MEHSANA DISTRICT EDUCATION FOUNDATION SANCHALIT

U. V. Patel College of Engineering

GANPAT UNIVERSITY, KHERVA - 382 711 DIST. MEHSANA. (N.G.)

Name : VIKAS . LOHAR

CLASS : CEIT - B

BATCH : 5B-2

ENROLLMENT : 210120111158

SUBJECT : 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 impacts Android app developers and businesses in the mobile app industry.

Ans: Artificial Intelligence and Machine learning is widely getting in trend which has influenced the Android platform.

As AI and Machine learning (ML) are changing the way Android apps interact with users and handle data. Apps driven by AI may monitor user behaviour, preferences, and trends to provide tailored suggestions and experiences.
Natural language processing and interpretation are getting increasingly advanced in voice assistants and chatbots.
AI and ML are being integrated by developers to improve app functionality, automate chores, and give predictive analytics, resulting in more intelligent and intuitive app.

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?

Ans: Inflater in Android means to read the XML file which describes a layout and to create the some objects of it. The purpose of an inflater for layouts can be understood in the context of dynamically creating user interface and incorporating them into your Android Application.

- Here, how inflator fits into architecture of Android:-

1) XML Layouts: In Android, UI layouts are typically defined using XML files. These XML layout files describe the arrangements and attributes of UI (Text views, buttons, Image views) within an activity or fragment. It is readable and convienient 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 Inflator: This is where the layout inflator 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 inflator forms the structure of the UI for an Activity or fragment. It includes UI Elements defined in the XML layout file, as organized in a tree-like structure.

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

```
val Inflator = layoutInflater
val rootview = inflator.inflate(R.layout.activity_main, null).
setContentView(rootView)
```

Q.3. Explain Concept of Custom dialouge box in Android Application provide Examples to illustrate its use.

→ A custom dialogBox in android is user interface Element that allows developers to create and display a customized popup dialog on the screen. Unlike the standard dialog 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 user to a different screen.

3.) User Interaction :

→ Custom Dialog boxes can contain buttons, textfields, check boxes or any other UI Elements 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 ip input, a customer 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 ways to input this information.

* Code

```
import android.app.AlertDialog
import android.appContent.DialogInterface
import android.OS.Burdle
import androidx.appcompat.app.AppCompatActivity

class your : MainActivity : AppCompatActivity() {

    override fun onCreate(SavedInstanceState : Bundles)
    {
      syper.onCreate(savedInstanceState)
      set ContentView(R.layout.activity_main)

      val builder = AlertDialogBuilder(this)
      builder.setTime("Custom Dialog box Example")
      builder.set messages("This is a Custom dialog")
      builder.set positionButton("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 files are Essential Components in the android app architecture each with distinct role that contribute to the funtionality 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 ~~more~~ note list, note editing and settings, can be implemented as per activities.

2.) Services :

→ Role : Services 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 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 bet" activities using buttons or gestures.

2. Services: While the user is using the app, a service runs in background to periodically save the user's notes to cloud storage
   → This service doesn't have a UI but ~~appears~~ operates independtly 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.
   → your 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 = "https://schemas.
      android.com/apk/res".
    package = "com.Example.mynoteapp">
<application>

    <activity android:name = ".mainActivity>
    <intent-filter>

    </action android:name = "." mainActivity "/>/
    < intent-filter>

    <action android:name = " android:intent.
                              action.MAIN" />

    <category android:name: " android.intent.
                              category.LAUNCHER"/>
```

Q.5. How does the Android manifest file impact the development of an android application provider an Example to demonstrate its significance

→ the android manifest file ~~manifest~~ impacts app development by

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

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

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

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

3. Intent filters: Defining how the app responds to External actions or requests.

Ex: Registering to open PDF files when trapped.

```
<activity android:name = ".PdfViewerActivity" >
<intent-filter>
<action android:name = "android.intent.action.VIEW" />
<category android:name = "android.intent.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-structure applications provide example to clarify your points.

→ Resources in Android development are essential components that helps you create well-structured and flexible app. They serves several purpose, 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 : There define the structure and appearence of your app's UI. They help keep the UI seperate from code logic. Making it easier to maintain 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 "ic_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 ("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 schemas 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 (App theme) to define fonts, colors and other visuals 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 resources file, making it simple to reuse and apply animations in XML to UI.

Example: 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.

→ 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 functions like music playback, location tracking or data sending can continue without disputing 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 ~~noti~~ notification to keep the user aware of ongoing tasks like navigation or chat application.

4.) Inter-Component Communication :

→ Service can communicate with other app components activity, fragment, through interfaces, allowing data Exchange and coordination.

* Developing an android service :

1.) Create a service class :

→ Extend the 'Service' class or one fits subclasses like 'Intent Service' or 'Job Service'.

→ Implement the service's functionality within the "on create" and "on Start" 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 services life cycle methods [onCreate, onStart, 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 requirement

4.) Start and Stop the Service :-

→ Start a service using 'start service (Intent) or bind to it using 'bind service (Intent, service connection, int)

→ Stop a service when it's no longer needed using 'Stop service (Intent)' or 'Stop self ()'.

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 Asynchtask to prevent blocking the main UI thread.

7.) Communications :-

→ Use Intent extras, broadcast receiver or interface to enable Communication bet^n 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 to uncessary battery drain.

9.) Testing :

→ Throughly test your service to ensure it works as expected, including scenarios, like app background task interactions and constants

6-10-23