

* Name : Ratiyaguru prasad. M.

* Class : CEIT/B

* Batch : 5B2.

* Enrollment. No : 21012011197

* Subject : Mobile Application Development.

Assignment :- 1.

Q.1) Based on your understanding, Identify a recent business trend that has influenced by the Android platform. Explain how this trend impacts Android App developers and business in the mobile App industry.

→ In recent business trend, Android is the most popular mobile operating system in the world. we have approx 6.378 billion mobile users in world and mostly they are using android as operating system.

→ As, per statistics In 2022 play store have 90,000 new mobile apps. It simply means that android having very big impact in recent business trend.

→ Now, this trend make big impacts on App developers, because as per statistics mobile and users are ~~passing~~ increasing day by day and their requirements also getting increasing so it's very bright future for android app developers in this mobile App industry.

→ Some future App developments Topic for developers.

- Augmented Reality (AR).
 - IoT
 - chatbots
 - cloud computing
 - AI
- Etc...

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?

→ In android development, an Inflater, specifically a layout Inflater, is a crucial component that is used to instantiate and create instances of view objects from XML layout resource files. Its primary purpose is to take an XML layout file and turn it into a corresponding hierarchy of view objects, which can then be displayed on the screen.

• Here, how inflater fits into architecture of android :-

1). XML Layouts :- In Android, UI layouts are typically defined using XML files. These XML layout files describe the arrangement and attributes of UI (Text views, buttons, Imageviews) within an activity or fragment. It is readable and convenient way to design UI.

2). Activity or fragment :- Activities and fragments are the building blocks of android app. They represent individual screens or parts of the UI. These components need to inflate the XML layout files to display their UI to user.

3). Layout Inflater :- This is where the layout inflater comes into play. It's an essential part of the Android framework that allows you to dynamically create instances of view objects from XML layout files.

4). View Hierarchy :- The view hierarchy created by the layout inflater forms the structure of the UI for an activity or fragment. It includes UI elements defined in the XML layout file, organized in a tree-like structure.

→ Here's basic Example how we use inflater in Android:

val inflater = LayoutInflater

val rootview = inflater.inflate(R.layout.activity_main,
null)
setContentView(rootview).

Q.3) Explain Concept of custom dialogue box in Android Applications. provide Examples to illustrate its use.

→ A custom dialogbox in androids is user interface element that allows developers to create and display a customized popup dialog on the screen. Unlike the standard dialogs provided by the Android framework, custom dialogs give developers complete control over the layout, appearance, and behaviour of the dialog. It used to present information, gather user input, or perform actions without navigating to a different activity or fragment.

1) Design and flexibility:

→ custom dialog boxes allows developers to create unique and tailored user interface.

2) Contextual use:

→ They are typically used when you want to capture input or show information without taking the user to a different screen.

3) User Interaction:

→ custom dialog boxes can contain buttons, textfields, checkboxes or any other UI element inside the dialog box.

- Example of custom dialog box uses ..
- 1). Confirmation dialog:-
 - A common use case is asking the user for information before performing a critical action.
- 2). Error message:-
 - When there is an error occurs such as network issues or invalid input, a custom dialog can display an error message with details, helping the user understand and correct the problem.
- 3). Date and time picker:-
 - You can create a custom dialog box for reflecting date or time, providing a more user-friendly way to input this information.

* Code:

```

import android.app.AlertDialog
import android.app.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 Box Example")
        builder.setMessage("This is a custom dialog")
        builder.setPositiveButton("OK") { dialog, which ->
            val dialog = builder.create()
            dialog.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 how they cooperate to design a mobile app?

→ Activities, services and 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 screen of an android app they handle user interactions, display all elements and manage UI flow.

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

2). Services:-

→ Role:- Service run in the background and perform long-running or background task without UI

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

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 and declare services your app uses.

* How they cooperate?

1). Activities: The app starts with an activity showing a list of notes.

- When the user taps on a note, another activity opens to display and edit the note's content.
- Users can navigate b/w activities using buttons or gestures.

2). Services: While the user is using the app, a service runs in background to periodically send the user's notes to cloud storage.

- This service doesn't have a UI but operates independently 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. launcher.

```

<manifest xmlns: android = "http://schemas.android.com/apk/res/android"
  package = "com.Example.myname.app" >
  <application>
    <activity android: name = ".main Activity" >
      <intent-filter>
        <action android: name = ".main Activity" android: intent-
          action.MAIN" />
        <category android: name = "android.intent.
          category.LAUNCHER" />
    </activity>
  </application>
</manifest>
  
```

Q.5) How does the Android manifest file impact the development of an android application provider on Example to demonstrate it's significance.

→ The android manifest file impacts app development by

1). Component Declaration: Declaring app components to define the app's structure.

Ex: <activity android: name = ".main Activity" />

2). App permissions: specifying permission for accessing device resources.

Ex: <user-permission android: name = " android.permission.
 ~~APPNAME~~" />

3): Intent filters:- Defining how the app responds to external actions or requests.

Ex:- Registering to open PDF files when tapped.

```
<activity android: name = ". Pdf viewers. Activity">  
<intent-filter>  
<action android: name = " android. intent. action VIEW" />  
<category android: name = " android. internet. category. DEFAULT" />  
<data android. mimeType = " application/ PDF " />  
</intent-filter>  
</activity>
```

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 example
to clarify your points.

→ Resources in Android development are essential components that helps you create well-structured and flexible app. They serve several purposes, such as separating 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:- They define the structure and appearance of your app's UI. They help keep the UI separate from code logic, making it easier to maintain and adapt.

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

2).

Drawable Resources:-

- Image and Icons:- Drawable resources store images, icons and other graphics used in your app. Different versions can be provided for different screen densities.
- Example:- You might have "Tc-launcher.png" for the app icon of separate versions for low, medium and high density screens.

3).

String Resources:-

- Text and strings:- Storing text in resource files allows for easy localization and updates without modifying code.
- Example:- A string resource `c"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 schema across your app and easily switch themes.
- Example:- A color resource (primary color) defines 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 (APP theme) to define fonts, colors and other visual attributes

6). Dimension Resources::

→ sizes and dimensions:: storing sizes and margins in res file makes it easy to adjust layouts for different screen sizes and orientations.

• Examples:: 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

• Examples:: 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 in XML to UI

• Examples:: A resource file (fade-in.xml) can define a fade in animations for an image view.

Q.7) How does an android service contribute to the functionality of a mobile application? Describe the process of developing an android service. ~~with simple language and include code portion~~

→ An android service plays a crucial role in the functionality of mobile application by allowing tasks to run in the background, even when the app is not in actively in use.

• Contribution of Android Services:-

1) Background processing:- Services run tasks in background ensuring the essential fun's like music playback, location tracking, or data sending can continue without disrupting the user interface.

2) Long- Runtime Operations:-

→ Services are ideal for operations that take a long time to complete such as downloading large file 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:

- service 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' and 'onStart' command 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 life cycle methods [onCreate, onStart, onBind, onDestroy] and override them as needed.

- service can run in these three modes:

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 'startForeground()' 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 receiver or interface to enable communication betⁿ services and other app components.

8). Clean-up 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 interactions and restarts.

610, 20