Name : Adeshna Palak B.
Class : B (5B-2)
: MAD Assignment - 1
Enrollment no : 21012021006

Que: 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.

Ans : one significant trend in the Android app industry was the increasing emphasis on user privacy and data security.

→ Impact on Android App developers:

1. Enhanced permissions and consent:
→ Developers had to be more transparent about the data their apps collect and request explicit user content, this meant redesigning permission dialogs and ensuring that user understood why certain data was being collected.

2 Limitations on Advertising:
→ For apps relying on advertising revenue, changed in ad tracking and targetting due to privacy concerns affected their monetization strategies.

Developers needed to adapt to these changes, possibly exploring automatic monetization models.

3. Data Handling and storage:
→ Developers had to review how they handled and stored user data implementing stricter dat protec measures this could lead to increased development time & costs.

→ impacts on Businesses.

1. compilance costs:
→ Businesses operating in the android app industry needed to create resources for compilance with stricter data privacy regulations. this could, includ legal and technical measures to ensure data Protection.

2. monetization challanges.:
→ Businesses relying heavily on user data for advertising and personalized content faced challange in maintaining their revenue streames. They needed to find new ways to engage users and generaliz income.

3. Repulation management:
→ Privacy breches or mishandling of user data could result on several reputational damage. Building and maintaining trust with users became even more critical.

Que: 2) what is Purpose of an Inflator of layout in Android development and How does it fit into the Architecture of Android layouts ?

Ans: In Android app Development, think of the "Inflater" like a magic tool. it helps turn your design plans into actual buttons, text boxes, and other things you see on your phone's screen.

→ Purpose of Layout Inflator.

(1) Dynamic UI inflaction : Layoutinflater is used to create instances of Android view objects from xml Layout resource files at runtime.

(2) Reusability : it promotes the reusability of UI components by defining their structure and appearance in xml Layout files, making it easier to instantiate and populate them in different parts of an app.

(3) separation of concerns : Layout inflater helps maintain clear separation b/w the UI design and the code that manipulates and interacts with these UI elements.

→ Architecture of Android Layouts:

(1) XML layout files: Developers design the layout structure of UI elements in layout resource file.

(2) Activity / fragment: In the Java or Kotlin code an android activity or fragment developers use the layout Inflater "inflate" or parse the XML layout creating a hierarchy of view objects. This is typically done within the 'onCreate' method.

(3) View Hierarchy: The result of inflating the layout XML is a hierachy of view objects, with the root view being the top-level layout.

(4) Data Binding & Event Handling: Developers often bind data to these views using data binding libraries or handle user interactions by attaching event listeners.

(5) Rendering on the screen: The android system is responsible for rendering this hierarchy of views on the device screen according to the layout specifications defined in the XML file.

ue: 3) Explain the concept of custom DialogBox in Android Application. Provide examples to illustrate its use.

ns: A Custom Provide examples to illustrate is a Pop-up window that developers can design and Customize the to show specific information, receive input from users or perform actions without navigating to a new screen or activity. Custom Dialog Boxes are helpful for displaying messages, alerts forms, or any custom content in controlled and visually appealing manner.

(1) Design Flexibility : Custom DialogBoxes allows developers to create unique and tailored user interface

(2) Contextual use : They are typically used when you want to capture user input or show information without taking the user to a different screen.

(3) user interaction : Custom Dialog Boxes can Contain buttons, text-fields, checkBoxes or any other UI element, allowing users to interact with the content inside the dialog.

→ Examples of custom Dialog Box uses:

(1) confirmation Dialog: A common use case is asking the user for confirm before performing a critical

(2) Login or Registration Dialog: instead of navig to a seprate screen for log in registration, a custom dialog bo pop up, Prompting the user to ent their credentials.

(3) Error messages: when there's an error, such as network issues or invalid input, a custom dialog can display an error message with details. helping the user understand and cor the Problem.

(4) Date & time Pickers: you can create a custom dialog Box for selecting dates or times, Providing a more user-friendly way to input this infor

Code:

```
import android.app.AlertDialog
import android.content.DialogInterface
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
```

lass your Activity : APP compact Activity() {
override fun on create. (saved Instance state?
Bundle ?)

{

Super. on create (saved Instance state)
set Content view (R. layout. activity. main)
val builder = Alert Dialog Builder (this)
builder. set Title ("custom Dialog example")
builder. set messager ("This is a custom
dialog box ex")
builder. set positive Button ("OK")
{ dialog, which → }
val dialog = Builder. create()
}
}
dialog. show()

ue: 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 how they cooperate to design a mobile app?

Ans: Activities, Services, the Android manifest file are essential components in the android app architecture each with distinct role that contribute to the functionality and behaviour of an app.

1. Activities:
→ Role : Activities represent the user interface and

(3) Android manifest file :

→ in the manifest file, you declare the activities and services used in your app.

→ you specify Permission like, "INTERNET" to allow the app to access the internet for cloud backups.

→ The manifest file also defines which activity to stand when the app launches.

```
<manifest xmlns:android = 'http://schemas
                          android.com/apk/res/
                                  android
    Package = "com.example.mynotesapp">
  <application>
    <activity android:name = ".mainActivity">
      <intent-filter>
      <action android:name =
                  'android.intent.action.
                          MAIN" />
      <category android:name =
                  "android.intent.category.
                          LAUNCHER" />
```

Que: 5 How does the Android manifest file impact the development of an android application? Provides an example to demonstrate it's significance.

Ans : The Android manifest file impacts app development by :

→ resources in android development are components that helps you create well structured and flexible applications. they serves several purposes, such as seperating code from content, adapting to different devices and simplifying localization. Here are the main types of resources and their significance.

(1) Layout Resources:

- xml layouts: These define the structure and appearance of your app's user interface. They help keep the UI separate from code logic, making it easier to maintain of adopt.

- Example: A layout xml file specifies how elements like buttons and text fields are arranged on the screen.

(2) Drawable Resources:

- images and icons: Drawable resources store image, icons and other graphics used in your app. Diff versions can be provided for diff. screen densities.

- Example: you might have 'ic_launcher.png' for the app icon & separate versions for low, medium & high-density screens.

(3) String Resources:

- Text and strings: storing text in resource file allows for easy localization and updates without modifying code.
  - Example: A string resource ('app_name') contains the app's name, which can be changed for different languages.

(4) color resources:

- colors: By defining colors in resources, you can maintain a consistent color scheme across your app and easily switch themes.
  - Example: A color resource (Primary_color) define the primary color used in the app's UI elements.

(5) style Resources:

- Themes and styles: styles define the appearance of UI elements, making it simple to apply consistent styling across app.
  - Examples: you can create a custom style (AppTheme) to define fonts, colors and other visual attributes.

(6) Dimension Resources:

- sizes and Dimensions: storing sizes and margins in res file moves it easy to adjust layouts for different screen sizes and orientations.

- Example : A dimension resource (margin-small) defines a consistent margin size for elements.

(7) Raw Resources :

- Raw Data : you can store non-compiled resources like audio, video, or text files in the 'res/raw' directory.

  - Example : storing a JSON file in the 'Raw' folder for configuration data.

(8) Animations and Drawable Animation Resources

- Animation : you can define animations in xml resource file, making it simple to reuse and apply animations to UI elements.

- Example : A resource file (fade-in.xml) can define a fade in Animation for an image view

Que: 7) How does an android service contribute to the functionality of a mobile application? Describe the process of developing an android service. Write in simple language and include main Points.

Ans: An Android services plays a crucial role in the functionality of a mobile application by allowing tasks to run in the background, even when the app is not actively in use.

- **contribution of Android service:**

(1) Background Processing: services run tasks in the background ensuring the essential functions like music, playback, location tracking or data syncing can continue without distrupting the user interface.

(2) long - Running operations: services are ideal for operations that take a long time to complete such as doconloading longe files or performing complex calculations 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 applications.

(4) inter - component communication: services can communicate with other app components (activities, fragments) through interfaces, allowing data exchange and coordination.

Developing an android service:

(1) create a service class:
→ Extend the 'service' class or one of its subclasses like 'IntentService' or 'JobService'.
→ Implement the service's functionality within the 'onCreate' & 'onStartCommand' methods.

(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 methods [onCreate, onStartCommand, onBind, onDistroy] and override them as needed.
→ Service can run in three methods: foreground, background, or bound choose the appropriate mode based on your app's requirement.

(4) start and stop the service
→ start a service using 'startService(Intent)' or bind to it using 'bindService(Intent, ServiceConnection, int)'.
→ stop a service when it's no longer needed using 'stopService(Intent)' or 'stopSelf()'.

(5) foreground - services:

→ To create a foreground service, provide a notification that informs the user about ongoing tasks.

→ use 'start foreground.() 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) Communications:

→ use intent extras, broadcast, receivers, or interfaces to enable communication b/w services and other app components.

(8) cleanup and resource management.

→ Ensure that you release resources. and stop the services when it's no longer needed to prevent unnecessary battery drain.

(9) Testing

→ thoroughly test your service to ensure it works as expected, including scenarios like app backgrounding, task interruptions and restarts.

6-10-23