android dialog styles.

activity_main.xml:

```xml
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:android="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView
        android:id="@+id/dialogTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="custom Dialog"
        android:gravity="center"
        android:padding="16dp" />
    <Button
        android:id="@+id/closeButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="close"
        android:padding="16dp" />
</LinearLayout>
```

MainActivity.kt

```kotlin
import android.app.Dialog
import android.os.Bundle
import android.view.View
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
```

Scanned with CamScanner

```kotlin
class MainActivity : AppCompatActivity () {
    override fun onCreate (savedInstanceState: Bundle?) {
        super.onCreate (savedInstanceState)
        setContentView (R. layout. activity_main)
        val showDialogButton = findViewById <Button> (R.id. showDialogButton)
        showDialogButton.setOnClickListener {
            showCustomDialog ()
        }
    }

    private fun showCustomDialog () {
        val customDialog = Dialog (this)
        customDialog. setContentView (R. layout. custom_dialog)
        val closeButton = customDialog. findViewById (R. id. Button)
        closeButton. setOnClickListener {
            customDialog .dismiss ()
        }

        customDialog. show ()
    }
}
```

4) How do activity, services, and the android manifest file work together to make an android app? Can you Describe their main roles and provid a basic example of how they cooperate to design a mobile app?

→ When a user launches an android app, the system starts the main activity. The main activity is responsible for creating the user interface and displaying the initial screen.

→ If the app needs to perform long running task, it can start a service. service run in the background and do not have a user interface.

→ Android manifest file plays a role in all of this by declaring the components of the app and the permission that they need. The system uses this information to determine how to start and manage the app's components

Ex.

1. The user launches the app by tapping on its icon

2. The Android System starts the main activity of the app

3. The main Activity displays the user interface and handles user input.

4. The user taps on a button in the main activity to start a service that downloads a file from the internet.

5. The service is started and begins downloading the file

6. The user continues to interact with the main activity while the file is downloading

7. The service finishes downloading the file and sends a notification to the user.

8. The user taps on the notification to open the file

9. The android system starts an activity that can open the file.

S> How does the Android manifest file impact the development of an android application? Provide an example to demonstrate its significance.

→ Android manifest plays a crucial role in the development of an android application as it serves as the blueprint that defines the app's structure. Here are some key ways in which the Android manifest file impacts app development:

1. Component Declaration: The manifest file is where you declare all the components of Android app. These declaration inform the Android system about the app's entry point and functionality

2. Permissions: Declaring Permissions in the manifest file is essential for ensuring that app functions correctly & securely.

3. Intent filters: They enable app to handle implicit intents and respond to events like opening a specific file type.

4. Launch configuration: It specify which activity should save as the main entry point of app. This activity is launched when the app starts.

5. App metadata: The manifest file contains metadata about the app, such as its name, icon, version and package name.

Ex

```
<application
        android:icon=" @mipmap/ic-launcher"
        android:lable="@ String/app_name">
<activity
        android:name=".mainActivity"
        android:lable="@ String/app_name">.
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
<activity
    android:name=".ChatActivity"
    android:lable=" chat">
</activity>
< uses-Permission android:name="android.Permission.INTERNET"/>
<uses-permission android:name="android.Permission.SEND_SMS"/>

<receiver
        android:name=".SmsReciver">
<intent-filter>
    <action android:name="android.Provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
    </receiver>
</application>
```

- mainActivity is declared as the launcher activity, which is the first screen.
- chatActivity is another activity that allows users to chat with others.
- permissions for internet access and sending sms message are declared
- An intent filter is set up for 'SmsReceiver' component to listen for incoming sms message.

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.

→ Resources play fundamental role in Android development by providing a structured way to manage assests, values, layouts and other elements used in app. They help create flexible, maintainable and device independent application. The various types of resources and their significance with examples:

1. Layout Resources:
   - type: xml files in the 'res layout' directory
   - significance: Define the structure and appearance of the app's user interface
   - Ex activity_main.xml defines the layout of your mainactivity, specifying UI components like buttons, text views and their arrangement.

   < Button
       android: id = "@ + id /my Button"
       android: layout-width = "wrap_content"
       android: layout-height = "wrap_content"

```
android : text = "Submit" />
```

2. **Drawable Resources:**
   - type: images and drawable assets in the res/drawable
   - significance: Store graphics, icons and images used in your app
   - Ex: 'ic_launcher.png' is the apps launcher icon

3. **String Resources:**
   - type: Strings defines in XML files under res/value.
   - significance: Store text strings, making it easier to provid translation and maintain consistancy

   Ex res/Values/Strings.xml contains String resources
   ```
   <String.name = "app_name"> my app </String>
   <String name = "weclom_message"> Welcome </String>
   ```

4. **Color Resources:**
   - type: colors defines in xml files under 'res/Values'
   - significance: Store color values, ensuring consistancy in the app's design

   Ex res/values/colors.xml defines Color resources.
   ```
   <color name = "Primary_color"> #007ACC </color>
   <color name = "accent_color"> #FFA500 </color>
   ```

5. **Style Resources:**
   - type: styles defined in xml files under 'res/values'
   - significance: define resusable style for UI components

   Ex res/Values/Styles.xml
   ```
   <style name = "myButtonStyle">
       <item name = "android:background"> @drable /my_button
       </item>
       <item name = "android:textColors"> @colors/primary_color
       </item>
   ```

6. **Dimension Resources:**
   - type: dimensions defined in XML files under res/values
   - significance: Store dimension values, ensuring constraint layout

   Ex res/Values/dimens.xml defines dimession resources.

<dimen name="margin.large"> 16dP </dimen>
<dimen name="padding.medium"> 8dP </dimen>
7 Raw Resourses
  - type: files stored in the res/raw directory
- significance: store non-xml files, such as JSON data, audio
- Exc: store a JSON file for app configuration.

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

Contributions 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 than time to complete it
3. Inter component communication: services enable components like activities, broadcast recievers and other services to communicate with each other efficiently.
4. Background 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 interactions

Process of developing an Android service:
1. Define the service class: create a kotlin class that ends the services class. override methods like onCreate(), onDestroy() onstartcommand() to define the behavior of service
2. configure service in manifest: Declare service in the Android manifest.xml file to inform the Android system about its existences and configuration.

3. Start or bind the service: Decide weather you want to start a service or bind it to other components

4. Implement service logic: in service class implement the specific logic your service needs to perform its task

5 Handle Lifecycle: Release resources when they are no longer needed

6. Interact with other components: use appropriate mechanisms to facilitate the communication.

7. Foreground services (optional): if your service needs to run in the forground.

8. Testing: Thoroghly test your service to ensure it functions as expected.

9. optimization: optimize your service for performance and resource efficiency to minimize battery usage

10. Error handling and lagging: " implement proper error handling and lagging mechanisms to dignose and address issuess that may occur while service is running