In styles file change maind style ending to the NoActionBar, it will delete default line on the top

Then create toolbar in xml:

<android.support.v7.widget.Toolbar  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize" - set size as at common toolbar   
 android:background="@color/colorPrimaryDark" - set main color of bg   
 android:theme="@style/ThemeOverlay.AppCompat.Light" - style  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Dark" - style   
 android:id="@+id/toolBar"  
 ></android.support.v7.widget.Toolbar>

Then add this file in main layout:

<include layout="@layout/tool\_bar"/>

To add some buttons or icons create new xml menu type and there add icons (search google icons for android in png and add them to the mipmap as hdpi type):

<menu  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 >  
 <item  
 android:id="@+id/action\_share"   
 android:title="Rotate" - set title  
 android:icon="@mipmap/baseline\_3d\_rotation\_white\_24" - add icon from mipmap   
 app:showAsAction="ifRoom" - add if it has enough space   
 />  
 <item  
 android:id="@+id/action\_settings"  
 android:title="settings"  
 app:showAsAction="never"- add as menu-part (it will be visible when you tap a menu icon, it will appear automaticly

/>   
</menu>

Then in main class create variable Toolbar and set in main function, than override onCreateOptionsMenu(Men menu):

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
  
 MenuInflater inflater = getMenuInflater();  
 inflater.inflate(R.menu.bar\_menu, menu); - set your new xml file with icons  
 return true;  
}

In main:

setSupportedActionBar(toolbar);

getSupportedActionBar().setTitle(“…”);

Action Handlers

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()){  
 case R.id.action\_share:  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
}

Set button go home

In main activity:

ActionBar actionBar = getSupportActionBar();  
actionBar.setDisplayHomeAsUpEnabled(true);

actionBar.setHomeAsUpIndicator(R.drawable….); - change icon

@Override – on click   
public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()){  
 case android.R.id.home:  
   
 }  
 return super.onOptionsItemSelected(item);  
}

In androidManifest.xml find activity and add parameter android:parentActivityName=”…” and supporting for lover versions: add between activity:

<meta-data android:name="android.support.PARENT\_ACTIVITY" android:value="…"></meta-data>

Search button

Create item in menuToolbar and set there parameter:

app:actionViewClass="android.support.v7.widget.SearchView"

Then in main activity in onCreateOptionsMenu method create listener:

MenuItem.OnActionExpandListener onActionExpandListener = new MenuItem.OnActionExpandListener() {  
 @Override  
 public boolean onMenuItemActionExpand(MenuItem item) { - when you open it  
 return true;  
 }  
  
 @Override  
 public boolean onMenuItemActionCollapse(MenuItem item) { - when you close it  
 return true;  
 }  
};  
MenuItem rotate = menu.findItem(R.id.action\_share); - get menu item  
rotate.setOnActionExpandListener(onActionExpandListener); - set listener at menu item

Search listener on change and on submit

Implement SearchView.OnQueryTextListener and override needed methods (can be in example), then in oncreate options menu:

MenuItem rotate = menu.findItem(R.id.action\_share); - get MenuItem with search action (when you create it, you add this parameter to item app:actionViewClass="android.support.v7.widget.SearchView" )

SearchView searchView = (SearchView)rotate.getActionView(); - get search  
searchView.setOnQueryTextListener(this); - set listeners, what you have override

If you have many activities with the same toolbar, you can move it to another class and extend your classes from it, in that case if you want to add new button in some kind activity you should:

1 – override onCreateOptionsMenu as well as at example but in super set:

Super.onCreateOptonsMenu(menu)

And override onOptionsItemSelected, where you handle new buttons and if there are presses common one, you return:

Super.onOptionsItemSelected(item)

Create and delete items at runtime

@Override  
public boolean onPrepareOptionsMenu(Menu menu) { - override this method, it happens after each invalidation   
  
 menu.removeItem(101); - remove item  
  
 MenuItem menuItem = menu.add(menu.NONE, 101, 1, "aaaa"); - create item, second – id, third - order  
 menuItem.setIcon(R.drawable….); - set icon  
 menuItem.setShowAsActionFlags(MenuItem.SHOW\_AS\_ACTION\_ALWAYS);  
 menuItem.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener(){} – set listener  
}  
 invalidateOptionsMenu(); - ran this method to open overrode one