2101202101 7_ CE IT-A (A-3)

5/10/23

MAD Assignment - 1

Ans-1 → One significant trend that was shaping the android platform & mobile app industry was the growth of Progressive web apps (PWAs) & their impact on native android app development.

* PWAs

- PWAs are web applications that offer a native app like experience within a web browser. They leverage modern web technologies to provide features such as offline, access, push notifications & a responsive design.

* Impact on android app Developers:

(i) Reduced development effort: Developers could choose to build PWAs instead of traditional native android apps, reducing the need for separate codebases & resources for different platforms. This could save time & effort.

(ii) Declare the service in the manifest: In the android manifest XML file declare the service with its name & intent filter.

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

(iii) Implement the service's functionality: Write the logic for the service inside the overridden methods.

(iv) Start the service: To start the service, create an intent that specifies the service's class & call 'startService()' with the intent.

(v) Optionally, bind the service.

(vi) Stop or unbind from the service: You can stop a service using 'stopService()' or unbind it from using 'unbindService()'.

Ans-2 LayoutInflater is used to create a new view object from one of your XML layouts. FindViewById just gives you a reference to a view that has already been created.

(i) Dynamic UI creation: Android apps often need to create & display UI components dynamically based on user interactions, data or runtime conditions.

(ii) Separation of Concerns: Separating the UI layout definition from the code that manages it helps maintain a clear separation of concerns.

* How it works:

(i) Layout definition: Developers define the layout for the custom dialog using an XML layout file.

(ii) Dialog creation: A custom dialog is created & associated with the XML layout. Developers can set properties for the dialog, such as its size, style & specific behaviours.

(iii) Displaying the dialog: When needed the custom dialog can be displayed to the user.

(iv) User Interaction: Users can interact with the elements within the custom dialog, entering text, clicking buttons.

(v) Handling Actions: Developers can define actions or event handlers for the UI elements within the custom dialog to respond to user input or perform specific tasks.

* Examples:

(i) Login Display:

- Purpose: A custom dialog for user login or authentication.
- Layout: The dialog may contain EditTexts for username and password, a "login" button & "forgot password".

(iii) Code Reusability: Developers can reuse layouts

(iv) Localization & Theming.

(v) Efficiency

(vi) Custom Views

Ans-3 In android applications a 'custom Dialog box' often is a UI component that allows developers to create a customized pop-up dialog windows that can display information, collect user input or perform various actions.

=) How it works:

(i) Layout Definition: Developers define the layout for the custom display using an XML file layout.

(ii) Dialog Creation: In code a custom dialog is created & associated with the XML layout. Developers can set properties for the dialog such as its size, style & specific behaviours.

(iii) Displaying the Dialog: When needed, the custom display can be displayed to the user.

(iv) User Interaction: Users can interact with the elements within the custom dialog, entering text, clicking buttons

* Examples

(i) Login Display:

- Purpose: A custom dialog for user login.
- Layout: The dialog may contain edit texts for username & password, & 'login' button & 'forgot password'.
- Use case: Displayed when the user needs to login.

(ii) Confirmation Dialog:

- Purpose: A dialog that asks the user for confirmation before proceeding with a critical action.
- Layout: Contains a message describing the action & 'Yes' & 'No' buttons.
- Use case: Used to prevent accidental actions like deleting data.

(iii) Custom Picker Dialog:

- Purpose: A dialog with a custom picker such as a date picker, time picker or color picker.
- Layout: It includes the picker widget & "Ok" & "Cancel" buttons.

(iv) Information Dialog:

- Purpose: Displaying informative messages or additional details.
- Layout: Contains text or multimedia elements to information.
- Use case: Used for displaying pop up help messages, terms of service.

Ans-4 Activities, Services & the android manifest file work together to create the structure & functionality of an app.

(i) Activities:

- Role: It represent individual screens or user interfaces in an android app.
- Eg: Imagine a simple email app with two activities one for composing emails & another for viewing the inbox.

(ii) Services:

- Role: Services are background components that perform long-running operations without a user interface.
- Eg - In our email app example, a service could be used to periodically check for new emails in the background & update the inbox.

(iii) Role: The android manifest File:

- Role: The AndroidManifest XML file is a configuration file that defines essential info about the app.
- Eg: In this, you specify which activities & services the app contains their properties & any permission req.

Ans-5 Examples

(i) Component declaration: The android manifest file is where you declare all the components of your android application, including activities, services, broadcast receivers & content providers.

=) <application ...>
<activity android:name = ".MainActivity">
<intent-filter>
<action android:name = "android.intent.action.MAIN"/>
<category android:name = "android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<service android:name = ".MyService"/>
<receiver android:name = ".MyReceiver"/>
<provider
android:name = ".MyContentProvider"
android:authorities = "com.example.myapp.provider"/>
</application>

(ii) Application Configuration: This includes setting the app's theme, icon, label, version info. & defining the min android API level req.

=) <application
android:icon = "@drawable/app_icon"
android:label = "@string/app_name"
android:theme = "@style/AppTheme"
android:allowBackup = "true"
android:versionCode = "1"
android:versionName = "1.0">

(iii) Permission: You declare the permissions your app require to access various system resources & data, such as the camera, location, internet.

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

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

(iv) Intent filters.

(v) App permission.

Ans-6 Resources are the additional files & static content that your code uses, such as bitmaps, layout defn, user interface strings, animation instructions & more.

- Resources are used for anything from defining colors, images, layout, menus & string values.

* Resources types overview:

(i) Animation resources: Define pre-determined animations. Frame animations are saved in res/drawable & accessed from the R.drawable class.

(ii) Color state list resources: Define a color resource that changes based on the view state. Saved in res/color/ & access from the R.color class.

(iii) Drawable resources: Define various graphics with bitmaps or XML.

(iv) Layout resources: Defines the layout for your app UI.

(v) Menu resources: Defines the content of app menus.

(vi) String resources

(vii) Style : Defines look & format for UI elements

Ans-7 (i) Declare the service in the manifest: In the android - Manifest.xml file, declare the service with its name & input filter.

⇒ <service android:name=".MyService" />

(ii) Implement the service's functionality:

- Write the logic for the service inside the overridden methods.

(iii) Start the service: To start the service, create an intent that specifies the service's class & call 'startService()'.

(iv) Optionally, bind the service.

(v) Stop or unbind from the service:

- You can stop a service using 'stopService()' or unbind from using 'unbindService()'.