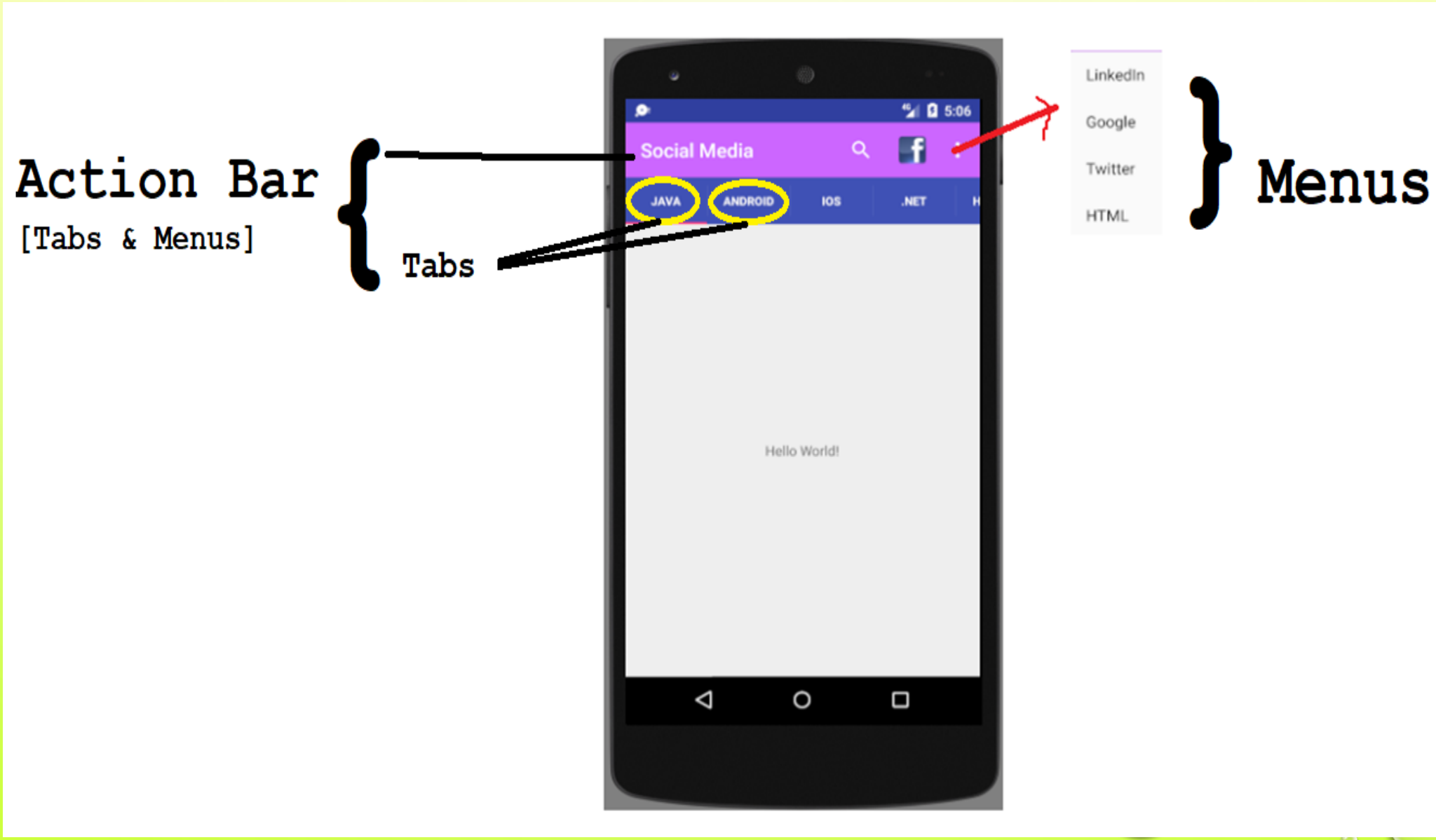
CHAPTER – 5 – ADVANCED UI COMPONENTS– 2

Menus & Action Bar, Dialogs

DAY - 3

MENUS & ACTION BAR



MENUS

- Menus in Android are usually xml resource files.
  - They can be built in code, but if your menu will be static, it should be prepared in xml.
- Add a folder to /res
  - This folder must be named menu
  - Add a new android XML file to this folder
- Must import android.View.Menu, android.View.Menuinflater, and android.View.Menuitem
- Refer https://developer.android.com/guide/topics/ui/menus.html to knows more information

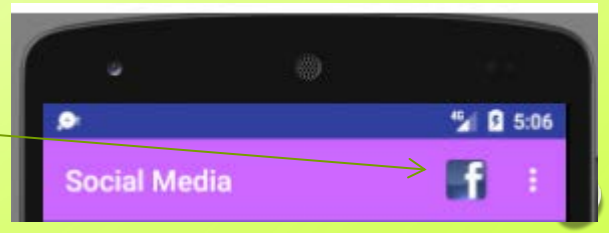
Steps to add action items to the action bar

1. Define the action items in a menu resource file.
2. Get the activity to inflate the menu resource in Activity class.
   - You do this by implementing the onCreateOptionsMenu() method.
3. Add code to say what each item should do when clicked.
   - You do this by implementing the onOptionsItemSelected() method.

STEP 1 : Menu XML

- Write click on app→res→New→Android resource directory and named the resource directory as menu
- Edit the menu directly in xml
- Menus are defined within <menu> Menu name <menu> tag pairs.
- Each menu item is defined in an <item> Item name<item/> Tag
- Submenus are defined as <menus> within an <item>. Only one level of sub-menuing is allowed.

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">  (name space is needed to use app:showAsAction="ifRoom“ )
<item
        android:id="@+id/m1"
        android:title="Facebook"
        android:icon="@drawable/fb_icon"
        app:showAsAction="ifRoom"/> // Here app is name space name which is defined in the menu header part.
</menu>
```



Items are added to the menu using the <item> element. Each action item is described using a separate <item>. The <item> element has a number of attributes you can use, here are some of the most common ones:

android:id	Gives the item a unique ID. You need this in order to refer to the item in your activity code.
android:icon	The item's icon. This is a drawable resource.
android:title	The item's text. This may not get displayed if your item has an icon if there's not space in the action bar for both. If the item appears in the action bar's overflow, only the text will be displayed.
android:orderInCategory	An integer value that helps Android decide the order in which items should appear in the action bar.

app:showAsAction: The showAsAction attribute is used to say how you want the item to appear in the action bar. If you are not using this attribute, always occurred in the overflow dot menu. As an example, you can use it to get an item to appear in the overflow rather than the main action bar, or to place an item on the main action bar only if there's room. The attribute can take the following values:

"ifRoom"	Place the item in the action bar if there's space. If there's not space, put it in the overflow.
"withText"	Include the item's title text.
"never"	Put the item in the overflow area, and never in the main action bar.
"always"	Always place the item in the main area of the action bar. This value should be used sparingly; if you apply this to many items, they may overlap each other.

STEP 2:Inflate the menu in the activity

- Once you've created a menu resource file, you add the items it contains to the action bar by implementing the activity's onCreateOptionsMenu() method.
- It runs when the action bar's menu gets created and takes one parameter, a Menu object representing the action bar.
- Here's our onCreateOptionsMenu() method:

```
import android.view.Menu;
import android.view.MenuItem;
 public class MainActivity extends AppCompatActivity {
  ....
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.test,menu); //get the menu to activity by passing xml from res→menu→test.xml
            return super.onCreateOptionsMenu(menu);
}
}
```

STEP 3 : React to action item clicks

- You get your activity to react to when an action item in the action bar is clicked by implementing the onOptionsItemSelected() method. This method runs whenever an item in the action bar is clicked.
- The onOptionsItemSelected() method takes one attribute, a MenuItem object that represents the item on the action bar that was clicked. You can use the MenuItem's getItemId() method to get the ID of the item on the action bar that was clicked so that you can perform an appropriate action, such as starting a new activity.
- Here's the code for our onOptionsItemSelected() method:

```
import android.view.Menu;
import android.view.MenuItem;

 public class MainActivity extends AppCompatActivity {
….

@Override
public boolean onOptionsItemSelected(MenuItem item) {

      switch(item.getItemId()) {

         case R.id.m1 :

            // Write your logic here

            return true;

         default : return super.onOptionsItemSelected(item);

      }

   }

}
```

Working with ActionBar in your Activity

- To work with ActionBar in your Activity, first create an object of ActionBar.

  ActiobBar aBar;

- To get the object of an ActionBar call getSupportActionBar() method;

  aBar= getSupportActionBar();

- To set the background color for your ActionBar use setBackgroundDrawable();

  aBar.setBackgroundDrawable(new ColorDrawable(Color.GREEN));

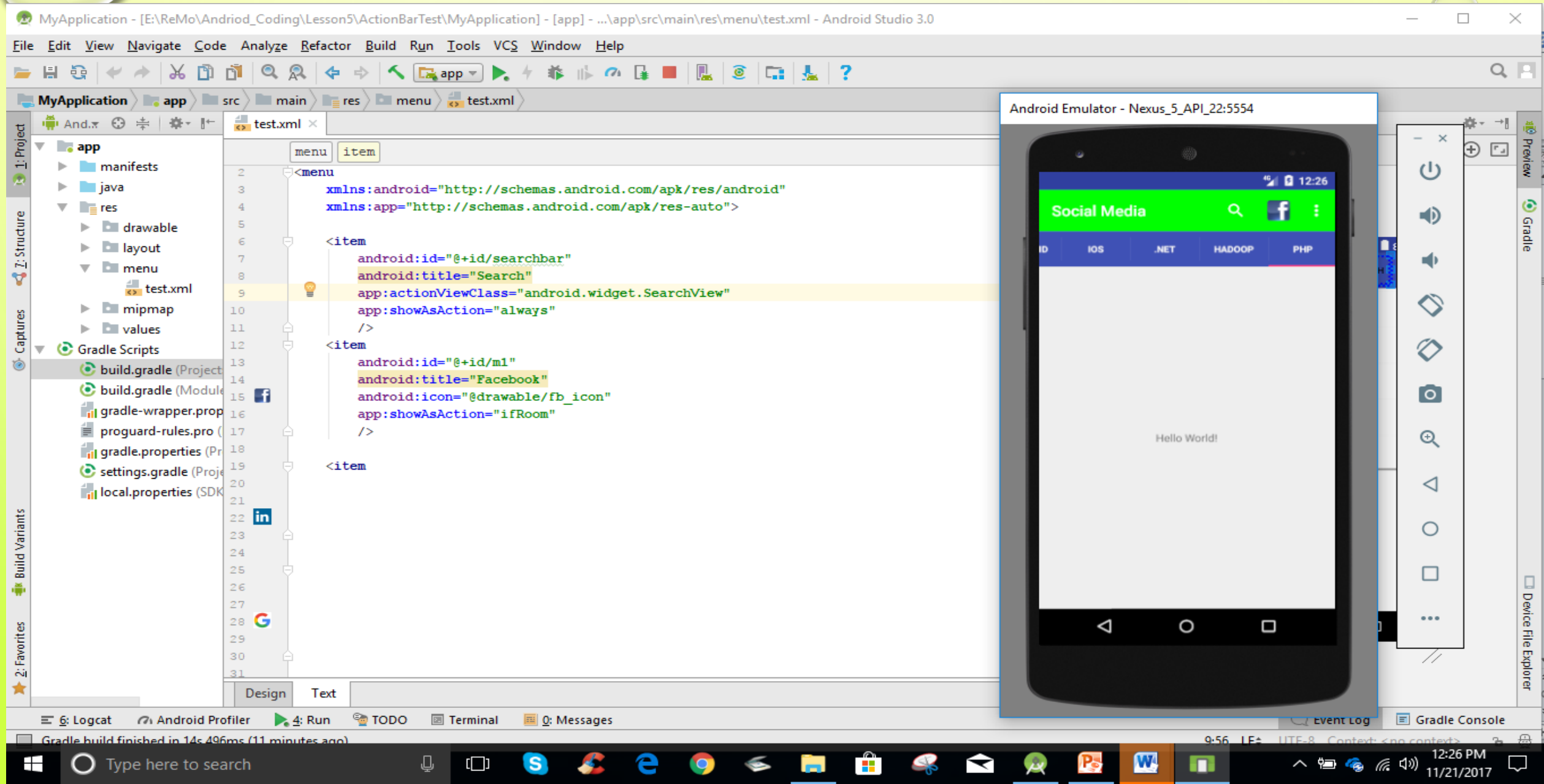
- To set the background image for your ActionBar use setBackgroundDrawable();

  aBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.abg));

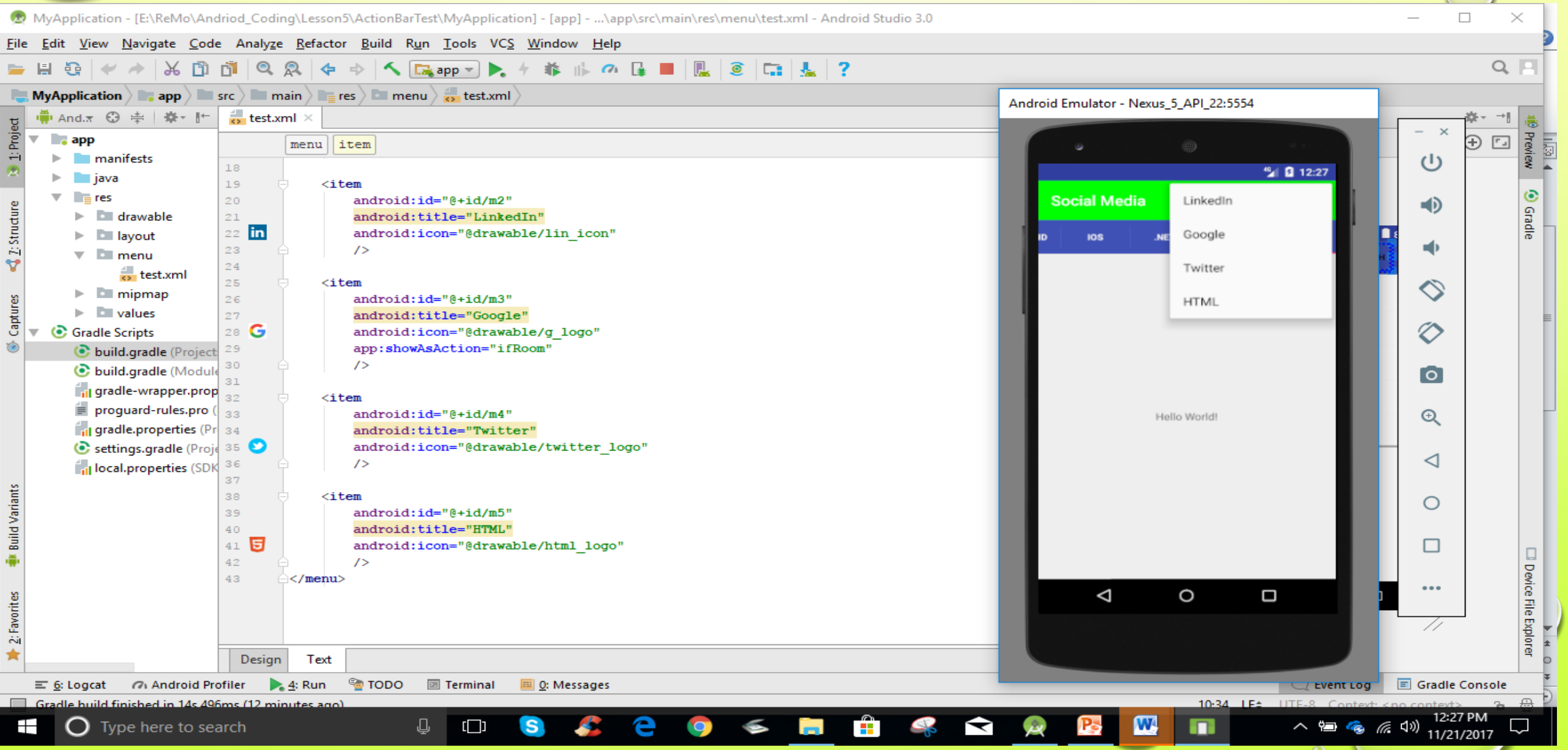
Hands on Example

- Need to create menu item using XML way as well as using Java. Perform Toast message if the user select the menu item. Also shows how to work with Search button. Refer the complete coding from demo Lesson5\ActionBarTest Folder.

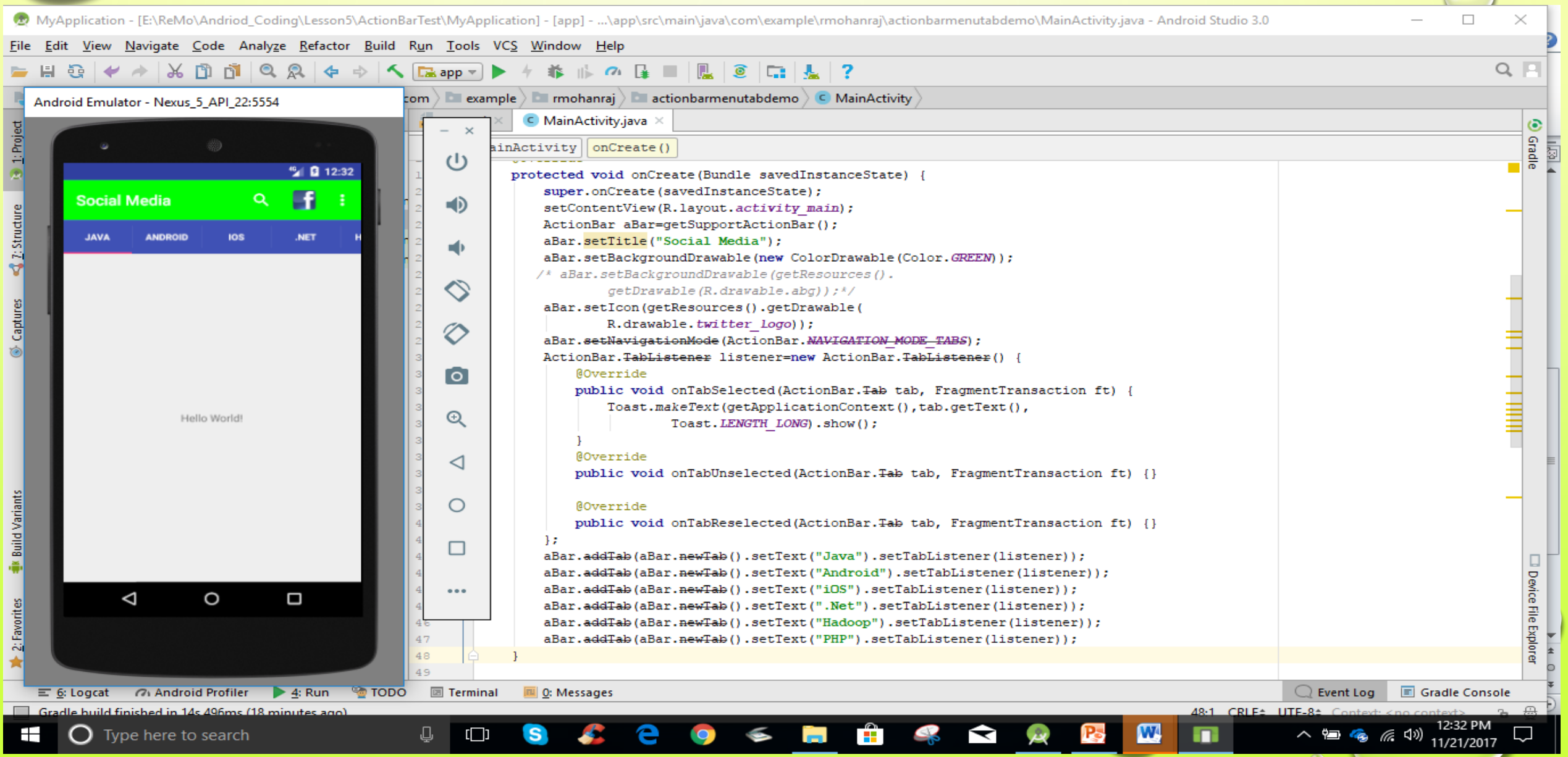
menu.xml – Screen 1- Shows the menu items of search and face book icon



menu.xml – Screen 2 shows the menu items in the three dots includes Linkedin, Google, Twitter and HTML

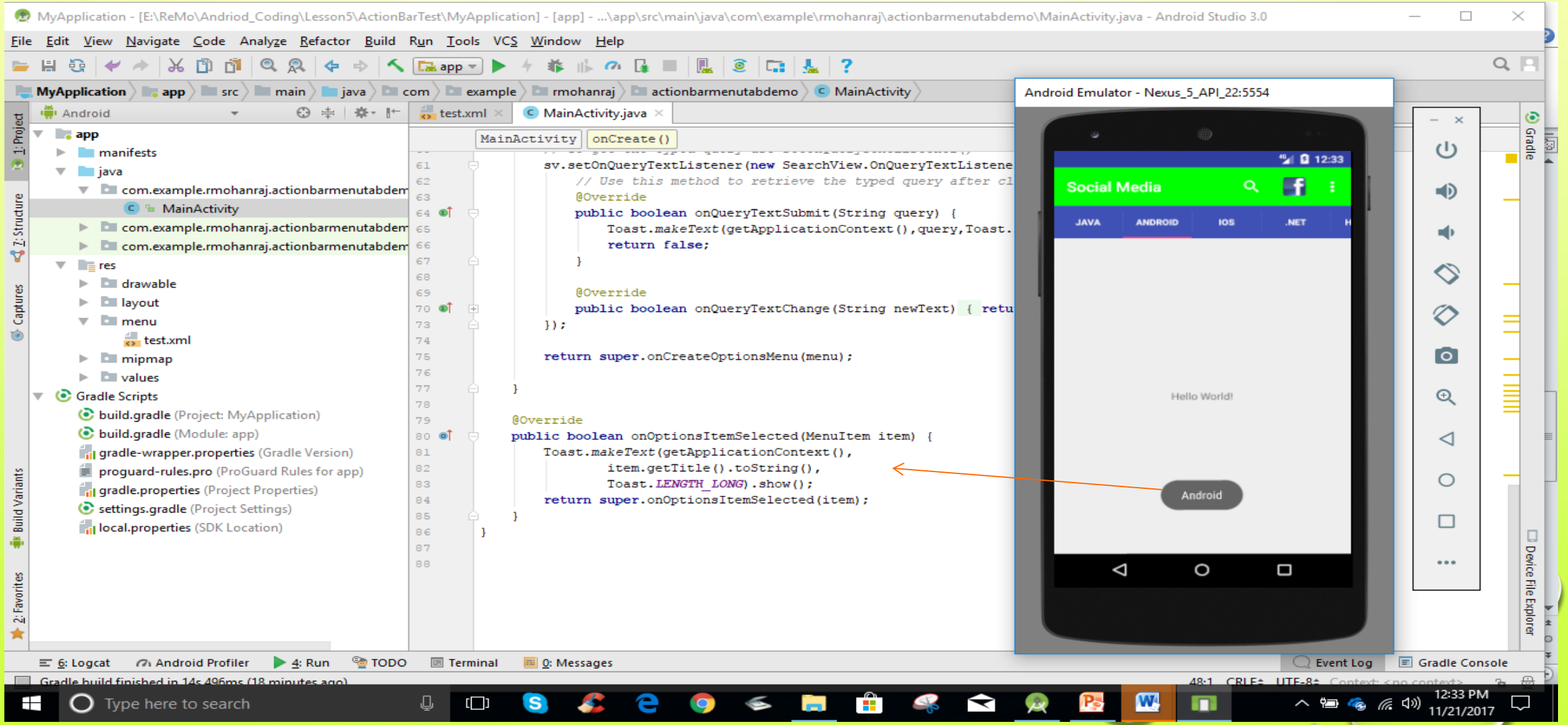


Way to add menu items from Java through Action Bar includes Java, Android, iOS,.Net, Hadoop and PHP



Action on Item selected

Once the user select the ANDROID menu, it display the Toast Message of the selected item.



Action on Item selected

Code to work with Search option.


```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.test,menu);
    // How make use of SearchView
    SearchView sv = (SearchView) menu.findItem(R.id.searchbar).getActionView(); //returns
item's action view
    // Whatever you typed to search the content, will be received using SearchManager object
    SearchManager sm = (SearchManager) getSystemService(Context.SEARCH_SERVICE);
    sv.setSearchableInfo(sm.getSearchableInfo(getComponentName()));
    // To get the typed query use setOnQueryTextListener()
    sv.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        // Use this method to retrieve the typed query after clicking the Search
        @Override
        public boolean onQueryTextSubmit(String query) {
            Toast.makeText(getApplicationContext(),query,Toast.LENGTH_LONG).show();
            return false;
        }
        @Override
        public boolean onQueryTextChange(String newText) {
            return false;
        }
    });
    return super.onCreateOptionsMenu(menu);
}
```

Action on Search option

After Click Search

After typing search Text, you will get Toast Message

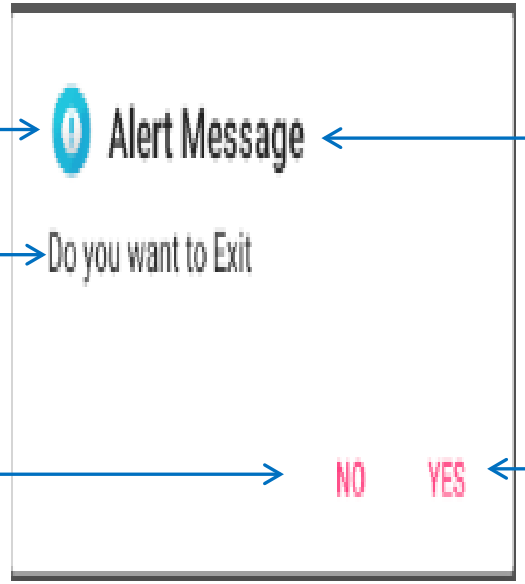
Android Dialogs

- Dialogs are prompt or alert displayed to the user to take a decision or to input any information. The dialogs are also used to notify user when a task has been completed. It does not fill the entire screen and usually appears when a user has to take a particular action before proceeding.
- Android supports different types of Dialogs
  - Alert Dialog
  - Date Picker
  - Time Picker
  - Custom Dialog
  - Progress(Custom and Alert)
  - Dialog Fragment
- In this chapter we will discuss Alert Dialog only. Date Picker and Time Picker will be discussed in Lesson -6 Fragment.

Alert Dialogs Example

- Alert Dialog is one of the built-in Dialog box with few functionalities like title, message and icon. We can create three possible choices of buttons(setPositiveButton(),setNegativeButton() and setNeutralButton(). Refer Demo : Lesson5\AlertExample
- Sample alert dialog

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setIcon(R.drawable.alerticon);
builder.setTitle("Alert Message");
builder.setMessage("Do you want to Exit");
builder.setNegativeButton("No",listener);
builder.setPositiveButton("Yes",listener);
```



Alert Dialogs Example Code and Screen shot

main > java > com > example > rmohanraj > alertdemoapp > MainActivity

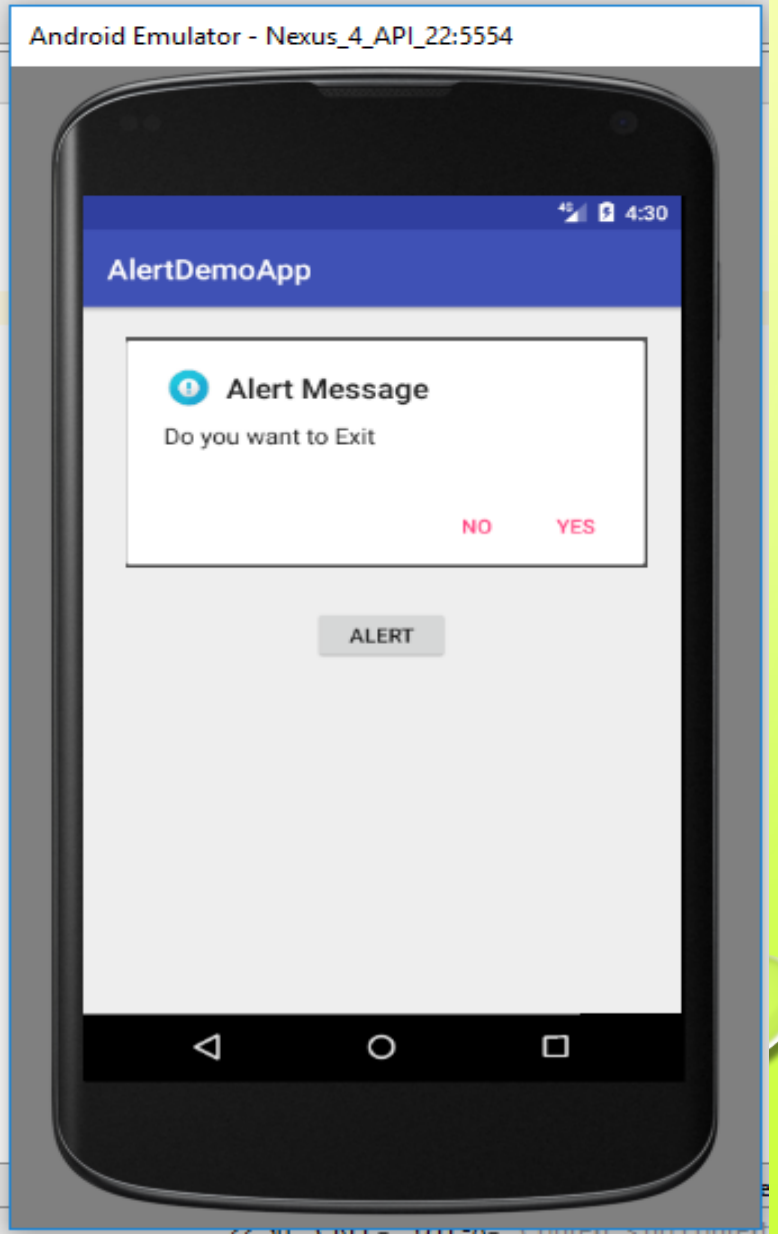
y_main.xml | MainActivity.java

MainActivity > alert()

```
public void alert(View view) {
    // 1. Create an object for AlertDialog by passing the current context object
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    // 2. Set the basic information for the builder object
    builder.setTitle("Alert Message");
    builder.setMessage("Do you want to Exit");
    builder.setIcon(R.drawable.alerticon);
    // 3. Implement the listener for the buttons you specified on Alert dialog
    AlertDialog.OnClickListener listener = new AlertDialog.OnClickListener(){
        // Write you logic for button click actions based on which buttin is clicked from the alert dialog
        @Override
        public void onClick(DialogInterface dialog, int which) {
            if(which==dialog.BUTTON_POSITIVE){
                dialog.dismiss(); // dismiss the dialog
                finish(); // to destroy the activity
            }
            else if(which==dialog.BUTTON_NEGATIVE){
                dialog.dismiss(); // dismiss the dialog, but activity is still alive
            }
        }
    };
    // 4. Here we are going to set two buttons on the Alert dialog
    builder.setPositiveButton( text: "Yes",listener);
    builder.setNegativeButton( text: "No",listener);
    // 5. need to show the dialog
    builder.show();
}
}
```

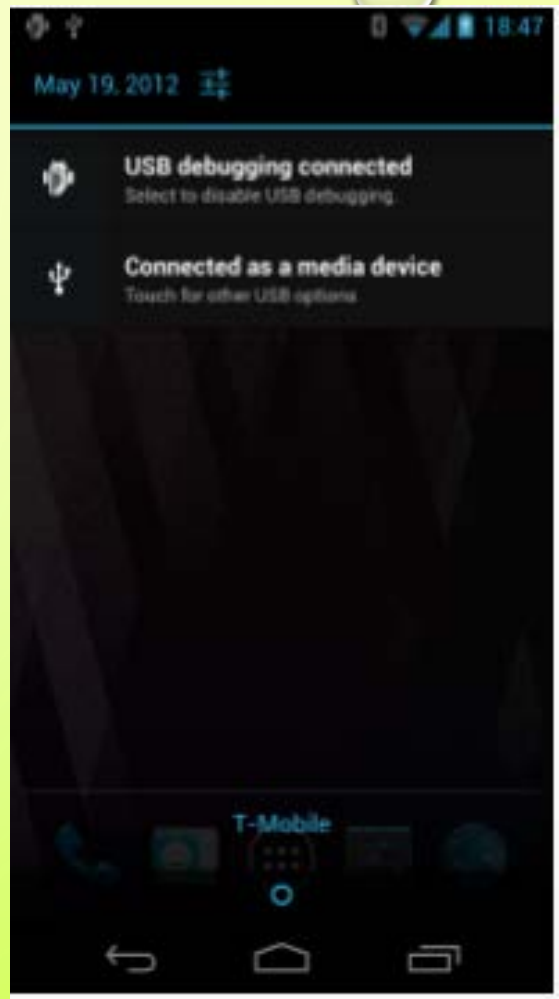
Android Profiler | 0: Messages | 4: Run | TODO

's 459ms (2 minutes ago)



Notifications

- Notifications are used to notify or provide alerts to the user when a message or notification arrives.
- Notification contains icon, title, body, and the notification arrival time.
- The Notification Manager ensures that the status bar icons are updated regularly.
- Your current phone may well have such icons, to indicate battery life, signal strength, whether Bluetooth is enabled and email notifications.
- All these actions start with and are represented by the Notification class. The Notification class defines how you want to represent a notification to a user.
- The NotificationManager class, though, is required in order to use the Notification class, because it's the system service that executes and manages notifications.



Notifications

- Using a notification follows these steps:
- Step 1: Here we set up a NotificationManager and instantiate it.

```
NotificationManager myNotificationManager;
private static final int NOTIFICATION_ID = 1;
myNotificationManager = (NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
```

- Step 2: Next we use the Notification.Builder to set Notification objects such as the message icon for the notification, the title, and much more. The Notification.Builder provides a much simpler mechanism for building notifications.

```
Notification.Builder builder = new Notification.Builder(this);
builder.setTicker("Message to Show when Notification pops up");
builder.setContentTitle ("Title of Message");
 builder.setSmallIcon(R.drawable.icon);
builder.setContentText("- Message for the User -");
```

Notifications

- Using a notification follows these steps:
- Step 3: Next we create PendingIntent for the Builder. You must create a PendingIntent for all notification.

```
Intent notificationIntent = new Intent(this, SimpleNotification.class);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0);
builder.setContentIntent(contentIntent);
```

- Step 4: Finally, to send the notification, all you have to do is use the notify() method and supply the Notification ID as well as the builder.

```
myNotificationManager.notify(NOTIFICATION_ID, builder.getNotification());
```

- Here the notify() method wakes up a thread that performs the notification task you have defined.
- Refer Demo : Lesson5\NotificationExample folder