Assignment - 1 (MAD)

NAME :- Patel Parth H.

Class :- CEIT-B

Batch :- B1

EN. No :- 210120110095

Assignment - 1 (MAD)

Q.1 Based on your understanding Identify a recent business that has butterhead the Android Platform Explain how this trend impacts Android developers and business in the mobile app industry.

⇒ One significant trend in the Android app industry was the increasing emphasis on user privacy and data security.

→ Impact on Android App Developer

1. Enhanced Permissions and consent
:- Developer had to be more borespaght about the data their app called and request explicit use consent. This meant redesigning permission dialogs and ensuring that use understood why creation data was being collected.

2. Limitations on Advertising.
:- For App relying on advertising revenue, changed in tracking and targeting due to privacy changes affected their monetization strategies. Developers need to adapt to these change.

3. Data Handling and Storage.
:- Developers had to reviews. how they handled and stord users data, implementing data protection measured This could lead to increased development time & cost.

→ Impacts on Businesses

1. Compliance Cost:-
→ Businesses operating in the android app industry require for compliance with strict data privacy regulation. This could include legal and technical measures to ensure data protection.

2. Monetization challenges
→ Businesses relying heavily on user data for advertising and personalized content faced challenges in maintaining their revenue streams. They needed to find new ways to engage users and generate income.

Q.2 What is the purpose of an Inflater of layout in Android development and how does it fit in to the architecture of Android layouts?
→ In Android development, an "Inflater" refers to the layout Inflater, which plays a crucial role in creating a user interface from xml layout files. It's primary purpose is to take an xml layout resource and convert it into a corresponding View object in memory. Here's how the layout Inflater fit in to the architecture of Android layouts:-

1) xml layout files :- In Android, UI components are often defined using xml layout files. These files describe the structure and appearance of the UI, specifying things like widgets, their properties their placement within the UI.

2. Layout Inflation :- When your Android app runs, it needs to turn these XML layout files into actual view objects that can be displayed on the screen. This process is known as Layout Inflation".

3. Layout Inflater :- It is responsible for reading the XML layout files and instantiating the corresponding View object in memory. It takes the XML file as input, parses it, and creates the view objects, such as TextViews, Buttons, etc.

4. Dynamic UI creation :- Layout inflation is particularly valuable when you need to create UI elements dynamically for example in response to user input actions or when working with RecyclerViews to display lists of items.

5. Displaying UI :- After inflation and customization the view objects can be added to the app's layout hierarchy and displayed on the screen.

Q.3 Explain the concept of a custom Dialog Box in Android applications. Provide examples to illustrate it's use.

→ In Android applications, a custom Dialog Box 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 custom Dialog Box :-

- 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, textview, images or input fields, tailored to the specific interaction you want to facilitate.
- Customization :- Developers can design the dialog's appearance, layout, and behavior according to their app's branding or specific requirement. This customization allows for creativity in design and functionality.

→ simple example of creating and using a custom dialog in Android

```
fun showCustomDialog() {
    val customDialog = Dialog(this)
    customDialog.setContentView(R.layout.custom_dialog)
    val messageTextView = customDialog.findViewById<TextView>(R.id.messageTextView)
    val okButton = customDialog.findViewById<Button>(R.id.okButton)
    messageTextView.text = "This is a custom dialog!"
    okButton.setOnClickListener {
        customDialog.dismiss()
    }
    customDialog.show()
}
```

→ use cases of custom dialog box : login, information Dialog
- settings, Informational Pop-up, Media Playback controls.

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 of how they contribute to design a mobile app?

→ 1. Activities

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

2. Services

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

3. Android Manifest file:-

:- Role - The AndroidManifest.xml 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.

Example

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

```
set startService Button.setOnClickListener {
Val serviceIntent = Intent(this, NotificationService
        ::class.java)
startService(serviceIntent) }
  }
}
```

→ class NotificationService: IntentService("Notification
service") {
override fun onHandleIntent(Intent: Intent?) {
if (Intent != null) {
createNotification()
}
}
private fun createNotification() {
Val channelID = "my_channel"
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
{ Val name = "my_channel"
Val notificationManager = getSystemService
(NotificationManager::class.java)
notificationManager.createNotificationChannel
}
Val builder = NotificationCompat.Builder(this, channelID)
.setSmallIcon(R.drawable.ic_launcher_foreground)
.setContentText("This is notification from service.")
}
}

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

→ The Android Manifest file is a crucial component in the development of an Android application. It serves several important purposes, and its correct significantly impacts how the android system interacts with and manages your app. Significance of the Android Manifest file:-

- App configuration
- Component Declaration
- Intent filters
- App lifecycle.
- Permissions

Ex <Manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.myapp">
<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>

```
<activity android:name=".SecondActivity">
    .... Declare additional activitys here....
</activity>
<uses-Permission android:name="android.Permission.
    .... Declare required Permissions here....
</application>
</manifest>
```

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

→ 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 type of resources and their significance with examples:-

1. Layout Resources:-
- type: xml files in the 'res layout' directory.
- significance: Define the structure and appearance of the app's user interface.
- Ex:- 'activity_main.xml' defines the layout of your main activity, specifying UI components like buttons, text view and their arrangement.

```
<Button
    android:id="@+id/my_button"
    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.
- Significance: Store graphics, icons, and images used in your app.
- Ex :- 'ic_launcher.png' is the app's launcher icon.

3. String Resources :-
- type :- Strings defined in xml files under 'res/values'.
- Significance : Store text string, making it easier to provide translation.
- Ex :- 'res/values/strings.xml' contains string resources.

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

4. Color Resources :-
- type : Colors defined in xml files under res/values
- Significance: Store color values, ensuring consistency in the app's design.

Ex :- 'res/value/color.xml' defines color resources.

```
<color name="primary_color">#007ACC</color>
<color name="accent_color">#FFA500</color>
```

5. Style Resources:-
- Type: Styles defined in xml files under 'res/values'
- Significance: Store dimension values, ensuring a consistent layout.
- Ex:- 'res/values/styles.xml' defines style.
<Style name: 'res/value/styles:xml' defines style
<Style name="my button style">
<item name='android:background "> @drawable/my_button
</item>
<item name="android:textcolor"> @color/primary_color
</item>
</Style>

6. Raw Resources:-
- Type: Files stored in the 'res/raw' directory.
- Significance: Store non-xml file, such as Json data, and audio.
- Ex: Store a Json file for app configuration.

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

→ 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 components like activities, broadcast receivers and other services to communicate with eachother efficiently.
4. Foreground services:-
- Android services can run in the background even when the app isn't in the foreground. This is useful for features that require ongoing user interaction, like music playback.

→ Process of developing an Android services.
1. Define the service class
- Create a new Java or Kotlin class that extends the 'Services' class.
2. Configure service in Manifest
- Declare your service in the Android Manifest.xml file to inform the Android system about its existence and configuration.
3. Start or bind the service
- Decide whether you want to start services or bind it to other components.
4. Implement service logic
- In service class, implement the specific logic that service needs to perform its task.

5. Handle lifecycle:
- Release resources when they are no longer need and consider using 'stopself()' or 'stopServices()'
6. Interact with other components:
- Use appropriate mechanisms like intents, broadcast or callback to facilitate communication.
7. Foreground services (optional):
- If your service needs to run in the foreground. 'startForeground()'.
8. Testing:
- Thoroughly test your service to ensure it function as expected, including handling various scenarios like network.
9. Optimization:
- Optimization your service for performance and resource efficiency 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.

5/10/23