UNIT V: MENUS, SMS &LOCATION-BASED SERVICES

51 Localization
5.2 Creating the Helper Methods, Options menu and Context menu
5.3 Dialogs- Alert dialog
5.4 SMS Messaging Using a Content Provider
5.5 Lists view
5.6 Displaying Maps, Getting Location Data
5.7 Monitoring a Location using GPS

5.1 Localization

An android application can run on many devices in many different regions. In order to make your application more interactive, your application should handle text, numbers, files etc. in ways appropriate to the locales where your application will be used.

The way of changing string into different languages is called as localization

Localizing Strings

In order to localize the strings used in your application, make a new folder under **res** with name of **values-local** where local would be the replaced with the region.

For example, in the case of Marathi, the **values-mr** folder would be made under res

Once that folder is made, copy the **strings.xml** from default folder to the folder you have created. And change its contents as in Marathi.

Create a folder values-hi to store the custom messages

Create a folder values-(Code of Language) by following the following steps. The -hi extension defines that if the device's preference is set to Hindi(hi), the context within the application would be set according to the data present in the values-hi folder.

Click on Android and select the Project option:

Now expand the folder until you find the res (Resources) folder, right-click on it, select new, and click Android Resource Directory.

Set the directory name as values-hi

The values-hi folder is now created

Create a strings.xml file

Create a strings.xml file in this folder, that shall contain a custom message. Entities of this file should match the entities of the default strings.xml file.

Now add a Values Resource File in the values-hi folder.

Give it a name, strings, it creates a .XML file

strings.xml file is created in the values-hi folder

Go back to the Android view and check if the newly created file is present.

The file is available under the values/strings folder

Add the custom message values to string.xml (regular) and string.xml (hi)

Add a custom message to the pre-existing as well as the newly created strings.xml file. The entities of both the file must be the same, context may differ

English(en) strings.xml

```
<resources>
   <string name="app_name">GFG | LanguageLocalization</string>
   <string name="custom_message">This application tests if the language localization works on the device</string>
</resources>
```

Hindi(hi)strings.xml

```
<resources>
  <!--custom_message in the desired language-->
  <!--App name also changes-->
  <string name="app_name">GFG | भाषा स्थानीयकरण</string>
  <string name="custom_message">यह एप्लिकेशन परीक्षण करता है कि भाषा स्थानीयकरण डिवाइस पर काम करती है या नहीं।</string>
</resources>
```

5.2 Creating the Helper Methods Options menu

Options Menu is a primary collection of menu items for an activity and it is useful to implement actions that have a global impact on the app, such as Settings, Search, etc.

By using Options Menu, we can combine multiple actions and other options that are relevant to our current activity. We can define items for the options menu from either our Activity or Fragment class.

Create Android Options Menu in XML File

To define **options menu**, we need to create a new folder **menu** inside of our project resource directory (**res/menu/**) and add a new XML (**menu_example**) file to build the menu.

```
<? xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

 <item android:id="@+id/mail"
      android:icon="@drawable/ic_mail"
      android:title="@string/mail" />

</menu>
```

Load Android Options Menu from an Activity

To specify the options menu for an activity, we need to override **onCreateOptionsMenu()** method and load the defined menu resource using **MenuInflater.inflate()**

```
@Override
public void onCreateOptionsMenu(ContextMenu menu, View v,
ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.menu_example, menu);
```

```
}
```

Handle Android Options Menu Click Events

We can handle options menu item click events using the **onOptionsItemSelected()** event method.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.mail:
         // do something
         return true;

      default:
         return super.onContextItemSelected(item);
   }
}
```

5.3 Creating the Helper Methods Context menu

Context Menu is like a floating menu and that appears when the user performs a long press or click on an element and it is useful to implement actions that affect the selected content or context frame.

The android Context Menu is more like the menu which displayed on right-click in Windows or Linux.

the Context Menu offers actions that affect a specific item or context frame in the UI and we can provide a context menu for any view. The context menu won't support any item shortcuts and item icons.

Create Android Context Menu in Activity

The views which we used to show the context menu on long-press, we need to register that views using **registerForContextMenu(View)** in our activity and we need to override **onCreateContextMenu()** in our activity or fragment.

When the registered view receives a long-click event, the system calls our **onCreateContextMenu()** method. By using the **onCreateContextMenu()** method

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Button btn = (Button) findViewById(R.id.btnShow);
   registerForContextMenu(btn);
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
   menu.setHeaderTitle("Context Menu");
   menu.add(0, v.getId(), 0, "Upload");
   menu.add(0, v.getId(), 0, "Search");
}
```

Handle Android Context Menu Click Events

we can handle a context menu item click events using the **onContextItemSelected()** method.

```
@Override
public boolean onContextItemSelected(MenuItem item) {
   if (item.getTitle() == "Save") {
      // do your coding
   }
   else {
      return  false;
   }
   return true;
}
```

5.3 Dialogs

A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for events that require users to take an action before they can proceed.

1. ***Alert Dialog***

Alert Dialog in an android UI prompts a small window to make decision on mobile screen. Sometimes before making a decision it is required to give an alert to the user without moving to next activity.

This Dialog is used to show a title, buttons(maximum 3 buttons allowed), a list of selectable items, or a custom layout.

AlertDialog.Builder is used to create an interface for Alert Dialog in Android for setting like alert title, message, image, button, button onclick functionality etc.

```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
```

1. setTitle(CharSequence title) – This component is used to set the title of the alert dialog. It is optional component.
2. setIcon(Drawable icon) – This component add icon before the title. You will need to save image in drawable icon.

```
// Icon Of Alert Dialog
    alertDialogBuilder.setIcon(R.drawable.question);
```

3. setMessage(CharSequence message) – This component displays the required message in the alert dialog.

```
// Setting Alert Dialog Message
    alertDialogBuilder.setMessage("Are you sure,You want to exit");
```

4. setCancelable(boolean cancelable) – This component has boolean value i.e true/false. If set to false it allows to cancel the dialog box by clicking on area outside the dialog else it allows.

```
alertDialogBuilder.setCancelable(false);
```

5. setPositiveButton(CharSequence text, DialogInterface.OnClickListener listener) – This component add positive button and further with this user confirm he wants the alert dialog question to happen.

```
alertDialogBuilder.setPositiveButton("yes", new DialogInterface.OnClickListen
er() {
```

```
        @Override
        public void onClick(DialogInterface arg0, int arg1) {
          finish();
        }
      });
```

6. setNegativeButton(CharSequence text, DialogInterface.OnClickListener listener) – This component add negative button and further with this user confirm he doesn't want the alert dialog question to happen.

```
alertDialogBuilder.setNegativeButton("No", new DialogInterface.OnClickListene
r() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
          Toast.makeText(MainActivity.this,"You clicked over No",Toast.L
ENGTH_SHORT).show();
        }
      });
```

7. setNeutralButton(CharSequence text, DialogInterface.OnClickListener listener) – This component simply add a new button and on this button developer can set any other onclick functionality like cancel button on alert dialog.

```
 alertDialogBuilder.setNeutralButton("Cancel", new DialogInterface.OnClickList
ener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
          Toast.makeText(getApplicationContext(),"You clicked on Cance
l",Toast.LENGTH_SHORT).show();
        }
      });
```

5.6 List View

List of scrollable items can be displayed in Android using ListView. It helps you to displaying the data in the form of a scrollable list. Users can then select any list item by clicking on it. ListView is default scrollable so we do not need to use scroll View or anything else with ListView.

ListView is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.

```
<ListView
    android:id="@+id/user_list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:dividerHeight="1dp" />
```

Atributes of ListView.

android:id

This is the ID which uniquely identifies the layout.

android:divider

This is drawable or color to draw between list items.

android:dividerHeight

This specify the height of the divider between list items. This could be in dp(density pixel),sp(scale independent pixel) or px(pixel).

listSelector: listSelector property is used to set the selector of the listView. It is generally orange or Sky blue color mostly but you can also define your custom color or an image as a list selector as per your design.

android:entries

Specifies the reference to an array resource that will populate the ListView.

In android commonly used adapters are:

1. Array Adapter
2. Base Adapter

Array Adapter:

Whenever you have a list of single items which is backed by an array, you can use ArrayAdapter. For instance, list of phone contacts, countries or names.

```
ArrayAdapter adapter = new ArrayAdapter<String> (this,R.layout.ListView,R.id.
textView,StringArray);
```

2. Base Adapter:

BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView. Whenever you need a customized list you create your own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying a custom list item. ArrayAdapter is also an implementation of BaseAdapter.

5.7 Displaying Maps , Getting Location Data

Callback methods in Google Map

OnMapRreadyCallback:

This callback interface invokes when it instance is set on MapFragment object. The **onMapReady(GoogleMap)** method of **OnMapReadyCallback** interface is called when the map is ready to used. In the **onMapReady(GoogleMap)** method we can add markers, listeners and other attributes.

LocationListener:

This interface is used to receive notification when the device location has changed. The abstract method of LocationListener **onLocationChanged(Location)** is called when the location has changed.

GoogleApiClient.ConnectionCallbacks:

This interface provide callbacks methods **onConnected(Bundle)** and **onConnectionSuspended(int)** which are called when the device is to connected and disconnected.

GoogleApiClient.OnConnectionFailedListener:

This interface provide callbacks method **onConnectionFailed (ConnectionResult)** which is called when there was an error in connecting the device to the service.

The **setMyLocationEnabled()** method of GoogleMap is used to enable location layer, which allows device to interact with current location.

Required Permission in AndroidManifest.xml

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

<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

<uses-permission android:name="android.permission.INTERNET" />

To get the google API key (map_key),

Visit the Google Cloud Platform Console (https://console.cloud.google.com/google/maps-apis/overview).

And select APIs & Services > Credentials.

On the Credentials page, click Create credentials > API key. The API key created dialog displays your newly created API key.

Click Close.

The new API key is listed on the Credentials page under API keys.

Finally add <meta-data android:name="com.google.android.geo.API_KEY" android:value="@string/map_key"/> in androidmenifest.xml.

5.8 Monitoring a Location using GPS

The Location object represents a geographic location which can consist of a latitude, longitude, time stamp, and other information such as bearing, altitude and velocity.

There are following important methods which you can use with Location object to get location specific information

float distanceTo(Location dest)

Returns the approximate distance in meters between this location and the given location.

float getAccuracy()

Get the estimated accuracy of this location, in meters.

double getAltitude()

Get the altitude if available, in meters above sea level.

double getLatitude()

Get the latitude, in degrees.

double getLongitude()

Get the longitude, in degrees.

float getSpeed()

Get the speed if it is available, in meters/second over ground.

void setLatitude(double latitude)

Set the latitude, in degrees.

void setLongitude(double longitude)

Set the longitude, in degrees.

To get the current location, create a location client which is LocationClient object, connect it to Location Services using **connect()** method, and then call its **getLastLocation()** method.

abstract void onConnected(Bundle connectionHint)

This callback method is called when location service is connected to the location client successfully. You will use connect() method to connect to the location client.

abstract void onLocationChanged(Location location)

This callback method is used for receiving notifications from the LocationClient when the location has changed.

abstract void onDisconnected()

This callback method is called when the client is disconnected. You will use disconnect() method to disconnect from the location client.

Thank You