MODULE

Navigation Drawer, Toolbar, FAB, and Snackbar

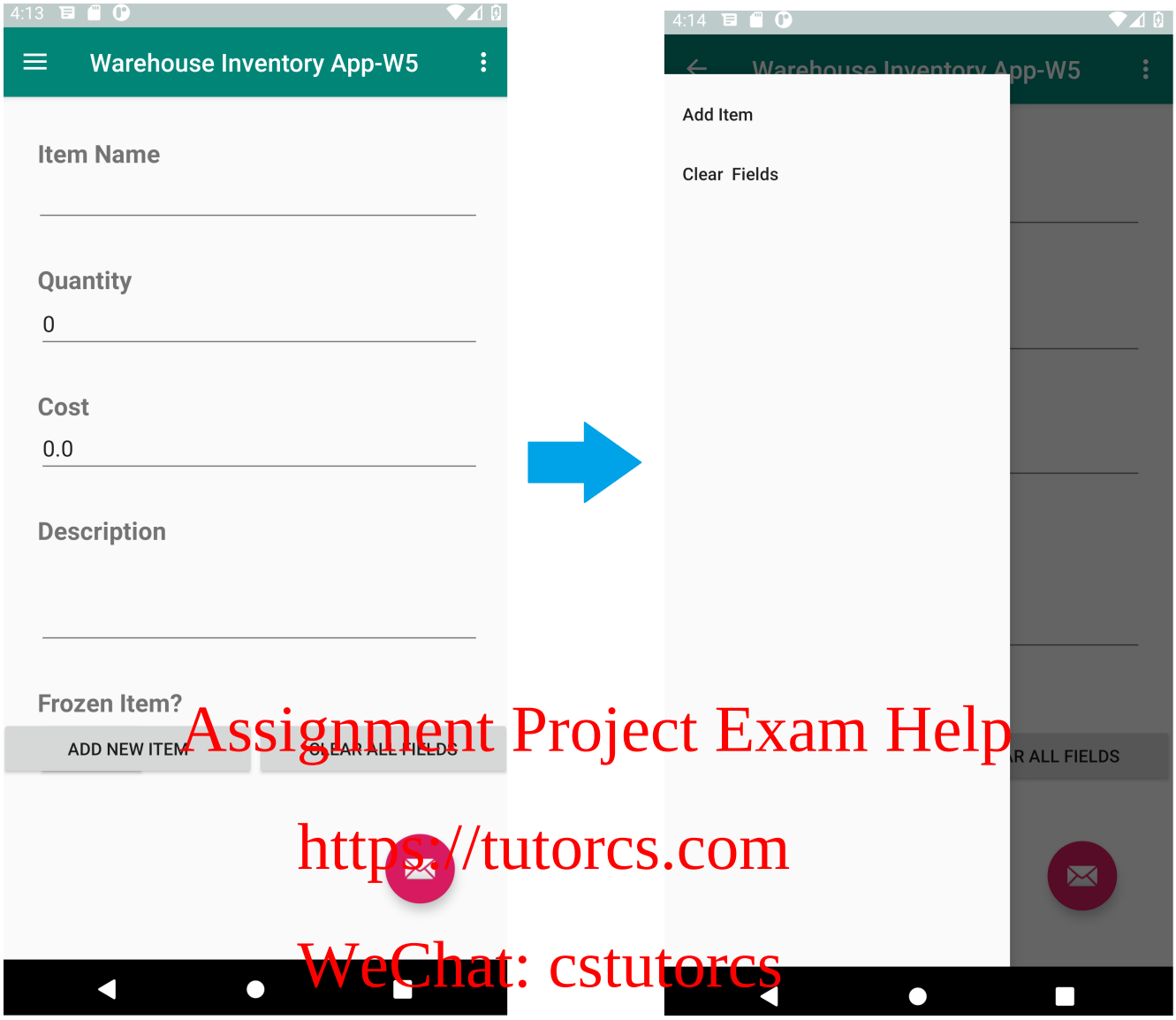
Nawfal Ali
Updated 25 November 2020

Click HERE to download the source code of this workshop material.

Drawer layout

DrawerLayout acts as a top-level container for window content that allows for interactive "drawer" views to be pulled out from one or both vertical edges of the window.

Drawer positioning and layout is controlled using the `android:layout_gravity` attribute on child views corresponding to which side of the view you want the drawer to emerge from: left or right (or start/end on platform versions that support layout direction.) Note that you can only have one drawer view for each vertical edge of the window. If your layout configures more than one drawer view per vertical edge of the window, an exception will be thrown at runtime.

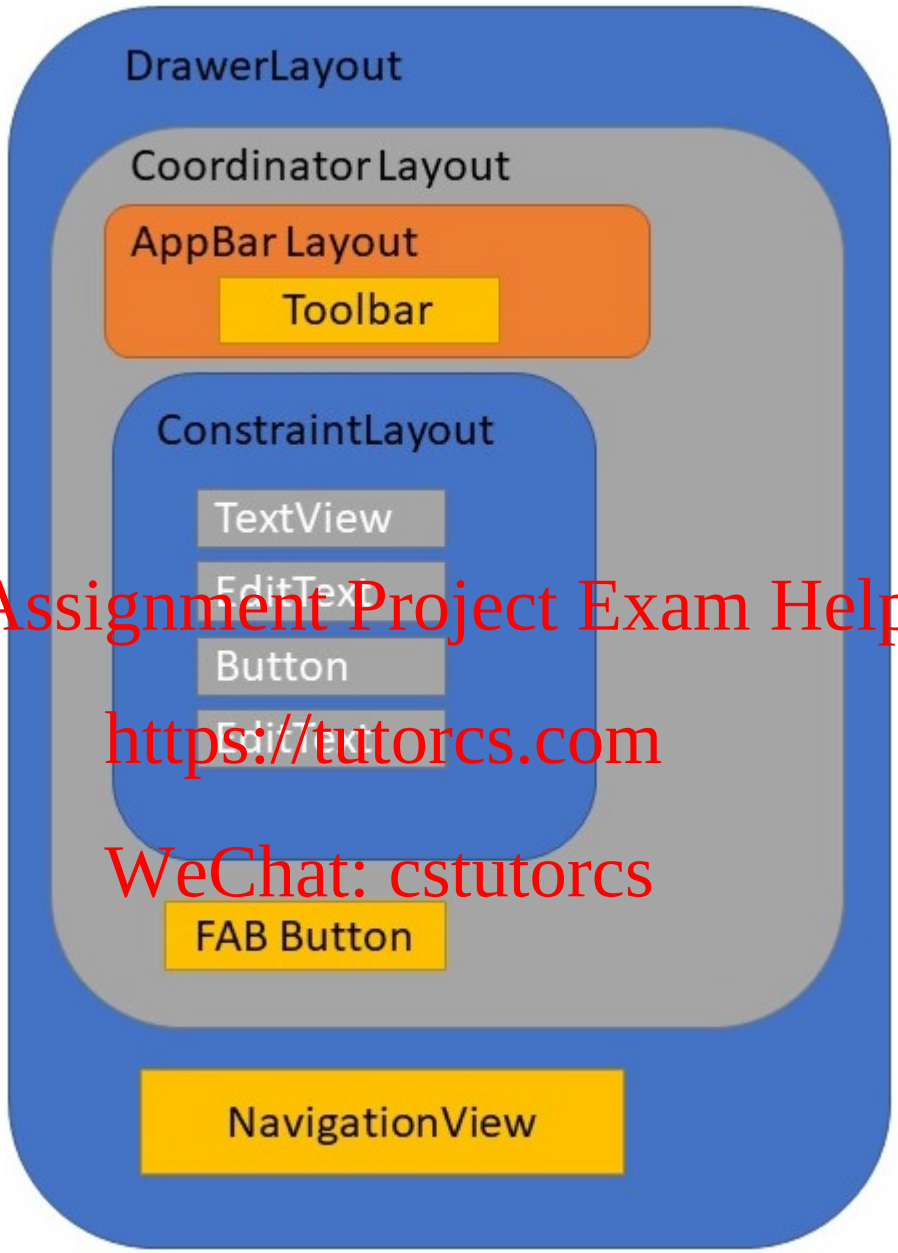


How to add a navigation drawer to your application and listen to its events?

The top-level of our hierarchy is a DrawerLayout. In this layout, we have two items:

a. a layout of type CoordinatorLayout, which contains:
   i. a layout of type AppBarLayout that wraps a Toolbar
   ii. a constraint layout that represents your main white area and should include your views such as text views, buttons, edit texts, etc.
   iii. a floating action button (FAB button)
b. a navigation view that represents the drawer that can be pulled from left or right

i. A menu resource file containing the options to be displayed within the navigation drawer.



> The navigation drawer is a panel that slides out from the left of the screen and contains a range of options available for selection by the user, typically intended to facilitate navigation to some other part of the application.

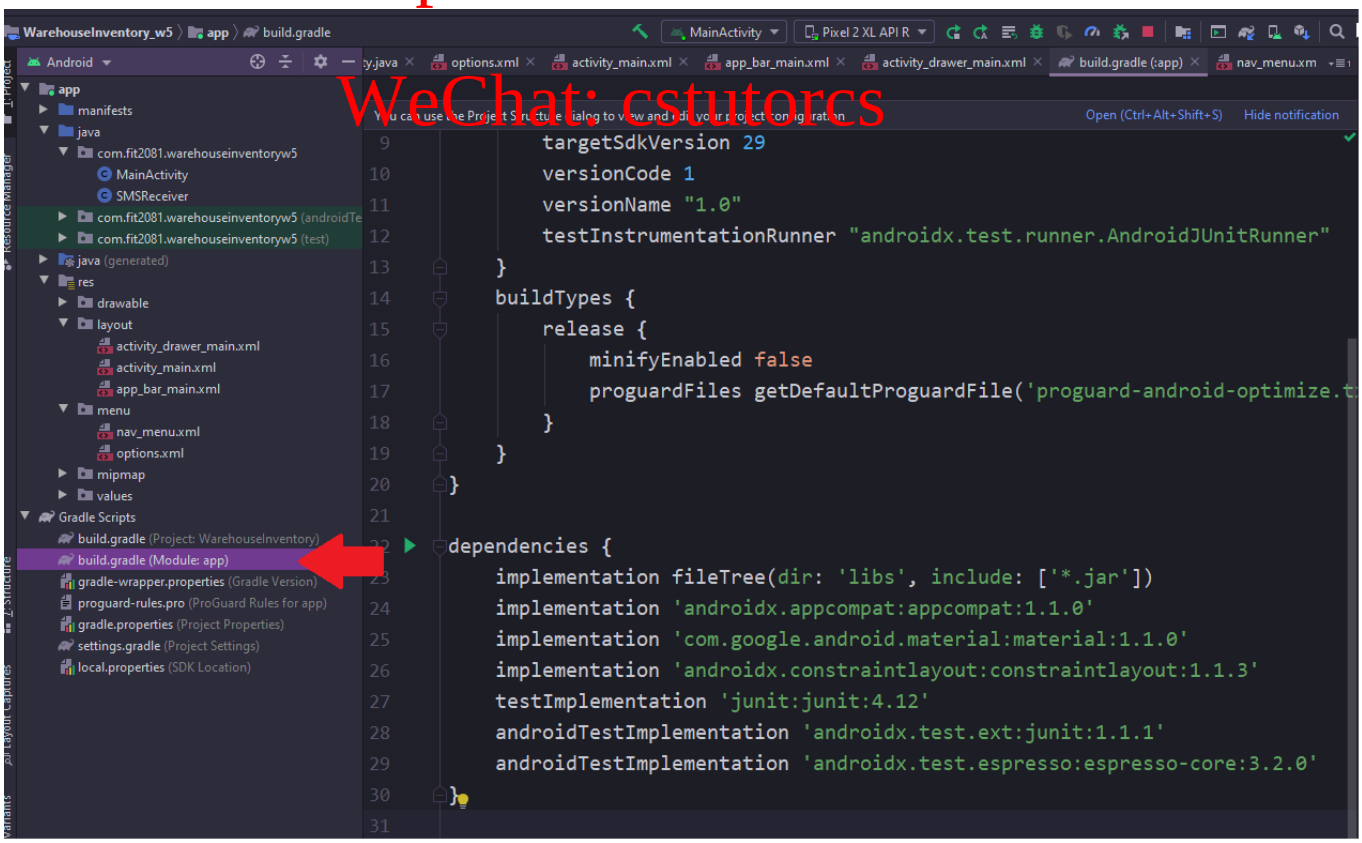
> AppBarLayout is a ViewGroup, most commonly used to wrap a Toolbar, that provides many of the Material Design features. Inside Toolbar we can design our action bar now as we want.

CoordinatorLayout

CoordinatorLayout is a general-purpose container that allows for coordinating interactive behaviors between its children.

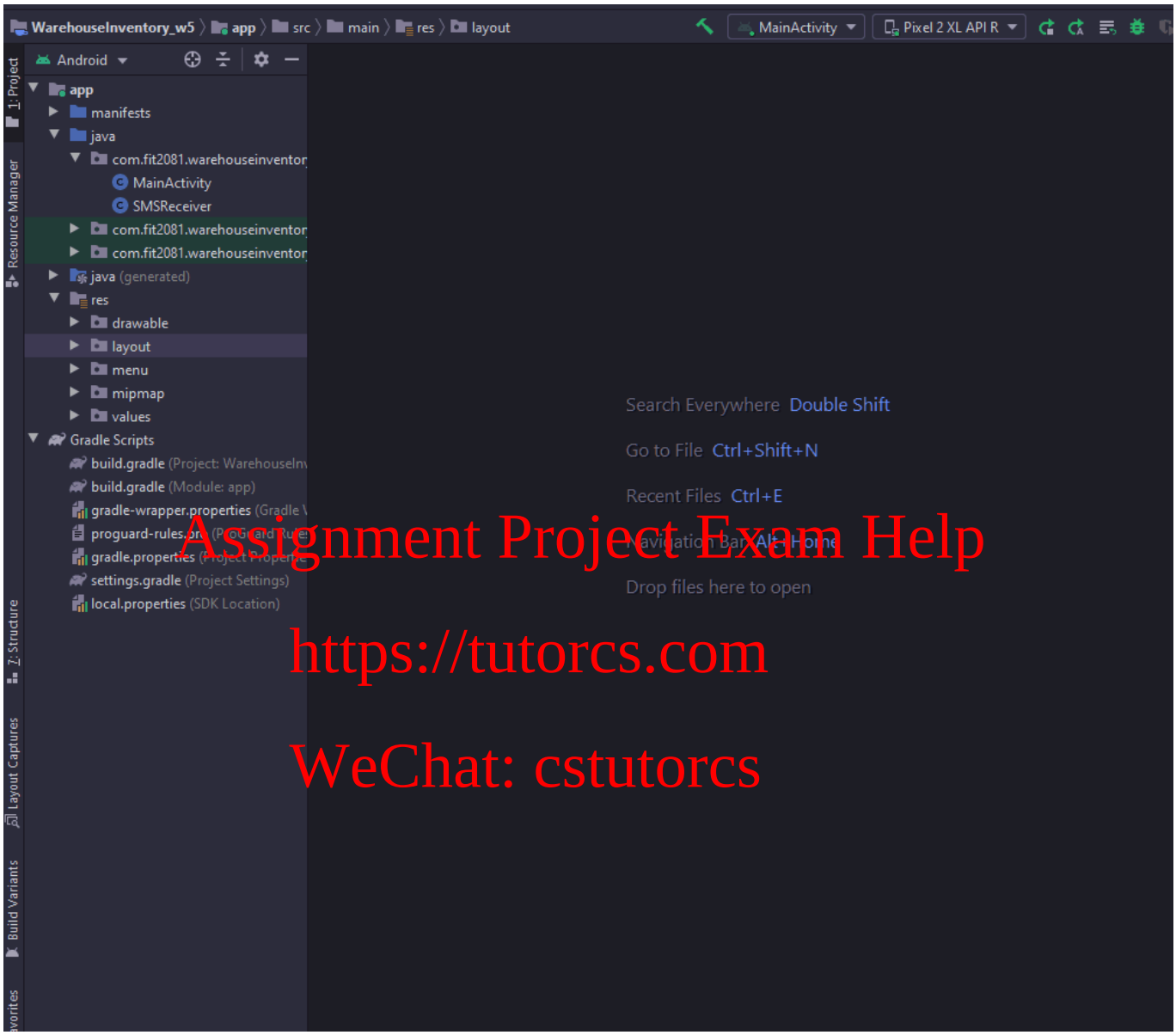
open build.gradle(Module: app) and add the following dependencies:

```
implementation 'androidx.appcompat:appcompat:1.1.0'
implementation 'com.google.android.material:material:1.1.0'
implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
```

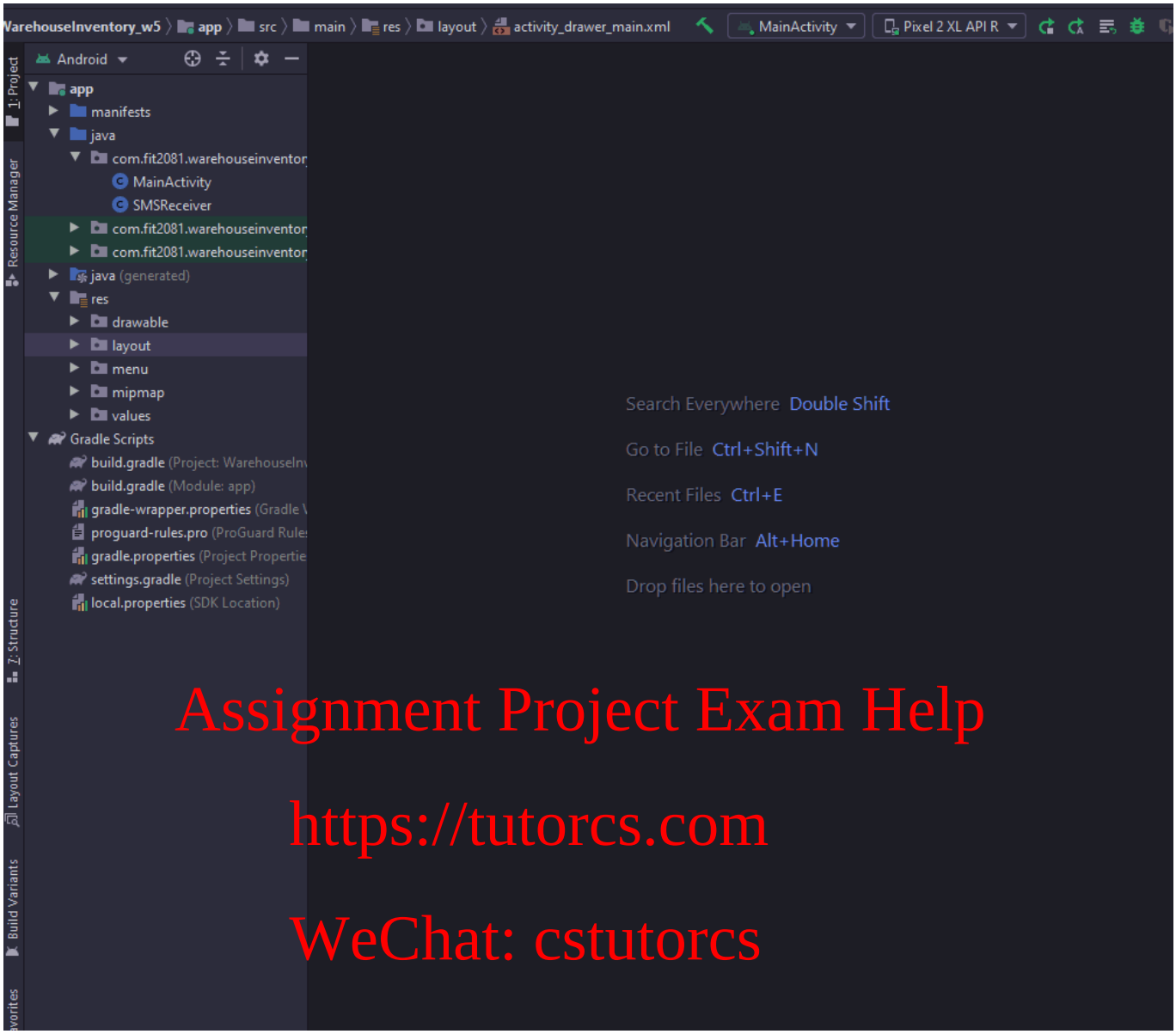
Add a new layout resource file where:

- file name: activity_drawer_main.xml (or any name you prefer)
- root element: androidx.drawerlayout.widget.DrawerLayout



Inside the drawer layout, we have to include two items:

- a layout that should contain the toolbar, a constraint layout and a floating action button.
  - file name: app_bar_main (or any name you prefer)
  - root element: androidx.coordinatorlayout.widget.CoordinatorLayout



The file that we have just created should be included into the drawerlayout using <include tag as shown below:

```
<include
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

The second item that should be added to the drawer layout is the navigation view that represents the navigation drawer

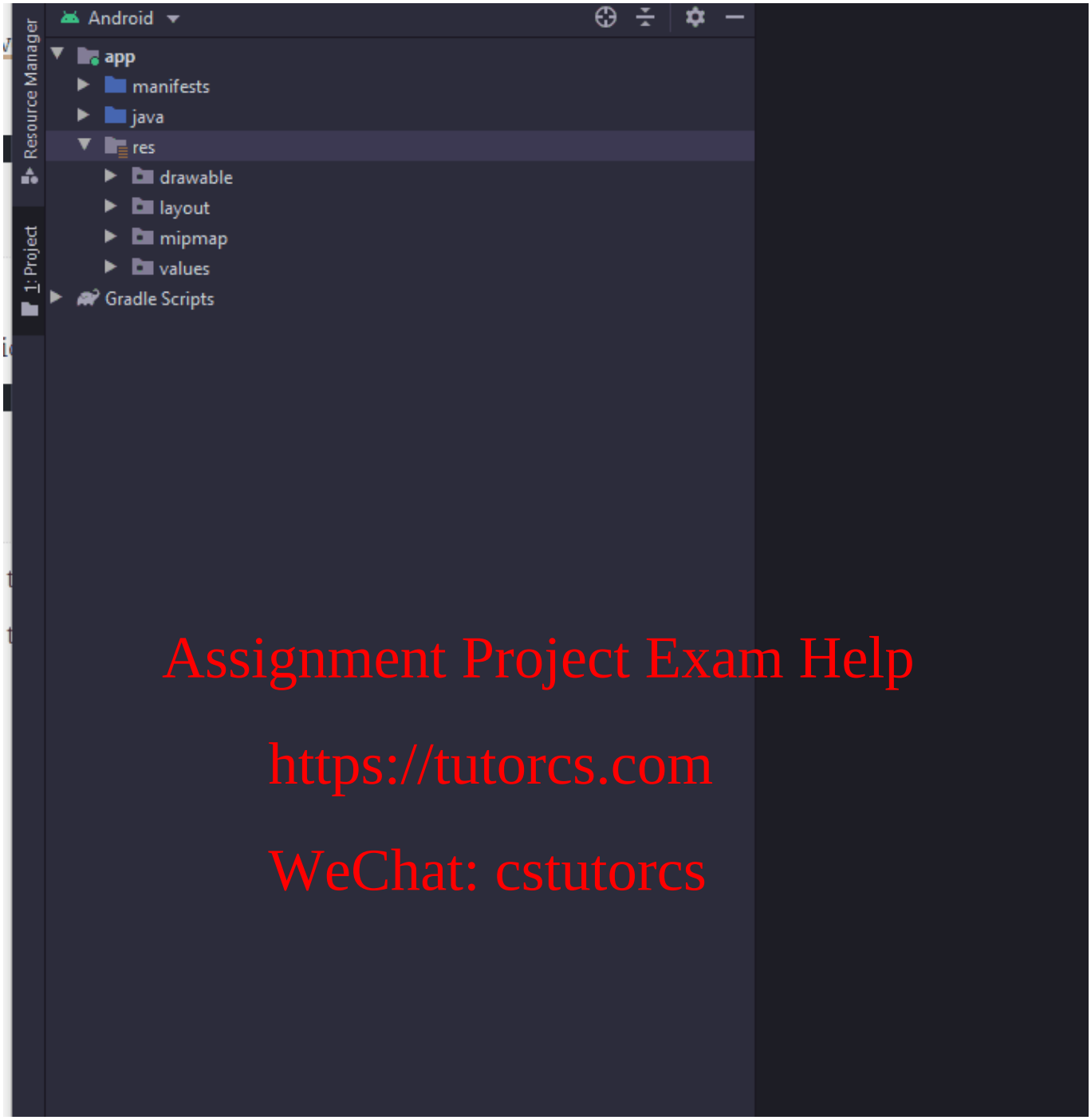
```
<com.google.android.material.navigation.NavigationView
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/nav_view"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:menu="@menu/nav_menu" />
```

The attribute android:layout_gravity=”start” tells the parent to position the drawer on the left side of the screen.

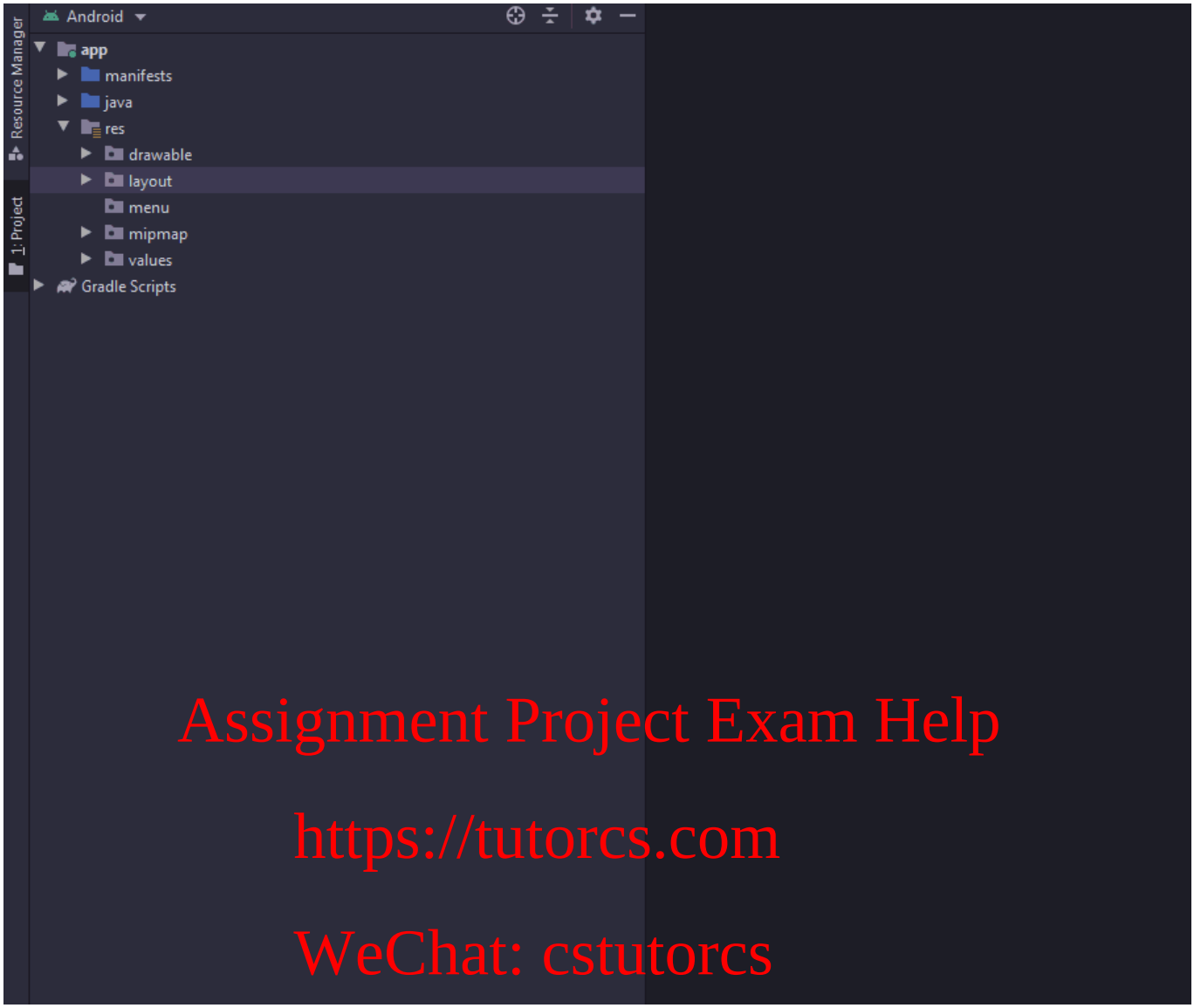
The attribute app:menu=”@menu/nav_menu” represents a reference to the menu file *nav_menu*

How to create a resource menu file

Step 1: Create an Android Resource Directory named “menu” as shown below:



Step 2: right-click the menu folder select New–>Menu Resource File–> provide a name



Step 3: Let's create two menu items.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/item_id_1"
        android:title="Item One" />
    <item
        android:id="@+id/item_id_2"
        android:title="Item Two" />
</menu>
```

Where:

- Attribute android:id="@+id/item_id_1" represents a unique ID for the menu item
- attribute android:title="Item One" sets the title for the menu item

app_bar_main.xml

Add a toolbar to your coordinator layout.

```
<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
    />
</com.google.android.material.appbar.AppBarLayout>
```


The toolbar is wrapped by AppBarLayout to get the material design features.

Now, beneath the toolbar, we should include our main content (activity_main).

```
<include
    layout="@layout/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

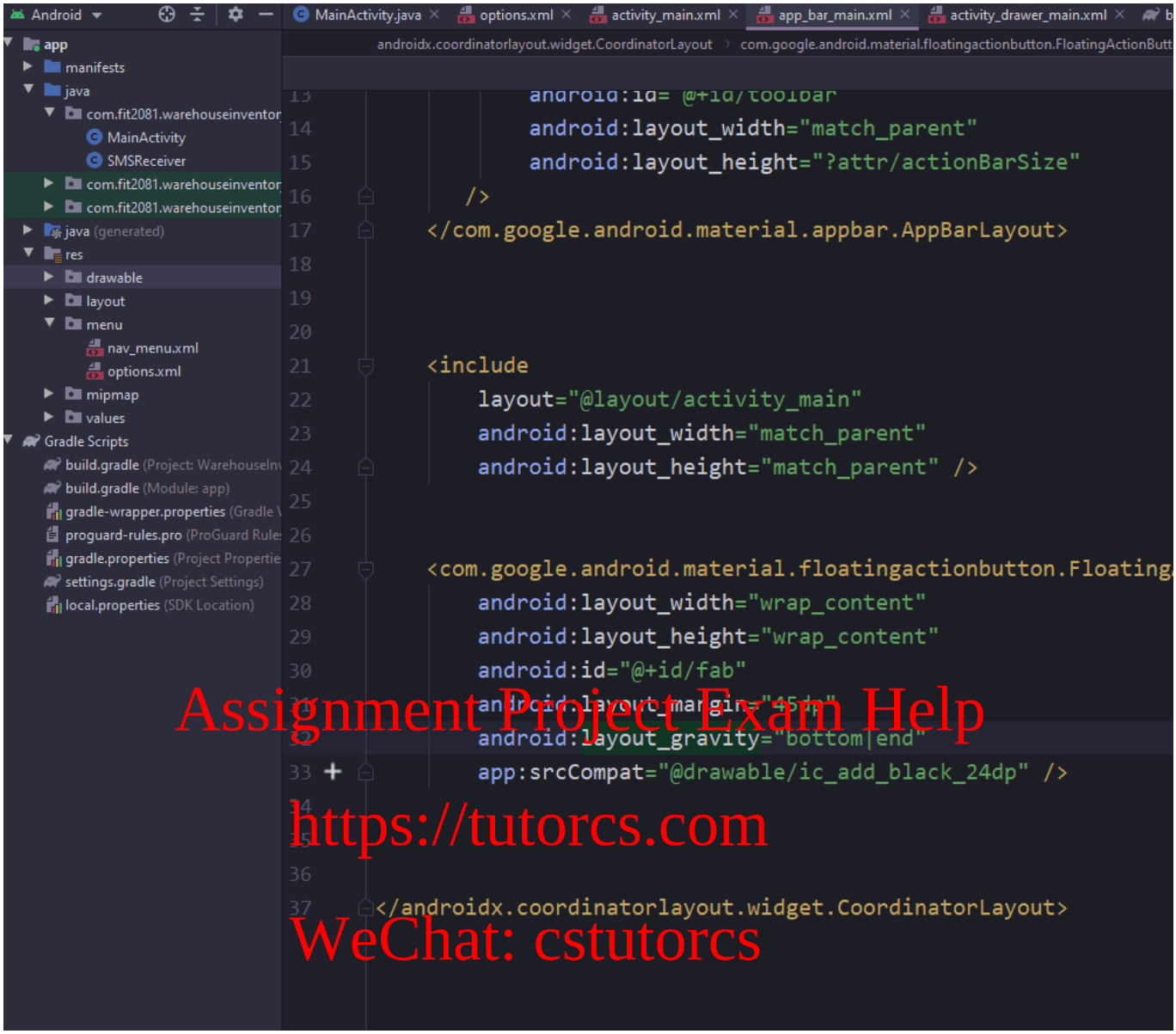
At the bottom of the page, let's add a floating action button (FAB button).

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/fab"
    android:layout_margin="45dp"
    android:layout_gravity="bottom|end"
    app:srcCompat="@drawable/ic_add_black_24dp" />
```

where:

- android:layout_margin: specifies the margin around the fab button
- android:layout_gravity=”bottom|end”: specifics the location of the fab button. As shown, we are asking to be placed in the bottom right corner of the screen
- app:srcCompat=”@drawable/ic_add_black_24dp”: this attribute specifies the icon that should be used for fab button.

To add a new ICON for your app:

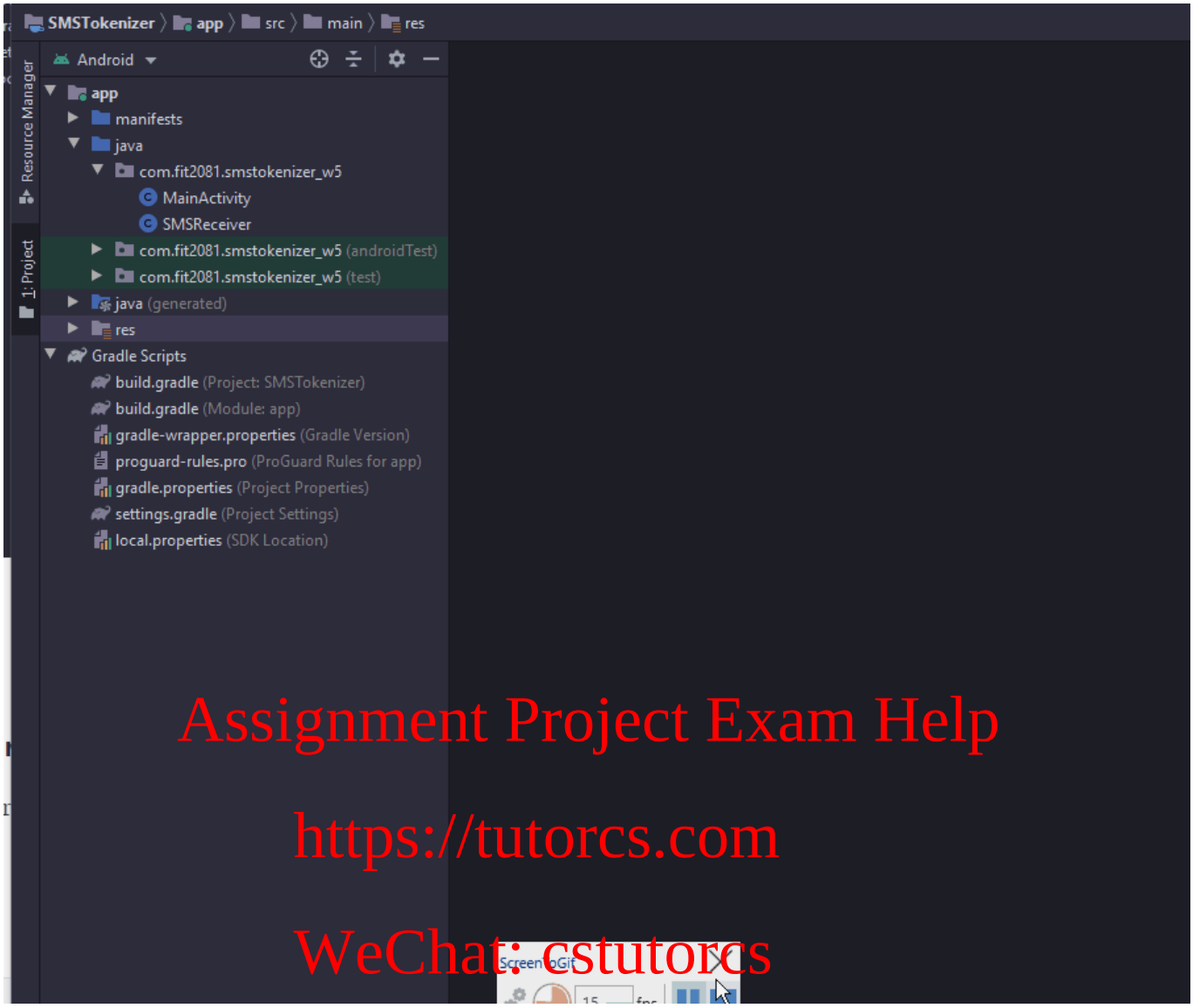


You cannot implement a new Toolbar if the activity already has an action bar supplied by the window decor

To disable the default toolbar, you need to use a theme that has no toolbar.

Step 1: open values–>styles.xml

Step 2: change the default parent theme to be "Theme.AppCompat.Light.NoActionBar"



MainActivity.java

In the MainActivity.java class, we have to do the following:

- Hook the drawer and the toolbar
- Set the navigation items listener to the navigation view
  - Implement the navigation items listener
- Listen to FAB events

Hook the drawer and the toolbar

Create references to the drawer layout, navigation view, and the toolbar

```
drawerlayout = findViewById(R.id.drawer_layout);
navigationView = findViewById(R.id.nav_view);
toolbar = findViewById(R.id.toolbar);
```

Set the toolbar to be the current toolbar for the activity

```
setSupportActionBar(toolbar);
```

Hook the drawer and the toolbar using ActionBarDrawerToggle class

```
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        this, drawerlayout, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
drawerlayout.addDrawerListener(toggle);
toggle.syncState();
```


ActionBarDrawerToggle is for displaying an drawer indicator in the appbar which needs 5 arguments.

1. Current Activity context
2. DrawerLayout variable
3. Toolbar variable
4. Drawer open description message via Resource string
5. Drawer close description message via Resource string

Set the navigation items listener to the navigation view

```
navigationView.setNavigationItemSelectedListener(new
MyNavigationListener());
```

Implement the navigation items listener

```
class MyNavigationListener implements
NavigationView.OnNavigationItemSelectedListener {

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        // get the id of the selected item
        int id = item.getItemId();

        if (id == R.id.item_id_1) {
            // Do something
        } else if (id == R.id.item_id_2) {
            // Do something
        }
        // close the drawer
        drawerLayout.closeDrawers();
        // tell the OS
        return true;
    }
}
```


Listen to FAB events

Step 1: Create a reference to the fab button:

```
FloatingActionButton fab = findViewById(R.id.fab);
```

Step 2: set a new listener

```
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
```

```
        Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
```

The listener is an anonymous class of type View.OnClickListener. If the user clicks or taps on the fab button, the callback method "onClick" will get executed.

The method shows a Snakbar from the bottom of the screen.

> The `Snackbar` widget provides brief feedback about an operation through a message at the bottom of the screen. Snackbars disappear automatically either after a timeout or after a user interaction elsewhere on the screen, and can also be swiped off the screen.

References:

- https://www.androdocs.com/kotlin/implementing-navigation-drawer-in-android-app-using-kotlin.html
-
