A290/A590 – Meeting 7 Guide
Building a "Real App" — Part 1

Goal: We want to create something a bit more realistic and practical, and try to learn a variety of things along the way. We will try to build an Android version of the Sudoku game, **but we will not actually complete it**. If you are unfamiliar with the game, that is fine, you will learn about it as we progress.

The important thing to know is that we will be trying to build some version of a grid that will be 9x9 (81 squares) and allow the user to "place" numerical values 1-9 in any of the open squares or tiles. Some will already have values present, which will be different every time you start a New Game.

I. Getting Started.

We will create a new Project. Name the Application "A290 First Real Application". Accept the project and package names, making sure to use "edu.indiana" in the package name. REMEMBER: Depending on where you create this, you need to be sure your Minimum API is "23" and your Target is API 23, to match what is in our lab, PV151.

Let's see if we can set a new icon for this. I will have some samples, but if you choose to follow along, you need to find your own set of free icons (JPG or PNG format only) and make sure they are really "free."

Name your Activity "FirstRealAppMainActivity" and name your Layout "activity_first_real_app_main." These will be important, as we add more activities and more layouts. We need to be able to distinguish each one with an easy to remember name. Set the Title to "A290 Sudoku" or A590 Sudoku" as is appropriate.

Click "Finish" and open your new application.

II. Setting Up the Layout and Creating some Controls.

Because we want to keep things simple and learn a bit more about how to create and customize our visible items using Text mark-up instead of the Design view, we want to change our layout. Make sure both FirstRealAppMainActivity.java and activity_first_real_app_main.xml are open. First, look at your layout file in Text view. It should confirm you have a "Constraint Layout." We will stick with this for the time being.

Next, we want to replace the default string and its reference, and then add some controls.

Find "@string/hello_world" and change it to "@string/MainTitle". You could choose a different naming scheme, like "main_title" or something similar. This is up to you, but choose a strategy and **stick with it**.

Open up your strings.xml in the "res/values" folder and change the string resource name to match your new name (MainTitle for this example) and change the string to "Welcome to my Sudoku App". Save All to clear any warnings. You may also wish to change they textAppearance and keep the TextView centered but move it to the top of the screen.

Next, we want to add 4 buttons. We will do this with XML directly instead of the Graphical Interface. As we go along, we will get a variety of warnings until everything is done, so don't worry about that.

We will create 4 buttons. Now one thing we can do, if the buttons are going to have similar properties, is create one, copy the XML, and the paste 3 more copies and follow that with the modifications to make them unique.

The first button will be added with the following:

```
<Button
        android:id="@+id/btn_Continue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/ContinueButtonLabel"
        />
```

Remember that you need to be sure every instance of this element is properly terminated with the "/>" at the end. Also notice the naming convention I have selected for both the button instance name (btn_Continue) and the string reference (ContinueButtonLabel). You are free to choose, but be consistent.

If you wish, switch to your strings.xml and add this new string and its reference so you know that is also done.

Now, we can select this entire block of XML, copy it, and paste it 3 times to get the "boilerplate" for our other 3 buttons. They will be named "btn_New" "btn_About", and "btn_Exit." The string resources will be similarly named as with our first button.

NOTE: For the actual string for "NewButtonLabel" be sure it says "New Game" or something very similar.

Once you think you have all this complete, switch to Design View and see how it looks. You may wish to "reorder" the buttons, as I have, once you see how they look. Let's go ahead and test this in the emulator, just to be sure we know how it looks. Not that great, because all the buttons are on top of each other and all are the full width of the screen. We need to make choices about size and location. I have changed my text to App Compatible Title, the button width to 150dp and I have horizontally aligned the buttons to the screen and vertically aligned them with each other.

III. Creating and Adding "Color" Resources.

Assuming that all looks OK, we want to use another resource file that will allow us to control and use custom colors. If we use the file correctly, Android Studio will automatically add these colors it to our project. The name of the file is **colors.xml**. Open the file and it should be in XML/Text view. Once you have done that, try adding the following:

```
<color name="background">#3500ffff</color>
```

Let's try applying this to our layout file in Text view. We need to make some additions to our Constraint Layout. We want to add the following 2 lines:

```
android:padding="30dp"
android:background="@color/background"
```

As you type, notice that you are shown the options. Make sure you are clear why we chose what we did. Feel free to try other options on your own and see their impact by switching to Design View. Go ahead and do that now to be sure you are clear on the consequences of these changes.

Let's also try to modify the MainTitle by adding to new properties. In the TextView element, add:

```
android:layout_marginBottom="25dp"
android:textSize="24.5sp"
```

and you may wish to also add:

```
android:textAlignment="center"
```

Do you know what these will do, before you look at Design view to see? Check it out. **NOTE:** Hopefully you see these either alter our layout in a negative way or have no immediately visible impact. So, we will simply "Undo" them so we are back where we were before we tried to alter the title.

Hopefully, you finally see something that is looking a bit more like a real application, even though all we have done so far is layout. Maybe it's time to start making it actually "do" something.

IV. Creating all the Components for the "About" Box.

We want our "About" button to react as most, and display some information about our game. This means some version of a Dialog. Since we want to have a new "box" appear (we will look at various options as we proceed), we will actually create a new layout and a new Activity.

Let's start with the layout file.

Make sure you have highlighted the “res/layout” folder and the do File/New/XML/Layout XML File. Name this file activity_first_real_app_about, and replace the “Linear Layout” **Root Tag** with “ScrollView” and click Finish.

What appears looks like a HomeScreen, so we need to make some layout changes right away. First, confirm the “layout” in Text view is ScrollView. Since this is an “About” box, it will need to have a TextView element. Add that to your file, so the entire thing looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/AboutScrollView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp" >

    <TextView
        android:id="@+id/txt_AboutContent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/AboutText"
        />

</ScrollView>
```

Notice that I have given this “ScrollView” a unique name: AboutScrollView. Once this is done, we want to create two string resources, even though we won’t use one of them immediately.

Create one for “AboutText” and make it say something like the following:

```
Sudoku is a logic-based number placement puzzle. Starting with a partially completed 9x9 grid, the
 objective of the puzzle is to fill the remaining &quot;open&quot; tiles or spaces in the grid so
that each row, each column, and each of the 3x3 boxes (also called blocks) contains the digits 1
to 9 exactly once. HAVE FUN!
```

If you enter all this text in Resource view, it will automatically convert the “” around “open” to the proper XML markup for quotation marks. That is what you see in my sample, copied from XML view after I entered the string in Resource view.

Also create another string resource, name it AboutTitle and make the string A290 Sudoku - About My First Real Application.

Have a look at what your “About” page looks like before we move on.

V. Adding “Activity” to our “About” box.

Now that we have most of our layout done for our “About” box, we need to create the activity that will make it “do” something.

We need to create a new activity, which means a new *.java file in our “src” folder. Make sure your “app/”java/[activity]” folder is highlighted, and use File/New/Java Class to create our new activity. Be sure the new class is in the same folder as your MainActivity.java and name it “FirstRealAppAboutActivity.” The new file should open automatically, and have nothing more than this for content:

```
package edu.indiana.a290firstrealapplication;

public class FirstRealAppAboutActivity {

}
```

To make this activity do something, we need to add some libraries and then use our new class to override onCreate and setContentView. Your edited version should look like this:

```
package edu.indiana.a290firstrealapplication;

import android.app.Activity;
import android.os.Bundle;

public class FirstRealAppAboutActivity extends Activity {
	@Override
	protected void onCreate(Bundle savedInstanceState){
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_first_real_app_about);
	}
}
```

Now we need to connect this new activity with our About Button, so we need to modify our main Activity. Open FirstRealAppMainActivity.java. We need to add some libraries, including click listener and intent. First, confirm you already have the following:

```
import android.view.Menu;
import android.view.View;
```

and then add these two:

```
import android.content.Intent;
import android.view.View.OnClickListener;
```

We need to setup our ClickListener for the About button, but let's go ahead and do this for all 4 buttons, since we know we will need them. All 4 will be part of the "onCreate" method at the beginning of our file. Don't worry about warnings that appear as you add them.

The code for our Continue Button will look like this, and will generate a warning until we do some additional work. Note that the first and last "ContinueButton" refers to the click listener while the middle one (in blue below) refers to the actual button resource.

```
	View ContinueButton = findViewById(R.id.btn_Continue);
	ContinueButton.setOnClickListener(this);
```

Repeat this for the other 3 buttons, substituting the appropriate names.

Now we need to do two things. First, we need to further modify our actual main activity class, so it can implement our onClickListener requirements, and we need to add the onClick method from OnClickListener.

To do the first, modify your public class declaration so it looks like this:

```
public class FirstRealAppMainActivity extends AppCompatActivity implements OnClickListener  {
```

We finally have to explicitly add an "Intent." So what is an intent and what does it do for us? This is what is required to allow any new activity to "use" the resources of the thread of execution or process being used by the main activity. In other words, "intent" allows us to sort of piggy-back our new activity on top of the main one. This is all related to the onPause, onResume and other states we have discussed previously. Although the words sound odd, we can only have one "active" GUI-based activity, most of the time, and intent is what allows us to control when a new activity is enabled/created. There is a bit more to it, however, because we need to be aware of the option of "service" as opposed to "intent."

A service differs from an intent because the service will be started with its own process or thread of execution and its own resources. This makes it possible for a service to be running in the background while each different GUI-based **activity** is being created, paused, resumed, or destroyed.

The method to create this new activity is added like this:

```
public void onClick(View v) {
```

```
            switch (v.getId()) {
            case R.id.AboutButton:
            Intent i = new Intent(this, FirstRealAppAboutActivity.class);
            startActivity(i);
            break;
            /*More buttons will/can go here as needed. I may have already added
            the basic parts to my version*/
            }
      }
```

Finally, our application will not be able to ”use” this activity unless we fully integrate it into the package, and that means we have to declare it in our AndroidManifest.xml file.

Open it by double-clicking and get Text view. You can already see our main activity is declared here and you can even see some of its properties. To add our About activity, we need to create a new <activity> element and then identify the class file that it will use.

```
      <activity
android:name="edu.indiana.jwhitmer.a290firstrealapplication.FirstRealAppAboutActivity"
            android:label="@string/AboutTitle" >
      </activity>
```

Because the package name is already declared elsewhere, we could get by with just `android:name=".FirstRealAppAboutActivity"` if we wished.

Save all, start the emulator, and let’s test what we have so far.

If you don’t think the “About” dialog looks all that much like a dialog, do you remember what we did with My First Activity? Let’s change the theme for our About dialog. If you recall, we do that in the AndroidManifest.xml file as well, by adding `android:theme="@android:style/Theme.Dialog"` to our set of properties for the FirstRealAppAboutActivity. Make this addition, save all, and run again in the emulator. What do you think now?

VI. What’s Next?

So next time, we will work on our Settings/Preferences menu, modify/configure how it is displayed, and then move on to setting up the activity and related resources to make our “New Game” button active. From there, we will need to move on to using some of the 2D options in Android to actually setup the 81 tiles for the game “space” and make things work.

BE SURE TO SAVE YOUR PACKAGE FOLDER TO YOUR THUMB DRIVE BEFORE YOU DISMOUNT IT.