2201201204l 5CEIT-B MAD chauhan Parth Dineshkumar.

INDEX

Sr. No.	Title	Page From	Page To	Date of Start	Date of Completion	Initials of Start	Remarks
1.	Assignment - 1	1	7	27/9/23	29/9/23		

V.R. Telecom

2CEIT5PE5 : Mobile Application Development

Assignment - 1

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 & businesses in the mobile app industry.

Ans: As per my knowledge, one eye catching trend in the mobile app industry is that has influencing the android platform was the rise of progressive web apps called PWAS. PWAS are web applications that often app-like experiences directly through web browsers.

→ Impact on android app developers:

(1) Cross-platform compatibility - PWAS are designed to work seamlessly accross various platforms and devices, including android. Developers had to consider creating PWAS alongside traditional android apps to ensure broad accessibility.

(2) Enhanced User Experience - PWAS aimed to provide a smoother and more engaging user experience, which set higher expectations for android app developers. This encouraged them to focus on improving the quality and performance of their apps to complete effectivity

(3) Progressive Enhancement - Developers needed to adopt Progressive enhancement strategies

V.R. Telecom & Xerox

to ensure that android apps remained competitive by offering progressive and responsive user experiences, similar to PWAS.

-> Impact on business in the mobile app industry:

(1) Cost savings :- Business could potentially save on development costs by investing in a single PWA that works accross multiple platforms, including Android, rather than building seperate native apps.

(2) Increased Reach :- PWAS enabled business to reach a wider audience, including users with android devices, without relying solely on app store distribution. This broader reach could lead to increased user acquisition.

(3) Improved Engagement :- The focus on delivering app-like experiences through PWAS encouraged business to prioritize user engagement and retention, ultimately benefiting their mobile strategy.

(4) Competition and Innovation :- The rise of PWAS introduced competition, driving businesses to innovate their android apps to keep up with evolving user-expectations and technology trends.

Que: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: In android development, an 'Inflator' is a crucial component that is used to instantiate or 'inflate' the layout XML files into their corresponding view objects. The process converts the XML code, which defines the structure and attributes of UI elements, into actual objects that can be manipulated programmatically.

- Let's see how it's fits into the architecture of Android layouts:

(1) Layout XML files :- In Android, UI elements are defined in XML files within the 'res/layout' directory of an Android project. These XML files contain the

structure of the user interface, specifying things like buttons, text fields, images, etc., as well as their attributes.

(2) Inflatting Layouts :- when an Android app runs the layout XML files need to be converted into actual view objects that can be displayed on the screen. This is where the 'Inflator' comes in. It's used to read the XML files and create the corresponding view objects.

(3) Inflating with Context : The 'Inflater' requires a 'Context' object to perform the inflation process. The 'context' provides essential information about the current state of the application. This context is typically provides by an Activity or Fragment.

(4) Attaching to Parent :- The 'Inflater' also has the option to attach the inflated layout to parent view, if specified. This allows the newly created view to be automatically added to the specified container within the parentview.

(5) Returning a View :- Once the inflation process is complete, the 'Inflator' returns the root vier of the inflatted layout, which can then be used within the application.

(6) Manipulatting views programmatically :- After inflation, developers can programmatically interact

V.R. Telecom & Xerox

with the views, such as setting text, applying styles, adding event listeners, and more.

(7) Displaying in the UI :- Finally, the manipulated views can be added to the user interface using methods like 'setContentView() - (for activities)' or by adding them to the view hierarchy through their parent ViewGroup.

Que:3 Explain the concept of a CustomDialogBox in Android applications. Provide examples to illustrate its use.

Ans: In Android applications, a customDialogBox is a pop-up window that overlays the current activity and is often used to interact with the user, gather input or display information.

→ Overview of a CustomDialogBox :

Purpose :- Custom-dialogs are used when you want to present information, receive user input or perform actions within a self-contained, isolated UI element that temporarily interrupts.

Components :- A custom dialog typically consists of various UI elements like buttons, textviews, images or input fields, tailored to the specific interaction you want to facilitate.

Customization :- Developers can be design the dialog's apperance, layout and behavior according to their app's branding or specific requirements. This customization allows for creativity in design and functionality.

→ Let's see an example of creating and using a custom dialog box in Android.

```
fun showCustomDialog () {
    val customDialog = Dialog (this)
    customDialog.setContentView ( R.layout.custom-
                                    dialog_layout)

    val messageTextView = CustomDialog.findViewBy
                                Id <TextView> (R.id.msg-
                                    TextView)

    val okBtn = CustomDialog.findViewById <Button>
                                (R.id.okBtn)

    messageTextView.text = " Custom Dialog Box "
    okBtn.setOnclickListener {
                    customDialog.dismiss()
                    }
    customDialog.show()
}
```

→ Use cases of Custom Dialog Box :-
- login
- confirmationDialog
- settings
- Information Pop-up
- MediaPlayback controls

Que: 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 of how they cooperate to design a mobile app?

Ans: Roles:

V.R. Telecom & Xerox

(1) Activities :- Activities represent individual screens or UI components in an Android app. They manage the user interface and user interactions.

(2) Services :- Services are background components that perform long running operations or handle tasks that don't require a user interface. They can run even if the app's UI not visible.

(3) Android Manifest File :- the Android Manifest file is like the app's blueprint. It declares the app's components and defines how they interact with the Android system and other components.

Exa:- In Android Manifest.xml, you specify which activities are part of your app, their launch modes, permissions and service declarations. This file acts as a blueprint for android system to understand your App's structure and behaviour.

```
class MainActivity : AppCompatActivity () {
    overrid fun onCreate ( savedInstanceState : Bundle ? )
    {
        super.onCreate ( savedInstanceState )
        setContentView ( R. layout. activity_main )
        startserviceButton . setonClickListner {
            val serviceIntent = Intent ( this , NotificationService ::
                                                class. java )
            startService ( serviceIntent )
        }
    }
}

class NotificationService : IntentService ("Notification Service") {
    override fun onHandleIntent (intent : Intent ) {
        if (intent ! = null) {
```

```
        }
    }

    Private fun createNotification() {
        val channelID = "new_channel"
        if (Build.version.SDK_Int >= Build.version_codes.O)
        {
            val name = "new_channel"
            val notificationManager.createNotification_
                channel(channel)
        }

        val Builder = Notification.compat.Builder(this,
                                channelID).
            setSmallIcon(R.drawable.ic_launcher_
                                foreground).
            setContentText("This is notification from
                                service")
        }
    }
```

Que: 5 How does the Andrard Manifest file impact the development of an Android application? Provide an example to demonstrate its significance.

Ans: The Android Manifest file is a crucial component in the development of an Android application. It serves several important purposes and its content significantly impacts how the andrard system interacts with and manages your APP. significance of the Android Manifest file:

V.R. Telecom & Xerox

- App Configuration
- Component Declaration
- Permission
- Intent filters
- App lifecycles

Exa:-

```
<? xml version = "1.0" encoding = "utf_8" ?>
<manifest xmlns:android = "http://schemas.android.com/apk
    /res/android"
    xmlns: tools = "http://schemas.android.com/tools" >
    < uses-permission android: name = "android. permission.
                                        INTERNET/>

    < application
        android : allowBackup = "true"
        android : dataExtractionRules = "@xml /data_extraction_
                                                rules"

        android : fullBackupContent = "@xml / backup_rules"
        android : icon = "@mipmap / ic_launcher"
        android : label = "Assignment"
        android : roundIcon = "@mipmap /ic_launcher_round"
        android : supportsRtl = "true"
        android : theme = "@style /theme. Assignment"
        tools : targetApi = "31" >
        < activity
            android : name = ".MainActivity"
            android : exported = "true" >
            < intent-filter >
                <action android :name = "android. intent. action. MAIN" />
                <category android : name = "android. intent. category.
                                        LAUNCHER" />
            </ intent-filter >
        </activity>
        < activity android :name = ".SecondActivity" >
            ...
        < /activity >
    < /application >
</manifest>
```

Que:6 What is the role of resources in Android development? Discuss the various types of resources and their significance in creating well-structured application. Provide examples to clarify your points.

Ans: Resources play a fundamental role in Android development by providing a structured way to manage assets, values, layouts and other elements used in your 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 file in the 'res/layout' directory.
Significance :- Define the structure and apperance of the app's user interface
Example :- 'activity_main.xml' defines the layout of your MainActivity, specifying UI components like buttons, textviews and their arrangement.

```
<Button
    android:id="@+id/myBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click Me" />
```

(2) Drawable Resources:

type :- Images and drawable assets in the 'res/drawable' directory.

V.R. Telecom & Xerox

Significance :- Store graphics, icons and images used in your app.
Example :- 'ic_launcher.png' is the app's launcher icon.

(3) Mipmap Resources:
type :- Storing images at multiple resolutions or sizes in the 'res/layout' directory.
Significance :- Used to provide multiple versions of an image at different resolutions. It ensures that images look crisp and clear on screens with varying pixel densities.
Example :- ic_launcher → mipmap-hdpi, mipmap-mdpi, mipmap-xhdpi, mipmap-xxhdpi, mipmap-xxxhdpi

(4) String Resources :-
type :- Strings defined in xml files under 'res/values'.
Significance :- Store text strings, making it easier to provide translations and maintain consistency.
Example :- 'res/values/strings.xml' contains string resources.

```
<string name="app_name">My App</string>
<string name="welcome_message">Welcome to my App
</string>
```

(5) Color Resources :-
type :- colors defined in xml files under 'res/values'.
Significance :- Store color values, ensuring consistency in the app's design.
Example :- 'res/values/colors.xml' defines color resources.

```
<color name="primary_color">#007ACC</color>
```

(6) Style Resources:-
type :- styles defined in xml files under 'res/values'
Significance :- Define reusable styles for UI components.
Example :- 'res/values/style.xml" define style.

```
<style name="MyButtonStyle">
    <item name="android:background">@drawable/my-button
    </item>
```

</style>

(7) Dimension Resources :-
type :- Dimensions defined in xml files under 'res/values'
Significance :- Store dimension values, ensuring a consistent Layout.
Example :- 'res/values/dimens.xml' defines dimension resources.
<dimen name="margin_large">16dp</dimen>

(8) Raw Resources :-
type :- files stored in the 'res/raw' directory.
significance :- Store non-xml files, such as JSON data, audio, video.
Example :- store a JSON file for app configuration.

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

Ans: 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 time to complete, such as playing music

(3) Inter-component Communication :- Services enable

V.R. Telecom & Xerox

components like activities, broadcast receivers and other services to communicate with each other efficiently.

(4) Foreground 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 interaction, like music playback. ~~Process of~~

→ Process of Developing an Android Service :-

(1) Define the Service class :- Create a new Java or kotlin class that extends the 'Service' class that override methods like onCreate(), onDestroy(), OnStartCommand() to define the behaviour of your service.

(2) Configure Service in Manifest :- Declare your service in the AndroidManifest.xml file to inform the Android system abot its existance and configuration.

<Service android:name = ".MyService" />

(3) Start or Bind the service :- Decide whether you want to start your service or bind it to other components. use startService() or bindService().

(4) Implement Service Logic :- In Service class, implement the specific logic your service needs to perform its task.

(5) Handle lifecycle :- Release resources when thy are no longer needed and consider using 'stopSelf()' or 'stopService()'.

(6) Interact with other Components :- Use appropriate mechanism like intents, broadcasts or callbacks to facilitate communication.

(7) Foreground Service (optional) :- If your service needs to run in the forground, 'startForeground()'.

(8) Testing :- Throughly test your service to ensure its functions as expected, including handling various scenarios like network failures.

(9) Optimization :- Optimization your service for performance and resource effeciently to minimize battery usage.

(10) Error Handling and logging :- Implement proper error handling and logging mechanisms to diagnose and address issues that may occur while service is running

18/10/23