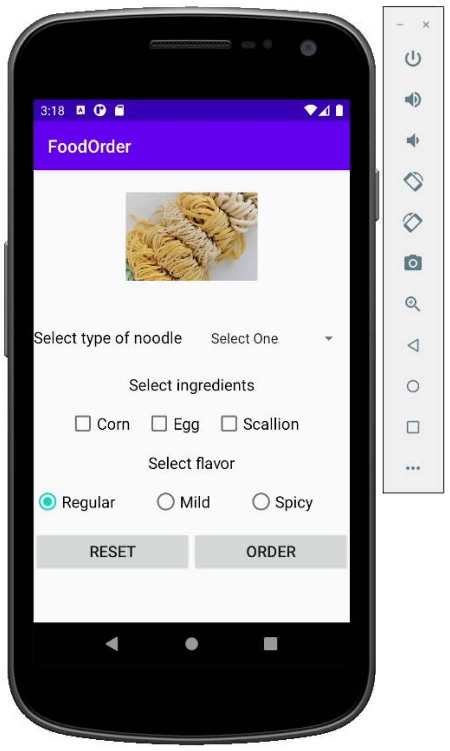
Android Studio Tutorial - Part 3

Welcome!

In this tutorial, you will learn how to build and run an interactive Food Ordering Android App in the Java programming language.

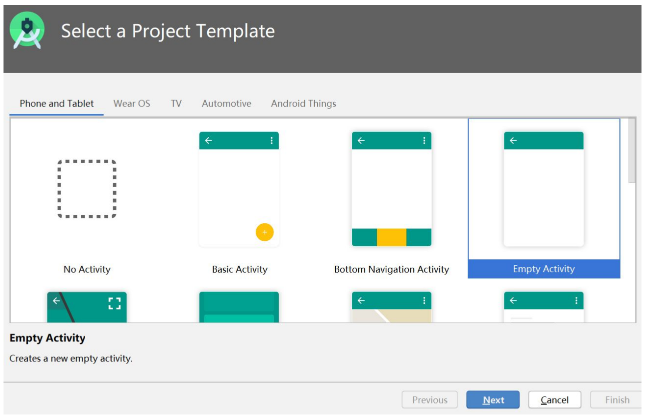
You will learn:

- How to add and set layouts, TextViews, ImageViews.
- How to add interactive Spinner, CheckBox and Radio Button/Group.
- How to zoom in and zoom out images.
- How to send email in background.



Create a new project

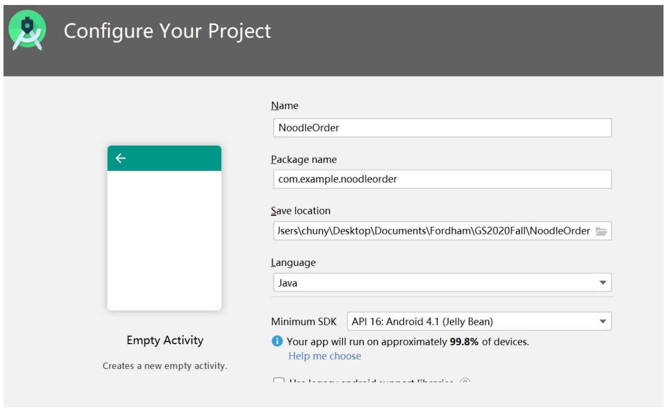
1. Follow the tutorial part 1 - Hello World App
2. Run your android studio app. Create a new project, select an empty activity template, name it as FoodOrder (or whatever you like), select the programming language as Java.



Select a Project Template

Empty Activity

Creates a new empty activity.



Configure Your Project

Empty Activity

Creates a new empty activity.

Name: NoodleOrder

Package name: com.example.noodleorder

Save location: Jsers\chuny\Desktop\Documents\Fordham\GS2020Fall\NoodleOrder

Language: Java

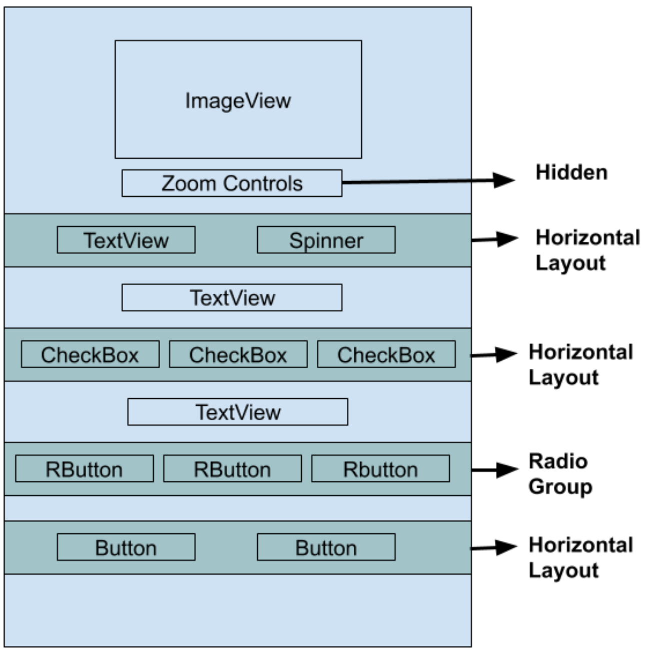
Minimum SDK: API 16: Android 4.1 (Jelly Bean)

Your app will run on approximately **99.8%** of devices.

Help me choose

Set the entire Layout and constraint

1. The following drawing shows the blueprint of the FoodOrder App. We need to build the app with ImageView, TextView, CheckBox, Radio Group, Button and extra Layout to make the page look nice.



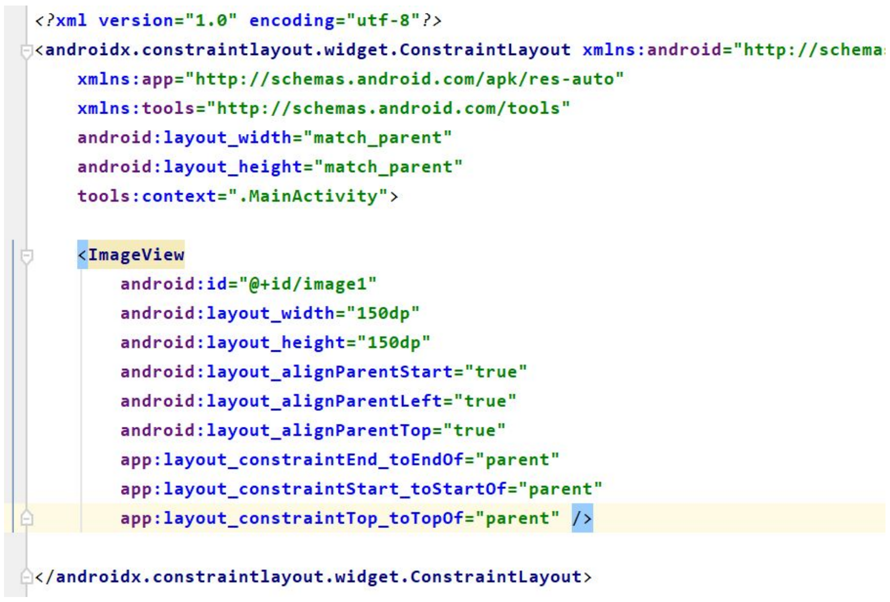
2. Go to the *activity_main.xml* and open the code interface. (or you can work with the design interface by dragging and dropping views into Component Tree and setting attributes)

3. First delete the TextView "Hello World" code and copy/paste following code to add the **ImageView**:

```
<ImageView

   android:id="@+id/image1"

   android:layout_width="150dp"

   android:layout_height="150dp"

   android:layout_alignParentStart="true"

   android:layout_alignParentLeft="true"

   android:layout_alignParentTop="true"

   app:layout_constraintEnd_toEndOf="parent"

   app:layout_constraintStart_toStartOf="parent"

   app:layout_constraintTop_toTopOf="parent" />
```

What's in the code: set the id to be image1, set the image size, set the layout position and constraint.

It should looks something like that:



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schema
    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=".MainActivity">

    <ImageView
        android:id="@+id/image1"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

For people who work with the design view:

- drag and drop the TextView component to the design interface.
- Set following attributes according to the code: id, layout_width, layout_height, layout_alignParentStart, layout_alignParentLeft, layout_alignParentTop
- Set the constraint for top, end and start.

4. Go to the design view, now you will see an empty ImageView.

5. Now, let's add a **zoom control** under the ImageView. Copy and paste the following code:

```
<ZoomControls
   android:id="@+id/zoom_controls"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_margin="10dp"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toBottomOf="@+id/image1" />
```

What's in the code: set the id to be zoom_controls, set the layout size, position and constraint (to be the bottom of the ImageView we just added).

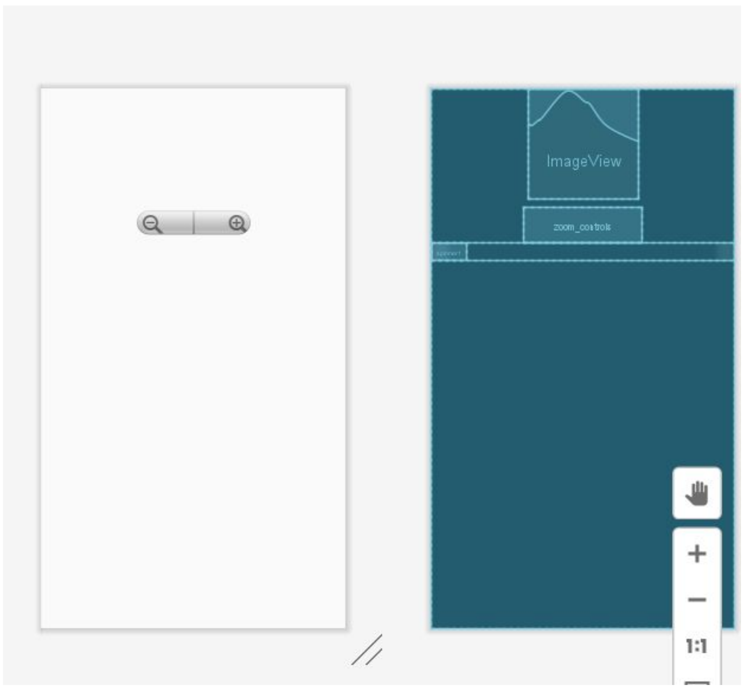
6. Let's add the **Horizontal layout** with the **TextView** and **Spinner inside**:

```
<LinearLayout
   android:id="@+id/layout1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toBottomOf="@+id/zoom_controls">
   <TextView
       android:id="@+id/text1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />
   <Spinner
       android:id="@+id/spinner1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />
</LinearLayout>
```

What's in the code:

- Set the layout id to be layout1
- Set the layout width to match_parent so we would have a nice and wide layout. Set the layout height to wrap_content.
- Set the layout to be horizontal.
- Set the layout be positioned right below the Zoom control we just added.
- Inside the layout, we added one TextView (id: text1) and one Spinner (id: spinner1) which we will continue formatting in the following section. Now, let's just leave it that way. (Make sure there is extra indentation for the TextView and Spinner)

7. Right now, you should have something like this in your design view.



8. For people who want to work with the design view only, keep in mind that everything you edited in the Attributes section would also be reflected in the code section. So in the end, you need to make sure the code part is exactly the same as what is shown above.

9. Let's continue and add more elements into the view: **TextView**

```
<TextView
    android:id="@+id/text2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/layout1"/>
```

What's in the code:

- Set the new TextView Id as text2, set the size
- Set the TextView be positioned right below the Layout1

10. Now add another **LinearLayout** with **checkboxes** inside.

```
<LinearLayout
    android:id="@+id/layout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintTop_toBottomOf="@+id/text2">
    <CheckBox
        android:id="@+id/cb1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBox
        android:id="@+id/cb2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBox
        android:id="@+id/cb3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

What's in the code:

- Set the layout id to be layout2
- Set the layout width to match_parent so we would have a nice and wide layout. Set the layout height to wrap_content.
- Set the layout to be horizontal.
- Set the layout be positioned right below the TextView we just added.
- Inside the layout, we added three checkboxes which we will continue formatting in the following section. Now, let's just leave it that way. (Make sure there is extra indentation for each checkbox)

11. Add another **TextView**:

```
<TextView
    android:id="@+id/text3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/layout2" />
```

What's in the code:

- Set the TextView id, size and position.

12. Add **RadioGroup** and **Radio Buttons**:

```
<RadioGroup
    android:id="@+id/rg"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text3">
    <RadioButton
        android:id="@+id/rb1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton
        android:id="@+id/rb2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton
        android:id="@+id/rb3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RadioGroup>
```

What's in the code:

- Treat the Radio Group similarly as the layout. Set the Radio Group id.
- Set the Radio Group width to match_parent so we would have a nice and wide layout. Set the height to wrap_content.
- Set the Radio Group to be horizontal.
- Set the Radio Group be positioned right below the TextView we just added.
- Inside the Radio Group, we added three Radio Buttons which we will continue formatting in the following section. Now, let's just leave it that way. (Make sure there is extra indentation for each Radio Button)

13. Add **LinearLayout** with two **Buttons**.

```
<LinearLayout
    android:id="@+id/layout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/rg">
    <Button
        android:id="@+id/resetButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/orderButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

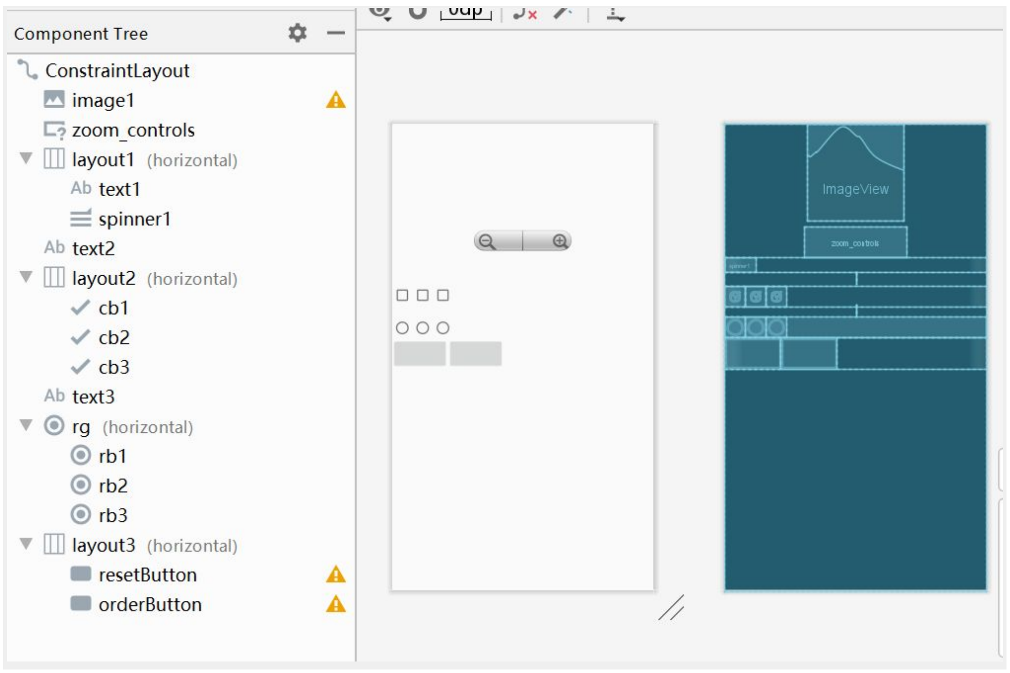
What's in the code:

- Set the layout id to be layout3

- Set the layout width to match_parent so we would have a nice and wide layout. Set the layout height to wrap_content.
- Set the layout to be horizontal.
- Set the layout be positioned right below the Radio Group we just added.
- Inside the layout, we added two buttons which we will continue formatting in the following section. Now, let's just leave it that way. (Make sure there is extra indentation for each button)

Now you would have something looks like this:

(Please check your Component Tree to see if you have the same layout as shown below)



```
Component Tree
ConstraintLayout
  image1
  zoom_controls
  layout1 (horizontal)
    text1
    spinner1
  text2
  layout2 (horizontal)
    cb1
    cb2
    cb3
  text3
  rg (horizontal)
    rb1
    rb2
    rb3
  layout3 (horizontal)
    resetButton
    orderButton
```

Format and set string values

Now, we have a good overall layout with all elements we want. Let's start to do the formatting as well as string setting for each element.

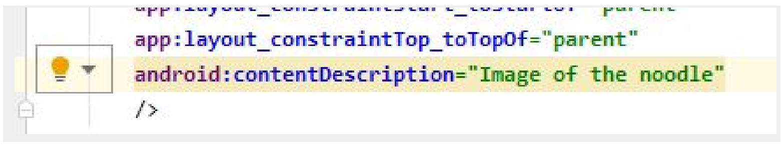
1. Margin Setting: In order to have a nice layout, we need to set the margin for each element and make sure all of the elements would have a nice spread out on the page.
2. Add image description and other necessary features.
3. Define text string for each element and set proper text size & color.
4. Again, you can work on the design view as well as the code view. This tutorial will only show the code view. However, it is fairly easy to change all attributes setting in the Design view as well. (Like we did in the Tutorial 1)

Please add following lines to different element:

- image1:

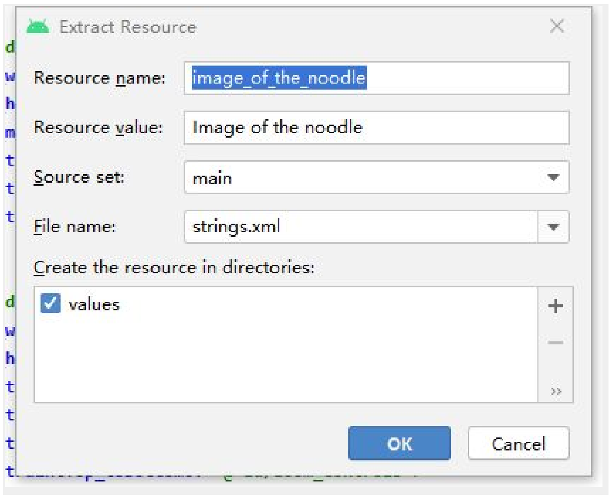
```
android:contentDescription="Image of the noodle"
```

Here is the trick to quickly create string resources. Instead of going to res>values>strings.xml to create new strings, we could type the target value in the activity_main.xml file, click the warning sign and select “Extract string resource…”.



```
app:layout_constraintTop_toTopOf="parent"
android:contentDescription="Image of the noodle"
/>
```

A new window would pop up showing auto-generated resource name and the typed string value. You can change the resource name if you want, but I will just leave it that way and click ok.



Extract Resource

Resource name: image_of_the_noodle

Resource value: Image of the noodle

Source set: main

File name: strings.xml

Create the resource in directories:

☑ values

- zoom_controls:

```
android:layout_margin="10dp"
```

- layout1:

```
android:layout_marginTop="16dp"

android:layout_marginBottom="16dp"
```

- text1:

```
android:layout_weight="1"

android:text="Select type of noodle"

android:textColor="@android:color/black"

android:textSize="18sp"
```

Following the previous instruction and extracting the string resource.

Set the layout_weight for this TextView and Spinner both to be 1. It would position those two elements with the exact same size within this horizontal layout.

- spinner1:

```
android:layout_weight="1"

android:textSize="18sp"
```

- text2:

```
android:layout_marginTop="16dp"
```

```
android:layout_marginBottom="16dp"
android:text="Select ingredients"
android:textColor="@android:color/black"
android:textSize="18sp"
```

Following the previous instruction and extracting the string resource.

- layout2:

```
android:layout_marginTop="16dp"
android:layout_marginBottom="16dp"
```

- cb1:

```
android:layout_weight="1"
android:text="corn"
android:textSize="18sp"
android:layout_marginLeft="40dp"
```

Following the previous instruction and extracting the string resource.

- cb2:

```
android:layout_weight="1"
android:text="egg"
android:textSize="18sp"
```

Following the previous instruction and extracting the string resource.

- cb3:

```
android:layout_weight="1"
android:text="scallion"
android:textSize="18sp"
android:layout_marginRight="40dp"
```

Following the previous instruction and extracting the string resource.

- text3

```
android:layout_marginTop="16dp"
android:layout_marginBottom="16dp"
android:text="Select flavor"
android:textColor="@android:color/black"
android:textSize="18sp"
```

Following the previous instruction and extracting the string resource.

- rg:

```
android:layout_marginTop="16dp"
android:layout_marginBottom="16dp"
```

- rb1:

```
android:layout_weight="1"
```

```
android:checked="true"
android:text="Regular"
android:textSize="18sp"
```

The checked="true" would set the "Regular" as default value.

Following the previous instruction and extracting the string resource.

- rb2:

```
android:layout_weight="1"
android:text="Mild"
android:textSize="18sp"
```

Following the previous instruction and extracting the string resource.

- rb3:

```
android:layout_weight="1"
android:text="Spicy"
android:textSize="18sp"
```

Following the previous instruction and extracting the string resource.

- layout3:

```
android:layout_marginTop="16dp"
android:layout_marginBottom="16dp"
```

- resetButton:

```
android:layout_weight="1"
android:text="Reset"
android:textSize="18sp"
```

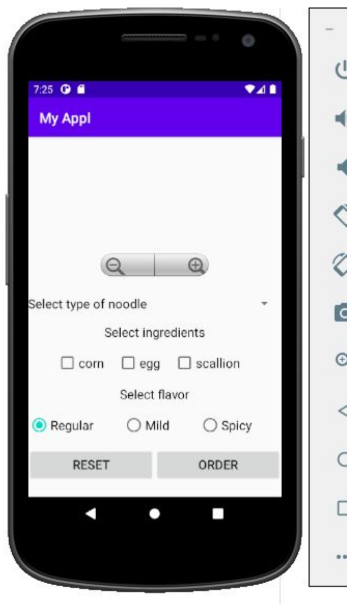
Following the previous instruction and extracting the string resource.

- orderButton:

```
android:layout_weight="1"
android:text="Order"
android:textSize="18sp"
```

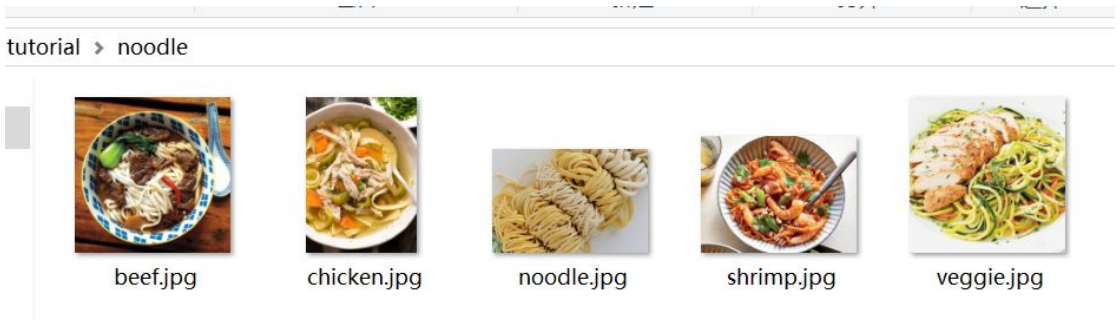
Following the previous instruction and extracting the string resource.

Now run the app, you should have something similar to this:

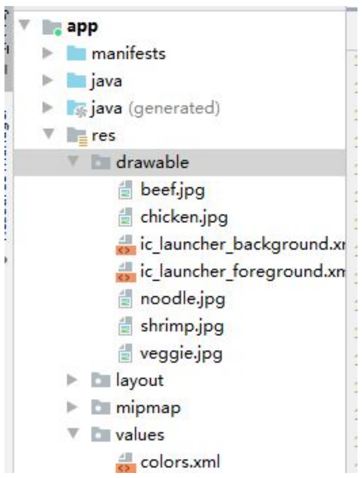


ImageView and Zoom control

1. First, let's download 5 pictures online (1. noodle in general 2. Beef noodle 3. Chicken noodle 4. Shrimp 5. Veggie) and name them properly.
2. Here is what I have:



3. Now, let's copy and paste those 5 pictures into the res>drawable folder.



4. Now, we have the picture to work.

5. Let's go ahead and set the Zoom control.
6. Go to the MainActivity.java, now we need to write some code and define the logic of the zoom control.
7. Within the class MainActivity and before any other function, lets define two variables:

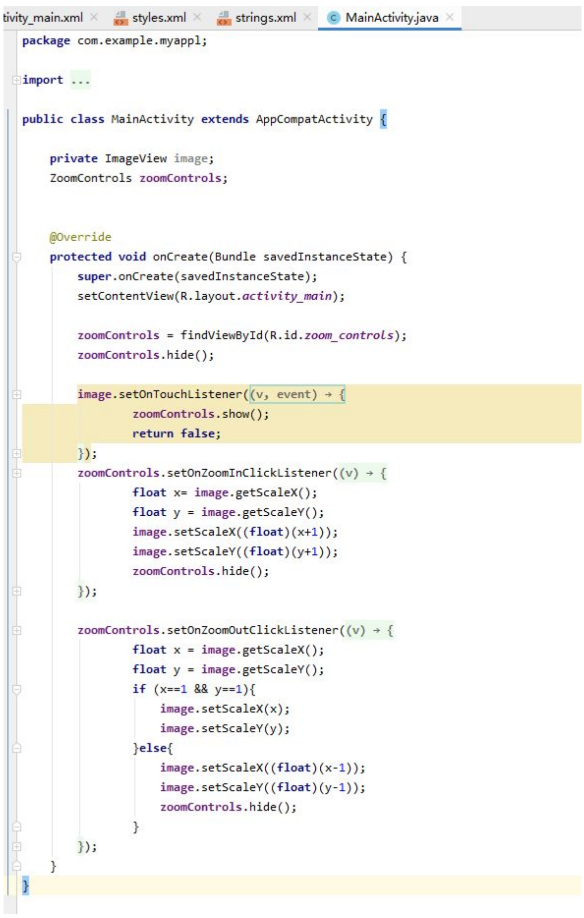
```
private ImageView image;

ZoomControls zoomControls;
```

8. Then with the onCreate function, copy and paste the following codes:

```
zoomControls = findViewById(R.id.zoom_controls);

zoomControls.hide();

image.setOnTouchListener(new View.OnTouchListener(){

   @Override

   public boolean onTouch(View v, MotionEvent event) {

        zoomControls.show();

        return false;

   }

});

zoomControls.setOnZoomInClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View v) {

        float x= image.getScaleX();

        float y = image.getScaleY();

        image.setScaleX((float)(x+1));

        image.setScaleY((float)(y+1));

        zoomControls.hide();

   }

});

zoomControls.setOnZoomOutClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View v) {

        float x = image.getScaleX();

        float y = image.getScaleY();

        if (x==1 && y==1){

            image.setScaleX(x);

            image.setScaleY(y);

        }else{

            image.setScaleX((float)(x-1));

            image.setScaleY((float)(y-1));

            zoomControls.hide();

        }
```

```
    }
});
```

9. Now you should have something looks like this:



```java
package com.example.myappl;

import ...

public class MainActivity extends AppCompatActivity {

    private ImageView image;
    ZoomControls zoomControls;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        zoomControls = findViewById(R.id.zoom_controls);
        zoomControls.hide();

        image.setOnTouchListener((v, event) → {
                zoomControls.show();
                return false;
        });
        zoomControls.setOnZoomInClickListener((v) → {
                float x= image.getScaleX();
                float y = image.getScaleY();
                image.setScaleX((float)(x+1));
                image.setScaleY((float)(y+1));
                zoomControls.hide();
        });

        zoomControls.setOnZoomOutClickListener((v) → {
                float x = image.getScaleX();
                float y = image.getScaleY();
                if (x==1 && y==1){
                    image.setScaleX(x);
                    image.setScaleY(y);
                }else{
                    image.setScaleX((float)(x-1));
                    image.setScaleY((float)(y-1));
                    zoomControls.hide();
                }
        });
    }
}
```

10. Do not run your app now. It will crash.

Interactive Spinner

1. Let's define the logic of Spinner.

2. What we want to show is: when a customer uses the dropdown list and selects the noodle type, the picture in the TextView would also be changed accordingly.

3. There are two steps we need to do: define a string array containing all string variables, set the spinner

4. Define the string array. Go to res>values>strings.xml. copy and paste following:

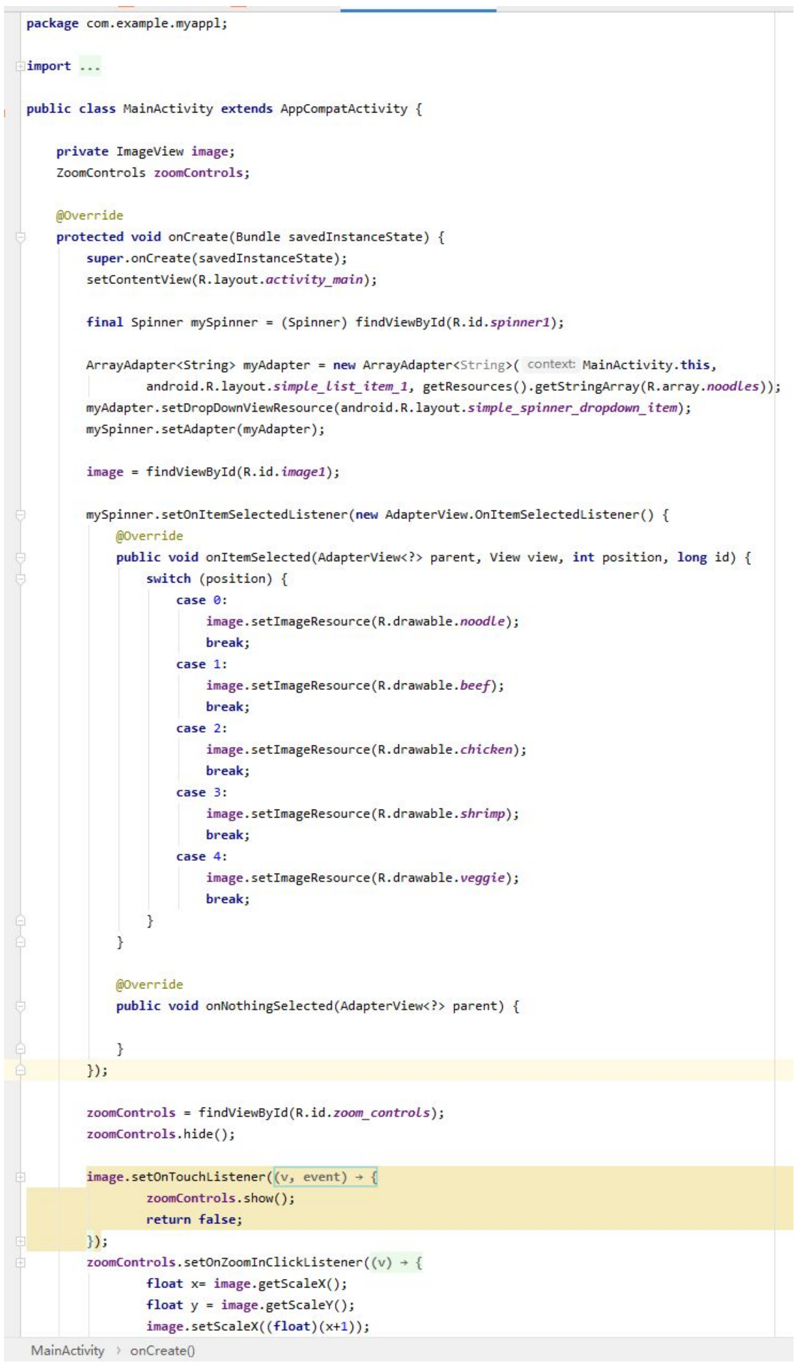
```xml
<string-array name="noodles">
   <item>Select One</item>
   <item>Beef</item>
   <item>Chicken</item>
   <item>Shrimp</item>
   <item>Veggie</item>
</string-array>
```

5. Now, Go to the MainActivity.java. Before the ZoomControl section, copy and paste the following items.

```
final Spinner mySpinner = (Spinner) findViewById(R.id.spinner1);

ArrayAdapter<String> myAdapter = new ArrayAdapter<String>(MainActivity.this,

                                          android.R.layout.simple_list_item_1,
getResources().getStringArray(R.array.noodles));

myAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);

mySpinner.setAdapter(myAdapter);


image = findViewById(R.id.image1);


mySpinner.setOnItemSelectedListener(new  AdapterView.OnItemSelectedListener()
{

   @Override

   public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) {

       switch (position) {

           case 0:

               image.setImageResource(R.drawable.noodle);

               break;

           case 1:

               image.setImageResource(R.drawable.beef);

               break;

           case 2:

               image.setImageResource(R.drawable.chicken);

               break;

           case 3:

               image.setImageResource(R.drawable.shrimp);

               break;

           case 4:

               image.setImageResource(R.drawable.veggie);

               break;

       }

   }

   @Override

   public void onNothingSelected(AdapterView<?> parent) {

   }

});
```

6. Make sure the spinner section is before the zoom control section, otherwise, the app will crash.

7. You will now have something looks like this:



```
package com.example.myapp1;

import ...

public class MainActivity extends AppCompatActivity {

    private ImageView image;
    ZoomControls zoomControls;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final Spinner mySpinner = (Spinner) findViewById(R.id.spinner1);

        ArrayAdapter<String> myAdapter = new ArrayAdapter<String>( context: MainActivity.this,
                android.R.layout.simple_list_item_1, getResources().getStringArray(R.array.noodles));
        myAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        mySpinner.setAdapter(myAdapter);

        image = findViewById(R.id.image1);

        mySpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                switch (position) {
                    case 0:
                        image.setImageResource(R.drawable.noodle);
                        break;
                    case 1:
                        image.setImageResource(R.drawable.beef);
                        break;
                    case 2:
                        image.setImageResource(R.drawable.chicken);
                        break;
                    case 3:
                        image.setImageResource(R.drawable.shrimp);
                        break;
                    case 4:
                        image.setImageResource(R.drawable.veggie);
                        break;
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        zoomControls = findViewById(R.id.zoom_controls);
        zoomControls.hide();

        image.setOnTouchListener((v, event) -> {
                zoomControls.show();
                return false;
        });
        zoomControls.setOnZoomInClickListener((v) -> {
                float x= image.getScaleX();
                float y = image.getScaleY();
                image.setScaleX((float)(x+1));
```

8. Run your app and try to use the zoom control as well as the spinner.

Checkbox

1. For checkbox, we want to show the user a toast message when the user checks or unchecks the box.

2. Stay in the MainActivity.java. Before the main function, let's define 3 variables.

```
CheckBox corn;
CheckBox egg;
CheckBox scallion;
```

3. After the Zoom section, let’s copy and paste the following:

```
corn = (CheckBox) findViewById(R.id.cb1);
egg = (CheckBox) findViewById(R.id.cb2);
scallion = (CheckBox) findViewById(R.id.cb3);


corn.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       if(corn.isChecked()){
           Toast.makeText(MainActivity.this, "Corn is checked",
                   Toast.LENGTH_SHORT).show();
       }else {
           Toast.makeText(MainActivity.this,"Corn is unchecked",
                   Toast.LENGTH_SHORT).show();
       }
   }
});
egg.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       if(egg.isChecked()){
           Toast.makeText(MainActivity.this, "Egg is checked",
                   Toast.LENGTH_SHORT).show();
       }else {
           Toast.makeText(MainActivity.this,"Egg is unchecked",
                   Toast.LENGTH_SHORT).show();
       }
   }
});
scallion.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       if(scallion.isChecked()){
           Toast.makeText(MainActivity.this, "Scallion is checked",
                   Toast.LENGTH_SHORT).show();
       }else {
           Toast.makeText(MainActivity.this,"Scallion is unchecked",
                   Toast.LENGTH_SHORT).show();
       }
   }
});
```

4. Now, try to check and uncheck the box and see if you can see a toast message.

Radio Button and Radio Group

1. for now, we don't need any function set for this section.

Interactive Button

1. For the Reset Button, we want to clear all inputs.
2. Before all functions, let's define a few variables.

```
RadioGroup rg;

Button bReset,bOrder;
```

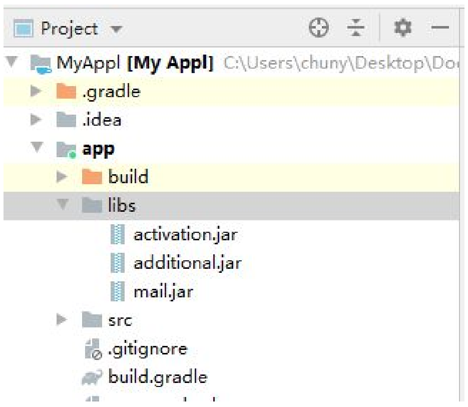
3. Then copy and paste the following and position it to the end of the checkbox section.

```
rg=findViewById(R.id.rg);

bReset=findViewById(R.id.resetButton);

bOrder=findViewById(R.id.orderButton);

bReset.setOnClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View v) {

       rg.clearCheck();

       mySpinner.setSelection(0);

       corn.setChecked(false);

       egg.setChecked(false);

       scallion.setChecked(false);

   }

});
```

4. Now, play around with your food ordering app.

Email Confirmation

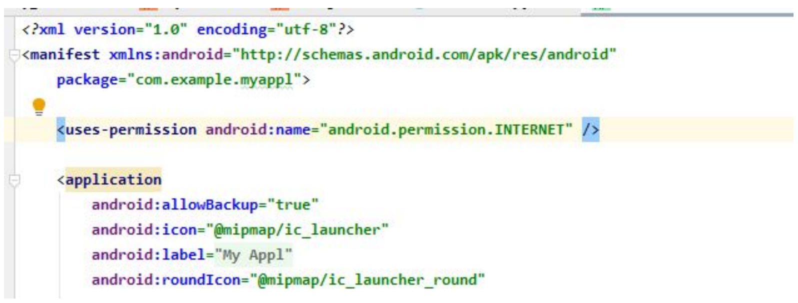
1. When we press the ORDER button, we would like to send a confirmation email to the store in the background.
2. Add three external libraries into the project(additional.jar, mail.jar, activation.jar) download. put them in the Project>app>libs folder.



3. Open app>src>main>res AndroidManifest.xml and add an internet permission to it.

   ```
   <uses-permission android:name="android.permission.INTERNET" />
   ```

   You should have something looks like this:

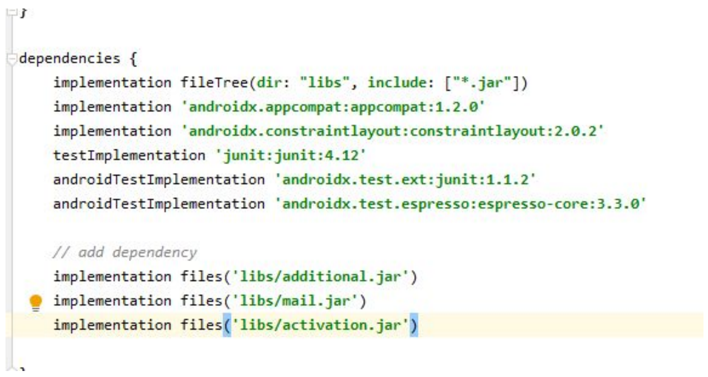


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myappl">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Appl"
        android:roundIcon="@mipmap/ic_launcher_round"
```

4. Open build.gradle(app) and add following dependency.

   ```
   implementation files('libs/additional.jar')

   implementation files('libs/mail.jar')

   implementation files('libs/activation.jar')
   ```

   You should have something looks like this:



```
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    // add dependency
    implementation files('libs/additional.jar')
    implementation files('libs/mail.jar')
    implementation files('libs/activation.jar')
```

5. Click Sync Now.

6. Go to MainActivity.java and update the following codes: (the code shows how to create a class(GMailSender.java) for authentication with your email account.)

   ```
   import android.os.Bundle;

   import android.util.Log;

   import android.view.View;
   ```

```java
import android.widget.Toast;


import androidx.appcompat.app.AppCompatActivity;


import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.Security;
import java.util.Properties;


import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;



class GMailSender extends javax.mail.Authenticator {
   private String mailhost = "smtp.gmail.com";
   private String user;
   private String password;
   private Session session;
   static {
       Security.addProvider(new JSSEProvider());
   }
   public GMailSender(String user, String password) {
       this.user = user;
       this.password = password;
       Properties props = new Properties();
       props.setProperty("mail.transport.protocol", "smtp");
       props.setProperty("mail.host", mailhost);
       props.put("mail.smtp.auth", "true");
       props.put("mail.smtp.port", "465");
       props.put("mail.smtp.socketFactory.port", "465");
       props.put("mail.smtp.socketFactory.class",
               "javax.net.ssl.SSLSocketFactory");

       props.put("mail.smtp.socketFactory.fallback", "false");

       props.setProperty("mail.smtp.quitwait", "false");

       session = Session.getDefaultInstance(props, this);

   }

   protected PasswordAuthentication getPasswordAuthentication() {

       return new PasswordAuthentication(user, password);

   }

     public  synchronized  void  sendMail(String  subject,  String  body,  String
sender, String recipients) throws Exception {

       try{

           MimeMessage message = new MimeMessage(session);

                              DataHandler   handler   =   new   DataHandler(new
ByteArrayDataSource(body.getBytes(), "text/plain"));

           message.setSender(new InternetAddress(sender));

           message.setSubject(subject);

           message.setDataHandler(handler);

           if (recipients.indexOf(',') > 0)

                              message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipients));

           else

                         message.setRecipient(Message.RecipientType.TO,  new
InternetAddress(recipients));

           Transport.send(message);

       }catch(Exception e){

       }

   }

   public class ByteArrayDataSource implements DataSource {

       private byte[] data;

       private String type;

       public ByteArrayDataSource(byte[] data, String type) {

           super();

           this.data = data;

           this.type = type;

       }

       public ByteArrayDataSource(byte[] data) {

           super();

           this.data = data;

       }

       public void setType(String type) {

           this.type = type;
       }
       public String getContentType() {
           if (type == null)
               return "application/octet-stream";
           else
               return type;
       }
       public InputStream getInputStream() throws IOException {
           return new ByteArrayInputStream(data);
       }
       public String getName() {
           return "ByteArrayDataSource";
       }
       public OutputStream getOutputStream() throws IOException {
           throw new IOException("Not Supported");
       }
   }
}
```

7. Within the public class MainActivity, add another function:

```
public void sendMail(View view) {
    try
    {
      LongOperation l=new LongOperation();
      l.execute();  //sends the email in background
      Toast.makeText(this, l.get(), Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
      Log.e("SendMail", e.getMessage(), e);
    }
  }
```

8. Now, create 2 more classes. Go to app>src>main>java right click select new to create new java class.

   Create a class(JSSEProvider.java) and update it like below code.

```
import java.security.AccessController;
import java.security.Provider;
```

```
public final class JSSEProvider extends Provider {

   public JSSEProvider() {

       super("HarmonyJSSE", 1.0, "Harmony JSSE Provider");

                                             AccessController.doPrivileged(new
java.security.PrivilegedAction<Void>() {

           public Void run() {

               put("SSLContext.TLS",

"org.apache.harmony.xnet.provider.jsse.SSLContextImpl");

               put("Alg.Alias.SSLContext.TLSv1", "TLS");

               put("KeyManagerFactory.X509",

"org.apache.harmony.xnet.provider.jsse.KeyManagerFactoryImpl");

               put("TrustManagerFactory.X509",

"org.apache.harmony.xnet.provider.jsse.TrustManagerFactoryImpl");

               return null;

           }

       });

   }

}
```

Now create a class(LongOperation.java) and update it like below code, for send mail in background. Remember to put your email address and password.

```
import android.os.AsyncTask;

import android.util.Log;

public class LongOperation extends AsyncTask<Void, Void, String> {

   @Override

   protected String doInBackground(Void... params) {

       try {

               GMailSender sender = new GMailSender("xxxx@xxx.com(your email
address as sender) ", "XXXXXX(your email password");

           sender.sendMail("New Order Arrived",

                 MainActivity.content,,"xxxx@xxx.com(sender's email address",

                   "yyyy@yyy.com(recipients email address");

       } catch (Exception e) {

           Log.e("error", e.getMessage(), e);

           return "Email Not Sent";

       }

       return "Email Sent";
```

```
        }
        @Override
        protected void onPostExecute(String result) {
            Log.e("LongOperation",result+"");
        }
        @Override
        protected void onPreExecute() {
        }
        @Override
        protected void onProgressUpdate(Void... values) {
        }
    }
```

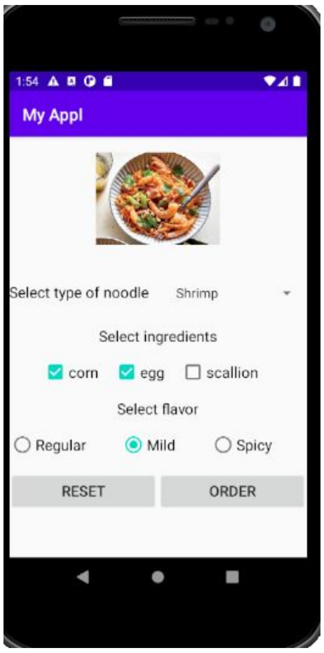
9. Open the MainActivity.xml file and update the function for the Order button.

```
    bOrder.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            noodle =mySpinner.getSelectedItem(). toString();
            if(corn.isChecked()){
                ingredients=ingredients+"corn";
            };
            if(egg.isChecked()){
                ingredients=ingredients+"egg";
            };
            if(scallion.isChecked()){
                ingredients=ingredients+"scallion";
            };

            int selectedId= rg.getCheckedRadioButtonId();
            selectedButton= (RadioButton) findViewById(selectedId);
            flavor=selectedButton.getText().toString();

            content= noodle+" " + ingredients+" " + flavor;
            try {
                LongOperation l = new LongOperation();
                l.execute();  //sends the email in background
               makeText(MainActivity.this, l.get(), LENGTH_SHORT).show();
            } catch (Exception e) {
                Log.e("SendMail", e.getMessage(), e);
            }
        }
    });
```

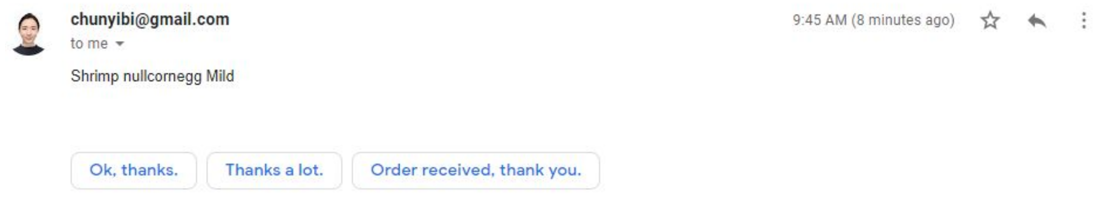
10. Scroll up in the MainActivity.java, within the MainActivity class, declare variables that we used above.

```
public static String content;
String ingredients;
String noodle;
String flavor;
RadioButton selectedButton;
```

11. Now, let's order the noodles! Run your app, and select your favourite noodle, you can reset many times as you like and then click order!



Here is the email I got.


chunyibi@gmail.com
to me
9:45 AM (8 minutes ago)

Shrimp nullcornegg Mild

Troubleshooting

While using a less secure app sending emails, Gmail accounts would tend to block it. Please do not forget to allow less secure apps. To do this, please look at : https://docs.mailshake.com/article/38-blocked-sign-in-attempt-and-less-secure-app-notification

Congratulations, Now you have your Noodle Order App!!!