Programme: B.Sc.
Subject: Computer Science
Semester: V
Paper Code: CSC108
Paper Title: Mobile Application Development

Unit III: Activities and UI
Module Name: Implicit Intent– Intent Filters and Intent Resolution Process, Pending intents

Name of the Presenter: Miss Ashweta Anand Fondekar

Implicit Intent in Android

- Implicit Type of Intents are those that do not mention any component name and only declare the actions.
- Implicit Intents specifies the actions that are to be performed. These actions can invoke any application component that can perform these actions. These are useful when your action is important but your application is not capable of performing it.

```
Intent intent = new Intent();
intent.setAction=(Intent.ACTION_DIAL);
```

Intent Filters in Android

- Intent Filters are expressions in the Manifest file of an Android Application. These are responsible for deciding the behavior of the Intent. Intent filters that we mention in the Manifest file can be nested with the application components.
- Intent Filters can define the behavior of Intents using three Elements, that are-
    1. <actions> – Action name defines the intent action that it'll accept.
    2. <data> – Data defines the data that is acceptable.
    3. <category> – Category defines the name of the Intent Category that is acceptable.

Intent Filter in Manifest File

Following is the code snippet of defining an activity with Intent Filter (**<intent-filter>**) in the Android Manifest file (**AndroidManifest.xml**) like as shown below.

```
<activity android:name ".MainActivity">
    <intent-filter>
    <action android:name= "android.intent.action.SEND">
    <category android:name= "android.intent.category.HOME">
    <data android:mimeType= "text/plain">
    </intent-filter>
    </activity>
```

We can define a filter with multiple instances of **<action>**, **<category>** or **<data>** elements and we need to make sure that component can handle all the combinations of filter elements.

Multiple Intent Filters in Manifest.xml:

In case if we want to handle multiple Intents with combinations of action, category, and data, we need to create multiple intent filters.

Following is the code snippet of defining multiple **Intent filters** in the android manifest file to handle multiple Intents.

```
<activity android:name ``.MainActivity">
    <!-- This activity is the main entry, should appear in app launcher -->
    <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
    <data android:mimeType="text/plain"/>
    </intent-filter>
    </activity>
    <activity android:name=".ResultActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
    </intent-filter>
```

</activity>

If you observe above code snippet the activity "**MainActivity**" will act as an entry point for our app because we defined an activity using **MAIN** action and **LAUNCHER** category attributes in intent filters (<intent-filter>).

MAIN- It indicates the app's main entry point that means it starts the activity which defines with the **MAIN** action when the user initially launches the app with a launcher icon.

LAUNCHER- It indicates that this activity icon should be placed on the home screen list of apps. In case if the**<activity>**element doesn't specify an **icon** with icon, then the system uses the icon from the **<application>** element.

These two (**MAIN**, **LAUNCHER**) elements must be paired together in order for the activity to appear in the app launcher.

The second activity "**ResultActivity**" is intended to help us to share the text. The user might enter this activity by navigating from **MainActivity** and they can also enter directly from another app using Implicit Intent which is matching one of the two activity filters.

Intent Resolution

- The intent resolution comes in the role when the system receives an implicit intent.
- The Intent Resolution compares the contents of Intent Object against the Intent Filters. **(Intent Filters are associated with the different Android components, and can receive Intent. Intent filter is a way for Android components to declare their capabilities to the Android system.)**
- Implicit intents do not mention the name of the components. Therefore the system needs to search for the appropriate application component that can perform the actions.
- This searching process takes place in the following three tests:

1. Action Test

   In this, the system checks the Intent action matches with the intents in the intent filter. If it does, it passes this test, otherwise, it fails.

2. Category Test

   In this, the system checks the Intent category matches with the categories in the intent filter. If it does, it passes this test, otherwise, it fails.

3. Data Test

In this, the system checks the Intent MIME data matches with the data in the intent filter. If it does, it passes this test, otherwise, it fails.

```
<intent-filter>

    <action android:name= “android.intent.action.EDIT”> // It is matched with the
    list of actions of Intents.

    <category android:name=“android.intent.category.ALTERNATIVE”>

    // It is matched with the list of categories of Intents.

    <data android:mimeType= “text/plain”>
    // It is matched with the list of MIME data of Intents.
</intent-filter>
```

Now, if these tests are passed, Intents are said to match.This is known as **Intent Matching.** Intent matching is only said to be successful, if all the actions and categories of Intents match against the filters of the Intent Filter class.

Pending Intent in Android

- Pending Intents are like a protective shield over the Intent objects. It covers them and grants permission to the external applications to access them.
- It is used to give permission to foreign application to start at later point of time a component in your application with intent passed in pending intent.
- The main differences between a pendingIntent and regular intent is pendingIntent will perform action at a later time where Normal/Regular intent starts immediately.
- Eg. Your application sends notification, user clicks the notification text, this click event sends intent (from Pendingintent) to start activity in your application.

```
//Creating a regular intent
Intent intent = new Intent(this, SomeActivity.class);
// Creating a pendingIntent and wrapping our intent
PendingIntent pendingIntent = PendingIntent.getActivity(this, 1, intent,
PendingIntent.FLAG_UPDATE_CURRENT)
```

1. **this** (context) : This is the context in which the **PendingIntent** starts the activity

2. **requestCode** : “1” is the private request code for the sender used in the above example. Using it later with the same method again will get back the same pending intent.
3. **intent** : Explicit intent object of the activity to be launched
4. **flag** : One of the PendingIntent flag that we’ve used in the example is **FLAG_UPDATE_CURRENT**. This one states that if a previous PendingIntent already exists, then the current one will update it with the latest intent. There are many other flags like **FLAG_CANCEL_CURRENT** etc.