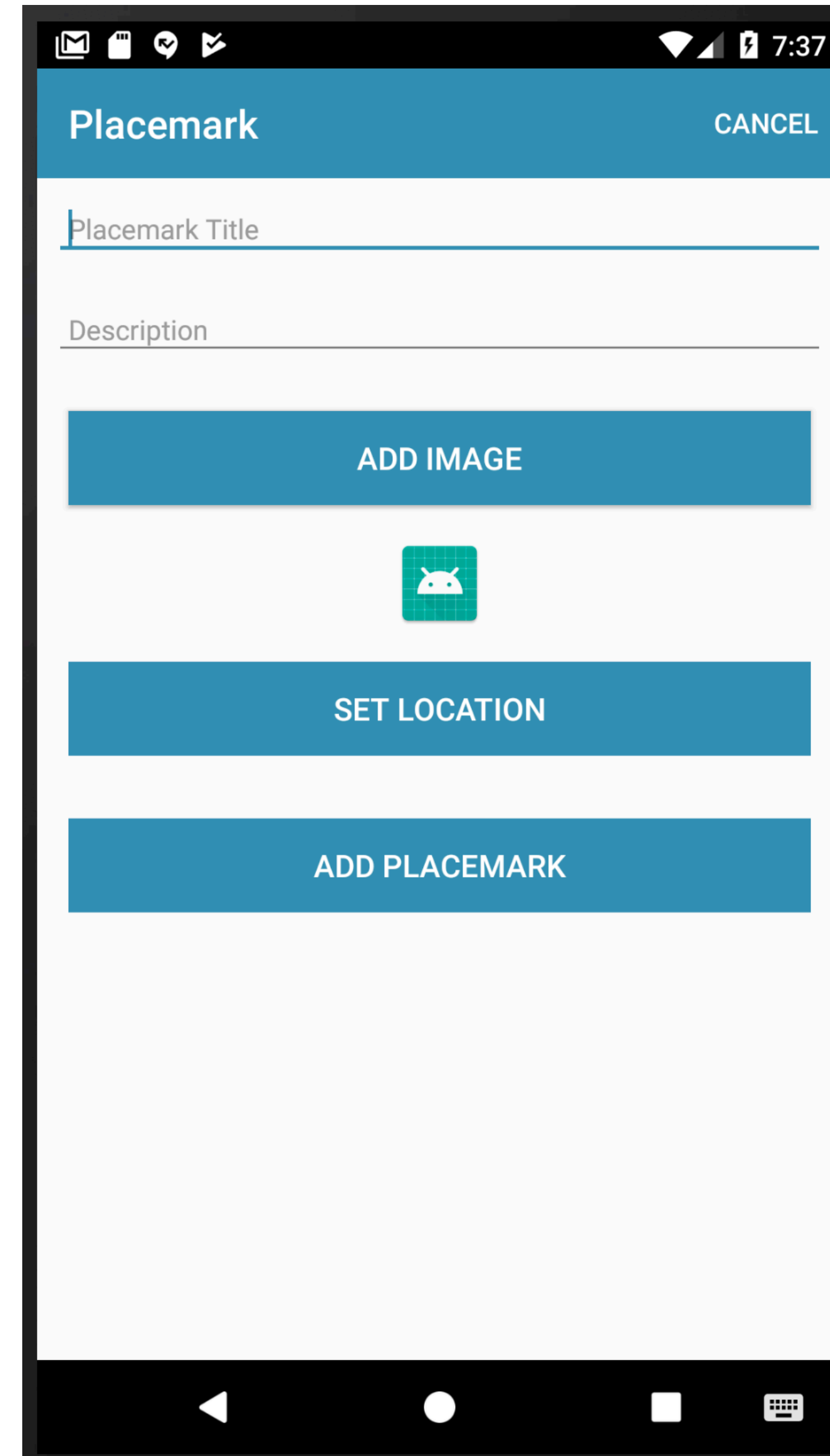


AppBarLayout + Toolbar



Support Library

<https://developer.android.com/topic/libraries/support-library/>

When developing apps that support multiple API versions, you may want a standard way to provide newer features on earlier versions of Android or gracefully fall back to equivalent functionality. Rather than building code to handle earlier versions of the platform, you can leverage these libraries to provide that compatibility layer. In addition, the Support Libraries provide additional convenience classes and features not available in the standard Framework API for easier development and support across more devices.

Originally a single binary library for apps, the Android Support Library has evolved into a suite of libraries for app development. Many of these libraries are now a strongly recommended, if not essential, part of app development.

build.gradle

```
dependencies {  
    ...  
    implementation 'com.android.support:design:28.0.0-rc02'  
    ...  
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="org.wit.placemark.activities.PlacemarkActivity">
```

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
```

```
<android.support.design.widget.AppBarLayout
    android:id="@+id/appBarLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/colorAccent"
    android:fitsSystemWindows="true"
    app:elevation="0dip"
    app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
```

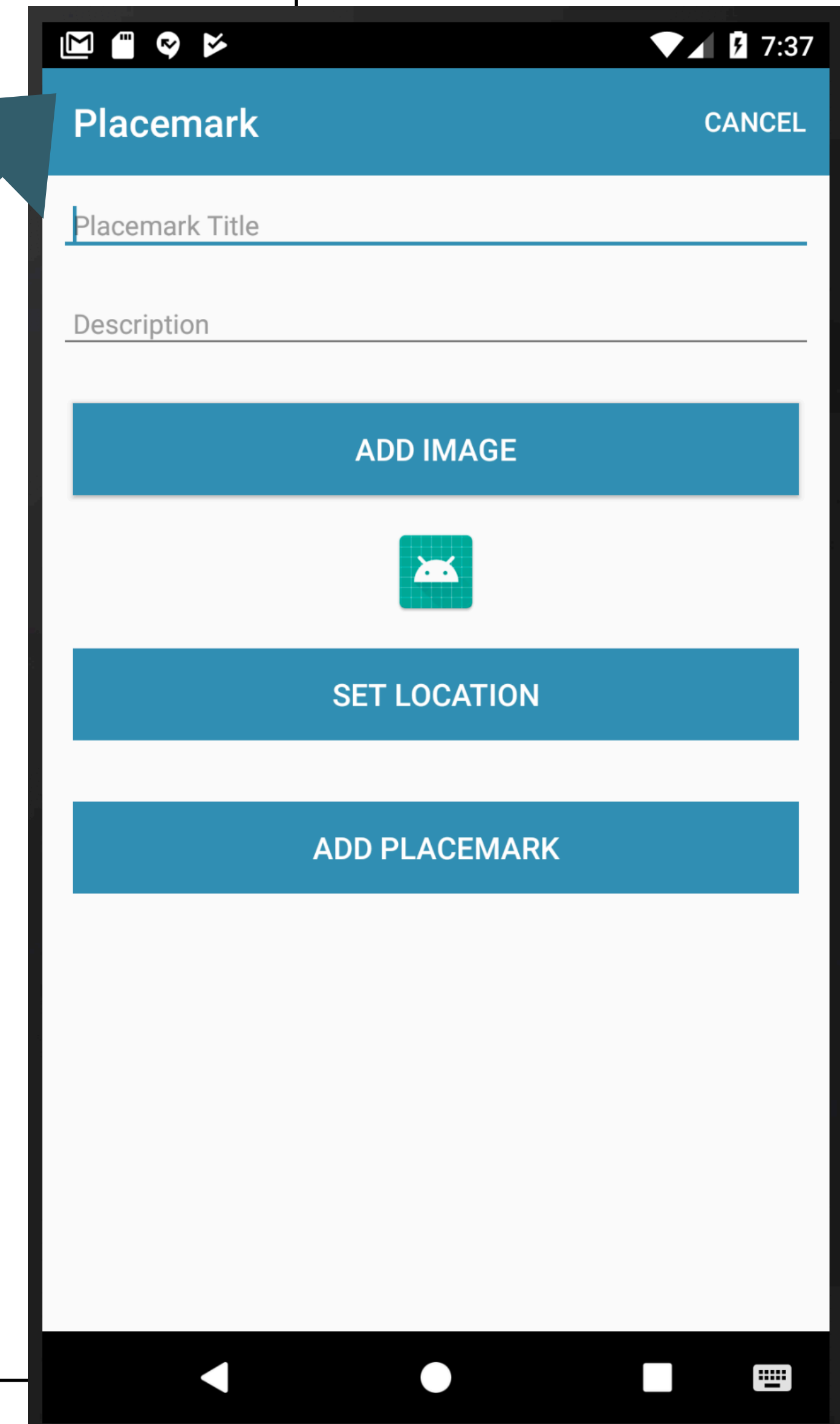
```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbarAdd"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:titleTextColor="@color/colorPrimary" />
```

```
</android.support.design.widget.AppBarLayout>
```

...

```
</RelativeLayout>
```

```
</android.support.constraint.ConstraintLayout>
```



```
...
override fun onCreate(savedInstanceState: Bundle?) {
    ...
    toolbarAdd.title = title
    setSupportActionBar(toolbarAdd)
    ...
}
...
```

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.menu_placemark, menu)
    return super.onCreateOptionsMenu(menu)
}
```

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

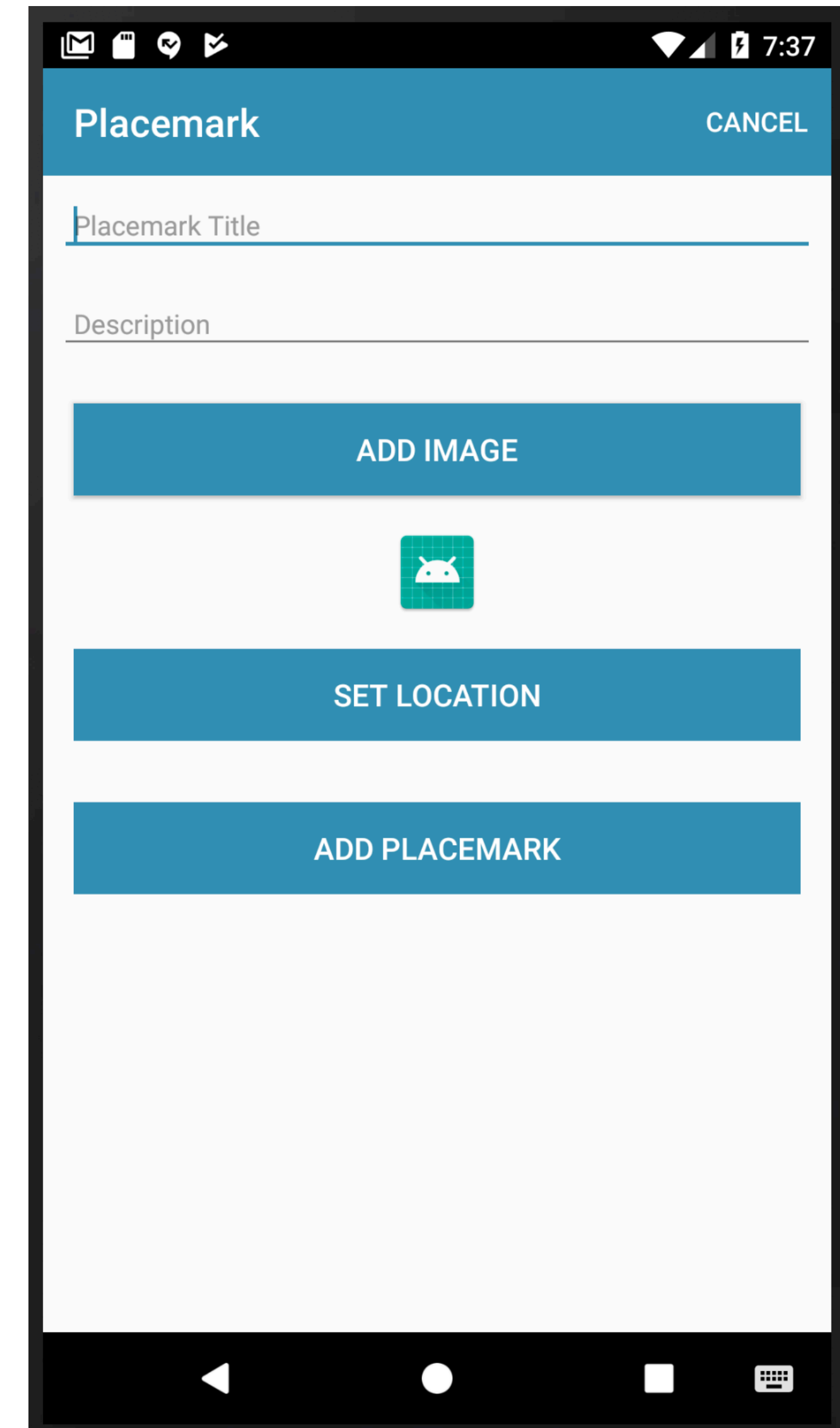
    <item
        android:id="@+id/item_cancel"
        android:title="@string/menu_cancelPlacemark"
        app:showAsAction="always"/>
</menu>
```

```
...
<string name="menu_cancelPlacemark">Cancel</string>
```

Enable the bar
Set the title

Load the Menu
Resource &
display on
toolbar

Menu Resource




Placemark CANCEL

Placemark Title

Description

ADD IMAGE



SET LOCATION

ADD PLACEMARK

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {  
    when (item?.itemId) {  
        R.id.item_cancel -> {  
            finish()  
        }  
    }  
    return super.onOptionsItemSelected(item)  
}  
...
```

```
<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto">  
    <item  
        android:id="@+id/item_cancel"  
        android:title="@string/menu_cancelPlacemark"  
        app:showAsAction="always"/>  
</menu>
```

```
class PlacemarkActivity : AppCompatActivity(), AnkoLogger {  
    ...  
}
```

- **AppCompatActivity** - Provides Material color themes, widget tinting, and **app bar** support to earlier devices. Use of this class requires that you use **Theme.AppCompat** themes for consistent visual presentation.

Add the app bar

<https://developer.android.com/training/appbar/>

The *app bar*, also known as the *action bar*, is one of the most important design elements in your app's activities, because it provides a visual structure and interactive elements that are familiar to users. Using the app bar makes your app consistent with other Android apps, allowing users to quickly understand how to operate your app and have a great experience. The key functions of the app bar are as follows:

- A dedicated space for giving your app an identity and indicating the user's location in the app.
- Access to important actions in a predictable way, such as search.
- Support for navigation and view switching (with tabs or drop-down lists).



This class describes how to use the [v7 appcompat](#) support library's `AppBar` widget as an app bar. There are other ways to implement an app bar—for example, some themes set up an `ActionBar` as an app bar by default—but using the appcompat `AppBar` makes it easy to set up an app bar that works on the widest range of devices, and also gives you room to customize your app bar later on as your app develops.

Lessons

Set up the app bar

Learn how to add a `AppBar` widget to your activity, and set it as the activity's app bar.

Add and handle actions

Learn how to add actions to the app bar and its overflow menu, and how to respond when users choose those actions.

Add an up action

Learn how to add an *Up* button to your app bar, so users can navigate back to the app's home screen.

Use action views and action providers

Learn how to use these widgets to provide advanced functionality in your app bar.