Class:- CEIT-B
BAtch:- 5B-4

En. No:- 21012011082
Name:- Patel Hir.

Assignment-1

1) Based on your understanding, identify a recent business trend that has influenced the Andriod platform. Explain how this trend impact Andriod app developers and businesses in the mobile app industry.

→ ~~An~~ Progressive web apps are web app that offer a native app-like experience but are built using web technologies.
- Here's how this trend could impact Andriod app developers and businesses in the mobile app industry:

1) Cross-platform compatibility:
- PWAs can run on any pattern with a modern web browser, including Andriod.

2) Improved performance:
- PWAs are designed to be fast and efficient, providing a smoother user experience.

3) Reduced App store Dependency:
- This can be beneficial for businesses looking to lunch quickly or avoid app store fees.

4) offline functionlity:
- PWAs can work offline, caching content and data. This capability challenges Andriod app ~~dve~~ developers to implement similar offline features in native apps to remain competitive.

5) Search Engine visibility:-
→ PWAs are easily discoverable by search engines, Potentially boosting a business's online Presence.

6) cost-effective maintenance:-
→ Maintaining a single PWA codebase is often more cost-effective than managing separate codebases for Android and other platforms.

7) User Experience:
→ PWAs aim to Provide seamless user experience, including Push notifications, which were Previously a native app feature.

→ It's essential to note that while PWAs offer several advantages, they may not be suitable for all types of applications. The choice between developing a native Android app or PWA depends on the specific requirements of the Project, target audience and business

2) what is the purpose of an Inflator of layout in Andriod development, and how does it fit into the architecture of Andriod layouts?

→ A layout Inflater is a crucial component used to create view objects from XML layout resource files at runtime.

• Purpose of layout Inflater:-
1) Dynamic UI Generation :-
→ Andriod applications often require dynamic user interfaces that can change or adapt based on user interactions or other factors.

2) Separation of concerns:-
→ Andriod encourages a separation of ~~ten~~ concerns between UI design and application logic layout Inflater facilitates this separation by allowing designers to work on UI layouts in XML files. while developers can focus on the code logic.

• How it fits into Andriod layout Architecture:
1) layout xml files:-
- Andriod layouts are typically defined in xml files that specify the structure and appearance of the user interface elements.
2) Resource IDs:- During the build process, Andriod generates a unique resource ID for each xml layout file making it accessible through the class.

3) Activities or fragments:-
→ within an Andriod activity or fragment, you can use the layout Inflater to 'inflate' a layout xml file into a view hierarchy.

4) Event Handling and logic:-
→ with the view objects in place, you can implement event handling and application logic to respond to user interactions or data changes.

Q-3 Explain the concept of custom Dialog Box in Andriod applications. Provide examples to illustrate its use.

→ concept of a custom Dialog Box:-
1) custom layout
→ Developers design a custom xml layout file that defines the appearance and content of the Dialog.
2) Dialog instance:-
→ In code, a custom dialog instance is created and associated with the custom layout.
3) User interaction:-
→ Event listeners are attached to the UI elements within the custom dialog, allowing developers to respond to user interaction, such as button clicks or text input.

example:-

```
    Dialog customDialog = new Dialog (context);
customDialog. setContentView (R. layout. custom_dialog_layout);
customDialog. requestWindowFeature (Window. Feature_NO_Title);

TextView dialogText = customDialog. findViewById (R.id. dialog_text);
Button closeButton = customDialog. findViewById (R.id. close_button);

dialogText. setText (" This is a custom dialog!");
closeButton. setOnClickListener (new View. OnClickListener () {
        @ Override
            Public void onClick (View v) {
                customDialog. dismiss ();
            }
        });
customDialog. show ();
```

Q-4 How do activies, services, and the Andriod Manifest file work together to make an Andriod app? can you describe their main roles and provide a basic example of how they cooperate to design a mobile app?

→ 1) activities:-
- Role: activities represent individual screens or UI components in an Andriod 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 user interface.

3) Andriod Manifest file:
- Role: The AndriodManifest.XML file is like that app's blueprint. It declares the app's components. and defines how they interact with the Andriod system and other components

example:-

```
class MainActivity: AppCompatActivity () {
  override fun onCreate (saved InstanceState : Bundle?) {
    super.onCreate (saved InstanceState)
    set contentview (R. layout. activity_main )
    startServiceButton . setonClick Listener {

    val service Intent = Intent (this, NotificationService:: class.java
    start service (service Intent)  } } }

- class NotificationService : IntentService ("Notification service") {
  override fun onHandle Intent (intent : Intent?) {
    if (intent != null) {
               createNotification ()  } }
  Private  fun  create Notification () {
     val channelID = "my_channel"
       if (Build.version. SDK_Int >= Build.version_codes. O) {
       val name = "my channel"
      val notification manager = getSystemService (Notification
                                                    manager:: class
   Notification manager. create   Notification channel (channel)
       }
val builder = NotificationCompat.Builder(this, channelID)
    .setSmallIcon(R.drawable.ic_launcher_foreground)
    .setContentText("This is notification from service.")
    }
    }
```

Q-5 How does the Andriod manifest file impact the development of an Andriod application? provide an example to demonstrate its significance.

→ The Andriod manifest file is a crucial component int the development of an Andriod application.

- App configuration
- component Declaration.
- Intent filters
- App Lifecycle.

example:-

```
<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"/>
```

```
< /intent.filter>
  </activity>
< activity  android : name = ". SecondActivity" >
    - - . Declare  additional  activities  here - -

  </activity>
  < user-permission  android : name = "android . permission .
                                              Intent" />

   </application>
   </manifest>
```

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

→ The various types 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:-

```
<Button
    android: id = "@+id/my Button"
    andriod: layout_ width = "wrap_content"
    andriod : layout _ height = "wrap_content"
    andriod : text = "click me" />
```

2) Drawable Resources:
- type: Images and drawable assets in the 'res/drawable' directory.
- significance: Store graphics, icon and images used in your app.
ex:- 'ic_launcher.png' is the app's launcher icon.

3) String Resources:
type: String defined in xml files under 'res/values'.
significance: store text strings, making it easier to provide translations and maintain consistency.
ex:-

```
<string name="app_name">myAPP</string>
<string name="welcome_message">welcome to myapp
                              </string>
```

4) 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>
<color name="accent_color">#FFA500</color>
```

5) style Resources:
type: style defined in xml files under 'res/values'.
significance: Define reusable styles for UI components

examples:-

```
<style name = "my Buttonstyle">
 <item name= "andriod: background"> @drawable/my-button </item>
<item name: "andriod: textcolor"> @ color/primary - color </item>
 </style>
```

6) Row Resources:
- type: files stored in the 'res/raw' directory.
- significance: store non-xml files, such as JSON data, audio
ex: store a Json file for app configuration.

Q-7 How does an Andriod service contribute to the functionality of a mobile applications Describe the process of developing an Andriod service.

→ contributions of Andriod services:

1) Background Processing:
→ services allow apps to perform tasks in the background without blocking the user interface.

2) Long - running operations:
→ services are for handling operations that require more time to complete.

3) Inter - component communication:
→ services enable components like activites, broadcast receivers and other services to communicate with eachother efficiently.

4) Define the service class:
→ create a new Java or Kotlin class that extends the 'service' class.

~~5)~~ Process of Developing an Andriod Service:

1) Define the service class:
- Create a new Java or Kotlin class that extends the 'service' class.

2) Configure services in manifest:
- Declare your service in the Andriod manifest.xml file to inform the Andriod system about its existence 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 component.

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

5) Handle Lifecycle:
- Release resources when they're no longer needed and considere using 'stopself()' or 'stopservice()'.

6) Foreground services (optional):
- If your service needs to run in the foreground, 'startforeground ()'

7) Optimization:
→ optimize your service for performance and resource efficiency to minimize battery usage.

8) Error Handling and logging:
→ Implement proper error handling and mechanisms to diagnose and address issues that may occur while service is running.

25/10/23