Assignment - 1

Ganpat University | U.V. Patel College of Engineering

U.V. Patel College of Engineering

GANPAT UNIVERSITY, KHERVA-384012, DIST - MEHSANA. (N.G.)

Name :- Javiya Shreya J.
Enrollment No :- 210P2011032
Class :- CEIT - B (5B-1)
Subject :- MAD (Assignment - 1)

1. Based on your understandings. Identify a recent bussiness trend that has influenced the android platform. Explain how that trend impact android developers and bussiness in mobile app industry.

→ One significant trend in android app industry was the increasing emphasis on user privacy and data security

→ Impact on Android app developers

1. Enhanced Permission and consent :-
Developers had to be more transparent about the data their apps collect and request explicit user content. This method redesigning permission dialouges and that users understood why certain data was being collected.

2. Limitations on Advertising :-
For apps relying on d advertising revenue, changed in an tracking and targeting strangies Developers needed to adopt to these changes, possibly exploring automative monitization models.

3. Data Handling and storage :-

→ Developers had to review how they handled and stor user data implementing data protection measures. This could lead to increased development time & co

→ Impacts on Bussiness

1. Compliance Cost :-
Bussinesses operating int the android app industry needed to allocate resources for compliance within stricter data privacy regulations. This could include legal and technical measures to ensure data protect

2. Monitization challenges
Bussiness relying heavily on user data for advertising and personalized content placed challenges in maintain their sequence stream. They needed to find new ways to engage users and generalize income.

3. Reputation Management
→ Privacy breaches or mishandling of user data could r in severe reputational damage. Building and maintai trust with users become even more critical.

2. What is purpose of an Infactor of layout in Android development and how does it fit into the architechture of Android layouts?

→ In android app development, think of the 'Inflator' of layout in Android development tool. it helps turn your design plans into actual buttons, text boxes, and ot things you see on your phone screen.

→ Purpose of Layout Inflater.

1) Dynamic UI Inflation :- Layout Inflater is used to creat instances of android view objects from xml layout resou files at runtime.

Architecture of Android Layout

XI) Dynamic UI Inflation

1) Xml Layout files: Developer design the layout a structure of UI elements in xml layout resources files.

2) Activity / fragment: In the login or kotlin code for an android activity or fragment, developers use the layout inflation to "inflate" or parse the xml layout files, 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 hierarchy of view objects with the root view being the top level layout.

4) Data Binding & Event Handling:- The android system is responsive for rendering this hierarchy of views on the device screen according to the layout specification defined in xml file.

3. Explain the concept of custom Dialog Box in Android application, produce example to illustrate its use.

→ A custom dialoge box in android application is a pop-up window that developers can design and customize to show specific information recieve input from user or perform actions without navigating to a new screen or

activity. custom dialog boxes are helpful for displaying message, alerts forms, or any custom content in controlled and visually appealing manner.

2) 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.

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 code that manipulates and interacts with these UI elements.

→ Architechuture of Android Layout

1) XML Layout Files :- Developers design the layout a structure of UI elements in XML layout resources file.

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

3) User Interaction : Custom dialoge boxes can contain buttons, text, fields, check boxes or any other UI element, allowing users text - to interact with the content inside the dialoge.

⟹ Examples of custom Dialouge Box User

1) Confirmation Dialog : A common use case is asking the user for confirmation before programming a critical action.

2) Login or Registration Dialog: a separate screen for login or registration, a custom dialog box can pop up, prompting the user to enter their credentials.

3). Error message: When there is an error, such as invalid input, a custom dialog can display an error message with details, helping the user understand and correct the problem.

4) Date & time picker :- you can create a custom dialog box for selecting dates or times, providing a more user friendly way to input this information.

Code:-

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

class YourActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val builder = AlertDialog.Builder(this)
        builder.setTitle("Custom Dialog example")
        builder.setMessage("This is a dialog box ex")
        builder.setPositiveButton("OK") { dialog, which ->
        val dialog = builder.create()
    } dialog.show()
```

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?

→ Activities, services, and the android a manifest file are essential components in the android app architecture each with distinct role that constribute to the functionally and behaviour of an app.

1) Activities :-

ROLE :- Activities represent the user interface and screen of an android app they handle user in interactions display UI elements and manage the UI flow.

Example :- Imagine a simple note-taking app. Each screen of the app such as the note list note editing, and setting, can be implemented as separate activities.

2) Service :-

ROLE :- Services run in the background and perform long-running or background takes without a user interface.

Example:- In our note-taking app, you might have a service that periodically backs up notes to a cloud server without showing a user interface.

3) Android manifest file:

→ Role:- The android manifest file is a configuration file provides essential information about the app to the android system. It declares the app's components, permission and other settings.

Example:- In the manifest file, you define which activities are part of your app, specify permissions & devices, declare services your app uses.

→ How they cooperate

1) Activities:

→ The app starts with an activity showing a list of notes.

→ When the user tap on a note, another activity open to display and edit the note's content.

→ Users can navigation between activities using buttons or gestures.

2) Services

→ While the user is using the app, a service runs in the background to periodically save the user's notes to a cloud storage.

→ This service doesn't have a user interface; it operates independly to ensure data is continuously backed up.

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 start when the app launchers

```
<manifest xmls:android = "http://schemas.android.com/apk/cod/package = "com.example.manifest app">
  <application>
    <activity android:name = ".MainActivity">
      <intent-filter>
        <action android name = "android.intent.action.MAIN" />
        <category android:name = "android:intent.category.LAUNCHER" />
```

5. How does the Android Manifest file impact the development of an android application? provides an example to demonstrate its significance.

→ The android manifest file impacts app dependenty by :-

1. Component declaration : Declaring app components to define the apps structure

ex <activity android:name=".mainActivity"/>

2) App permissions : specifing permissions for accessing device resources.

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

3) Intentfilters : Defining how the app responds to ea actions or requests.

eg. Registration to open PDF files when tapped.

```
<activity android:name="pdfViewer.Activity">
    <intent-filter>
        <action android:name="android.intent.action.VI
        <category android:name="android.intent.category
        <data android:mimeType="application/pdf"/>
    </intent-filter>
</activity>
```

6. What is the role of resources in Android devel Discuss the various types of resources and the significance in creating well structured applicati provide example to clarify your points

→ Resources in Android development are essential components that helps you create well-structred flexible applications they serves several purposes, as seperating code from content, adapting to dy devices, and simplifying localization. Hence are the main types of resources and their significa

1) Layout Resources:

- XML layout :- These define the structure and appeara

and appearance of your apps user interface. They help keep the UI separate from code logic, making it easier to maintain or adapt.

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

2) Drawable Resources
- images & icons :- Drawable resources store images, icons, and other graphics used in your app. different version can be provided for diff. screen densities.

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

3) String Resources:
- Text and strings: storing text in the resource files 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

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

5) Style Resource :
- Themes and styles : styles define the apperance element, making it simple to apply consis styling across app.
- Examples : you can create a custom style (App to define fonts, colors and other visual attributes.

6) Dimension Resources
- Size and dimentions : storing size and margins resourse file moves it easy to adjust layou for different screen size and otientations.
- Example : A dimension resource (margin_small) defines a consistent margin size for elements.

7) Raw Resources :
- Raw data : you can store non-compiled resources audio, vedio or text, files in the 'res/raw' dire
- Example : A dimension resource (margin_x storing JSON file in the 'raw' folder for configura data.

8) Animations & Drawable Animations Resources.
- Animation : you can define animations in xml resource file making it simple to reuse a apply animation to UI element

7. How does any 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 point.

→ An android service 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 acting in use.

- Contribution of Android Service

1) Background processing: Services run tasks in the background ensuring the essential function like music playground, location tracking or data syncing can continue without disintipting the user interface.

2) Long-Running operations: Services are ideal for operations that take long time to complete such as downloading large files or performing complex calculations without causing the app to freeze.

3) Foreground services: Some services can run in forgr displaying a persistent notification to keep the user aware of angoing tasks like navigation or chat applications.

4) Inter - component communication : Services
communication 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 AndroidManifest.xml file
to make it accessible to the system and other
components.

3) Service Lifecycle :
understand the Service's lifecycle methods [onCreate
onStartCommand, onBind, onDestroy] and override
them as needed
→ Service can run in three methods :
foreground, background, or bound choose the
appropriate mode based on your app's requirements

4) Start and stop the Service
→ Start a service using 'startService(Intent)' or bind
to it using 'bindService(Intent, ServiceConnection
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 foreground in the fore ground mode.

6) Thread Management
→ When performing time-consuming operations consider using worker threads or asynctask to oppi prevent blocking the main UI thread.

7) Communications
→ Use intent extras, broadcast, services, or interfaces to enable communication b/w services and other app components.

8) Cleanup and Resource Management.
→ Ensure that you release resource and stop the resource 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 restart restarts.

05/10/23