中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

课程名称：移动应用开发　　任课教师： 郑贵锋

年级	2015 级	专业（方向）	软件工程（移动信息工程）（互联网）
学号	15352154	姓名	赖萍萍
电话	13609750884	Email	1684676912@qq.com
开始日期	2015/9/24	完成日期	2015/9/30

一、 实验题目

基本 UI 界面设计

二、 实现内容

实现一个 Android 应用，界面呈现如下效果：

要求：

（1）该界面为应用启动后看到的第一个界面

（2）各控件的要求如下：

① 要求只用一个 ConstraintLayout 实现整个布局；

② 标题字体大小 20sp，与顶部距离 20dp，居中；

③ 图片与标题的间距为 20dp，居中；输入框整体距屏幕右边间距 20dp，上下两栏间距 20dp，内容（包括提示内容）如图所示，内容字体大小 18sp；

④ 学号对应的 EditText 只能输入数字， 密码对应的 EditText 输入方式为密码；

⑤ 两个单选按钮整体居中，字体大小 18sp，间距 10dp，默认选中的按钮为第一个；

⑥ 两个按钮整体居中，与上方控件间距 20dp，按钮间的间距 10dp，文字大小 18sp。按钮背景框左右边框与文字间距 10dp，上下边框与文字间距 5dp，圆角半径 10dp，背景色为#3F51B5

（3）使用的布局和控件：ConstraintLayout、 TextView、 EditText、 Button、 ImageView、 RadioGroup、RadioButton

三、 课堂实验结果

（1） 实验截图

（2） 实验步骤以及关键代码

按照要实现的界面效果从上往下来完成的。

《1》首先实现标题，使用控件 TextView。

① 组件的宽度或长度能够刚好包裹住组件内的字“中山大学学生教务系统”就可以，所以设置组件的长度和宽度为“wrap_content”。

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

② 先在 app\src\main\res\values\strings.xml 中定义字符串资源，然后直接引用字

符串资源。

```
<string name="system_name">中山大学学生信息系统</string>
```

③ 设置文本字体大小 `android:textSize="20sp"`

④ 在 app\src\main\res\values\colors.xml 定义颜色资源，然后直接引用颜色资源。

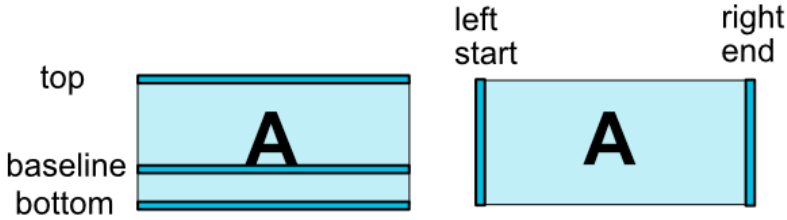
```
<color name="colorBlack">#000000</color>
```

⑤ 约束 TextView：

TextView 的左边和父控件的左边对齐，TextView 的右边和父控件的右边对齐，而控件又是 wrap_content 的，所以控件实现了居中。

```
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

Ps：这里的属性都形如：layout_constraintX_toYof= “id” ，X 代表这个子控件自身的哪条边，Y 代表约束控件的哪条边。例如：layout_constraintLeft_toLeftof= “parent”代表这个 TextView 控件的左边和父控件的左边对齐。

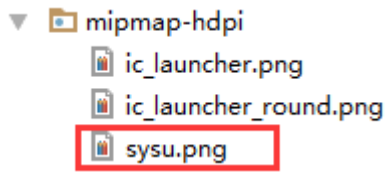


⑥ 设置标题和顶部距离为 20dp。

```
android:layout_marginTop="20dp"
```

《2》添加图片，使用控件 ImageView。

① 将图片资源 sysu.png 放到 res 文件夹中的 mipmap-hdpi 目录



② 设置控件的大小为 wrap_content，即由图片大小而定。

③ 引用图片资源。

```
android:src="@mipmap/sysu"
```

④ 设置控件的 id：（红框外即为控件的 id）

```
android:id="@+id/imageView"
```

⑤ 约束 ImageView。

左右分别都和父控件对齐，则实现了图片居中。

```
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```

该控件的顶部和标题的底部对齐。引号内是 TextView 控件（标题）的 id。

```
app:layout_constraintTop_toBottomOf="@+id/systemName"
```

⑥ 设置图片和标题间的距离为 20dp。

```
android:layout_marginTop="20dp",
```

《3》实现输入框，使用控件 EditText。

① 设置控件的长度和高度。

```
android:layout_width="285dp"
android:layout_height="wrap_content"
```

② 设置输入框内文字的颜色和大小。

```
android:textColor="@color/colorAccent"
android:textSize="18sp"
```

定义颜色资源：

```
<color name="colorAccent">#FF4081</color>
<!—粉色—>
```

③ 设置输入框输入内容的类型为数字。

```
android:inputType="number"
```

④ 设置在输入框没输入内容时显示的文字。

```
android:hint="@string/input_id"
```

定义字符串资源。

```
<string name="input_id">请输入学号</string>
```

⑤ 约束控件 EditText：

该控件的顶部和图片的底部对齐，然后设置图片和输入框之间的间隔为 20dp。

```
app:layout_constraintTop_toBottomOf="@+id/imageView"
android:layout_marginTop="20dp"
```

该控件的右边和父控件的右边对齐，然后设置输入框的右边距屏幕右间距 20dp。

```
app:layout_constraintRight_toRightOf="parent"
android:layout_marginRight="20dp"/>
```

⑥ 密码输入框的实现和学号输入框类似。

```
<EditText
    android:layout_width="285dp"
    android:layout_height="wrap_content"
    android:textColor="@color/colorAccent"
    android:textSize="18sp"
    android:inputType="textPassword"
    android:hint="@string/input_password"
    android:id="@+id/inputPassword"
    app:layout_constraintTop_toBottomOf="@+id/inputID"
    android:layout_marginTop="20dp"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginRight="20dp"/>
```

《4》实现文本“学号：”和“密码：”，使用控件 TextView。

① 设置控件的长度和宽度、文本的内容、文本的颜色、文本的大小以及控件的 id。

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/student_id"
android:textColor="@color/colorBlack"
android:textSize="18sp"
android:id="@+id/studentID"
```

```
<string name="student_id">学号:</string>
```

② 约束控件：

该控件的顶部和图片的底部对齐，然后将该控件的 baseline 和学号输入框的 baseline 对齐，那么文本学号就会和输入框的内容对齐。

```
app:layout_constraintTop_toBottomOf="@+id/imageView"
app:layout_constraintBaseline_toBaselineOf="@id/inputID"
```

该控件的左边和父控件的右边对齐，然后设置该控件左边距离屏幕 20dp。

```
app:layout_constraintLeft_toRightOf="parent"
android:layout_marginLeft="20dp"/>
```

③ 文本“密码：”实现方式和文本“学号：”类似。

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/password"
    android:textColor="@color/colorBlack"
    android:textSize="18sp"
    android:id="@+id/Password"
    app:layout_constraintTop_toBottomOf="@+id/imageView"
    app:layout_constraintBaseline_toBaselineOf="@id/inputPassword"
    app:layout_constraintLeft_toRightOf="parent"
    android:layout_marginLeft="20dp"/>
```

《5》实现平行单选按钮，使用控件 RadioGroup、RadioButton。

① 在 RadioGroup 中设置按钮 RadioButton 水平排列：

```
android:orientation="horizontal"
```

② 约束控件 RadioGroup（即约束了两个单选按钮）：

该控件的顶部和密码输入框的底部对齐，然后该控件与密码输入框的间隔为 20dp。

```
app:layout_constraintTop_toBottomOf="@+id/inputPassword"
android:layout_marginTop="20dp"
```

约束该控件的左边和父控件的左边对齐，该控件的右边和父控件的右边丢弃，并且长度和宽度都设置为 wrap_content，从而实现了两个单选按钮整体居中。

```
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent">
```

③ 设置第一个按钮“学生”：

```
<RadioButton
    android:id="@+id/id1"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/student"
    android:checked="true"
    android:textSize="18sp"
    android:layout_marginRight="10dp" />
```

设置默认选择项

设置两按钮间隔为10dp

```
<string name="student">学生</string>
```

④ 设置第二个按钮“教职工”：

```
<RadioButton
    android:id="@+id/id2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/teacher"
    android:textSize="18sp" />
```

```
<string name="teacher">教职工</string>
```

《6》实现按钮，使用控件 Button。

① 设置登录按钮的长度、宽度、id 等属性。

```
<Button
    android:id="@+id/LogIn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/log_in"
    android:textSize="18sp"
    android:textColor="@color/colorWhite"
```

```
<color name="colorWhite">#FFFFFF</color>
<!—白色—>
```

```
<string name="log_in">登录</string>
```

② 设置 button 的背景边框。

首先在 drawable 文件夹下新建一个 Drawable resource file，名称为 draw。然后将自动生成的 select 标签改为 shape。

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
```

使用 shape 的属性 corners，设置边框四个角的圆角半径为 10dp。

```
<corners android:radius="10dp"/>
```

使用 shape 的属性 padding，设置按钮内的文本与边框的距离。

```
<padding android:bottom="5dp"
        android:top="5dp"
        android:left="10dp"
        android:right="10dp"/>
```

设置 button 背景的颜色。

```
<solid android:color="@color/colorPrimary"/>
```

最后引用该背景边框资源。

```
android:background="@drawable/draw"
```

③ 约束该 button：

登录按钮的顶部和单选按钮的底部对齐，并且设置登录按钮与单选按钮之间的间隔为 20dp。

```
app:layout_constraintTop_toBottomOf="@+id/id0"
android:layout_marginTop="20dp"
```

登录按钮的右边和注册按钮的左边对齐。（在注册按钮的设置中，注册按钮的左边和登录按钮的右边对齐，从而形成链条 chain。）设置登录按钮和注册按钮之间的间隔为 10dp。

```
app:layout_constraintRight_toLeftOf="@+id/button"
android:layout_marginRight="10dp"
```

设置链条 chain 的模式 packed：它将所有 Views 打包到一起不分配多余的间隙（当然不包括通过 margin 设置多个 Views 之间的间隙），然后将整个组件组在可用的剩余位置居中：

```
app:layout_constraintHorizontal_chainStyle="packed"
```

将登录按钮的左边和父控件的左边对齐（在注册按钮中，注册按钮的右边和父控件的右边对齐，从而实现两个按钮整体居中。）

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

④ 注册按钮的实现和登录按钮类似。

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/register"
    android:textSize="18sp"
    android:textColor="@color/colorWhite"
    android:background="@drawable/draw"
    android:id="@+id/button"
    app:layout_constraintTop_toBottomOf="@+id/id0"
    android:layout_marginTop="20dp"
    app:layout_constraintLeft_toRightOf="@+id/LogIn"
    app:layout_constraintRight_toRightOf="parent"/>
```

形成链条

实现两个按钮整体居中

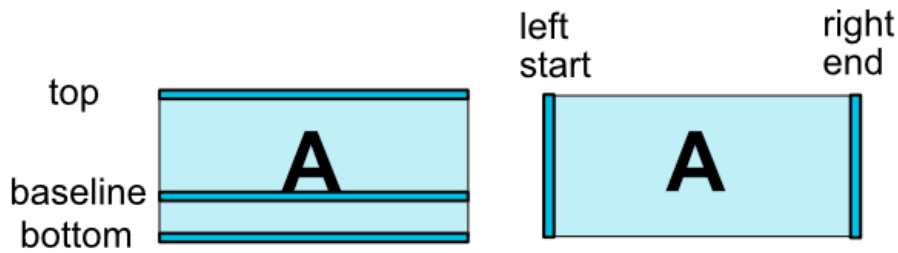
（3） 实验遇到困难以及解决思路

① **问题：**在通过拖拽把所有控件都安排好后，在虚拟机上运行时，几乎所有控件的位置都乱了。

解决思路：我觉得我可能对于 ConstraintLayout 不够了解，所以我阅读了一篇博客 http://blog.csdn.net/zxt0601/article/details/72683379。阅读之后，我知道了 ConstraintLayout 是一个为了解决布局嵌套和模仿前端 flexible 布局的一个新布局。ConstraintLayout 即为约束布局，通过控件之间以及控件和 Guideline 之间的相互约束来调整控件在整个布局中的位置，从而实现了减少多层嵌套布局的目的。因此，在设置控件时，我们要理清控件之间的约束关系。合理运用 ConstraitLayout 的属性来约束控件。

例如：相对属性：

```
* layout_constraintLeft_toLeftOf
* layout_constraintLeft_toRightOf
* layout_constraintRight_toLeftOf
* layout_constraintRight_toRightOf
* layout_constraintTop_toTopOf
* layout_constraintTop_toBottomOf
* layout_constraintBottom_toTopOf
* layout_constraintBottom_toBottomOf
* layout_constraintBaseline_toBaselineOf
* layout_constraintStart_toEndOf
* layout_constraintStart_toStartOf
* layout_constraintEnd_toStartOf
* layout_constraintEnd_toEndOf
```



这里的属性都形如：layout_constraintX_toYof=“id”，X 代表这个子控件自身的哪条边，Y 代表约束控件的哪条边。例如：layout_constraintLeft_toLeftof=“parent”代表这个 TextView 控件的左边和父控件的左边对齐。

② **问题：**在我明白要通过控件相互约束来实现布局后，还是实现不了想要的效果。

解决思路：经过多次实践和反思，我认为在实现控件之间的相互约束时，我们应该选定一个思考方向，例如在设置控件时，我们只要明确它被哪些控件约束着就好

了，不要再去思考它要约束哪些控件，这样才不会混乱。而且约束控件时不一定 4 个方向都要约束，视情况而定。

③ **问题：**通过 android:gravity 和 android:layout_gravity 都没办法让标题居中。

解决思路：先了解了一下 gravity 和 layout_gravity 的区别。gravity 用于指定设置该属性的组件下的子组件的位置，即设置 view 里面的内容相对于 view 的位置，而 layout_gravity 用于指定该属性的组件相对于父组件的位置。通俗的来说，就是 android:gravity 只对该组件内的东西有效，android:layout_gravity 只对组件自身有效。了解了两者的区别后，我觉得应该使用 android:layout_gravity，但是设置了 android:layout_gravity=“center”之后，标题的位置没有变化，后来百度发现 android:layout_gravity 只在 LinearLayout 和 FrameLayout 中有效，于是只能另找办法。在百度 ConstraintLayout 时发现可以通过将该控件的左、右两边分别和父控件（即 ConstraintLayout）的左右两边对齐来实现控件的居中。

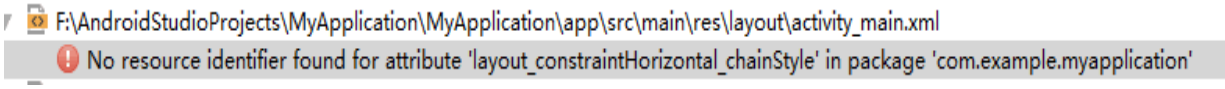
```
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```

④ **问题：** 不会实现两个按钮 button 的整体居中。

解决思路：刚开始是通过不断地调整两个 button 与屏幕左右边框的距离来把两个 button 居中，但是总是没有办法得到很好的效果。在不断的尝试中，偶然间将两个 button 形成了 Chain，在可视化界面中看到 chain 的图标后就去百度了 Chain。Chain 链是一种特殊的约束让多个 chain 链连接的 Views 能够平分剩余空间位置。通过 `app:layout_constraintRight_toLeftOf="@+id/button"` `app:layout_constraintLeft_toRightOf="@+id/LogIn"` 将两个 button 形成了一条 chain 链。但是形成 chain 链之后两个 button 之间的间隔很大，没办法设置两个 button 之间的间隔为 10dp。百度发现，chain 链有三种不同的模式：Spread Chain 链模式、Spread Inside Chain 链模式、Packed Chain 链模式。Chain 链的默认模式就是 spread 模式，它将平分间隙让多个 Views 布局到剩余空间。 spread inside 模式，它将会把两边最边缘的两个 View 到外向父组件边缘的距离去除，然后让剩余的 Views 在剩余的空间内平分间隙布局。packed ，它将所有 Views 打包到一起不分配多余的间隙（当然不包括通过 margin 设置多个 Views 之间的间隙），然后将整个组件组在可用的剩余位置居中。所以在没有设置 Chain 链的模式时，它默认为 spread 模式，导致两个 button 平分空间，没办法调两个 button 之间的间隔。于是我们设置 chain 链的模式 `app:layout_constraintHorizontal_chainStyle="packed"` ，然后再通过 `android:layout_marginRight="10dp"` 来设置两个 button 之间的间隔。设置好间隔后，

两个 button 还是没有居中，于是我们将 chain 链看成一个整体，在左边的 button 中设置 `app:layout_constraintLeft_toLeftOf="parent"/>` 在右边的 button 中设置 `app:layout_constraintRight_toRightOf="parent"/>` 来实现两个 button 的整体居中。

⑤ **问题：**



解决思路：

将 build.gradle 中的 `compile 'com.android.support.constraint:constraint-layout:1.0.0-alpha7'`

改成 `compile 'com.android.support.constraint:constraint-layout:1.0.2'`

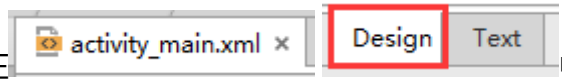
四、 课后实验结果

在课堂上是通过直接编写 XML 代码来实现的，课后尝试了一下使用可视化的方式来编写界面。

（1）实验步骤：

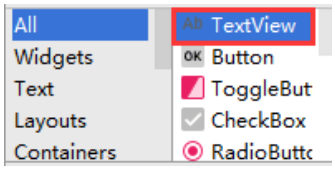
① 在 app\build.gradle 修改版本号：

```
compile 'com.android.support.constraint:constraint-layout:1.0.2'
```

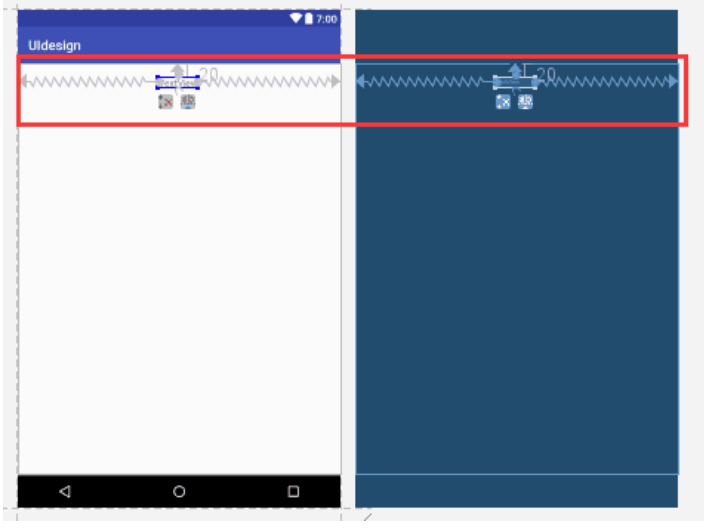
② 在 

中进行设计页面布局。

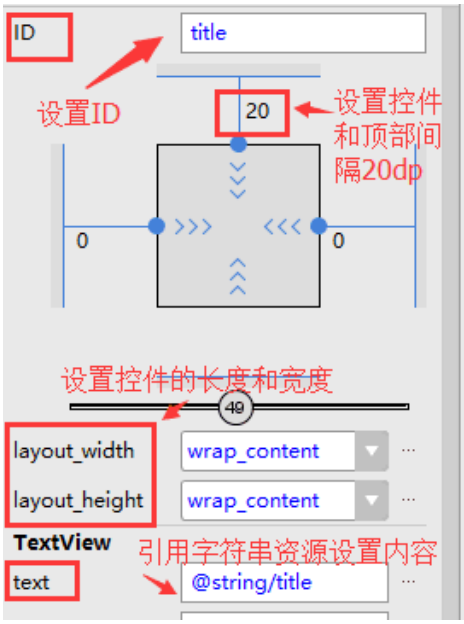
③ 将 TextView 控件直接拖到 layout 中，然后创建约束。

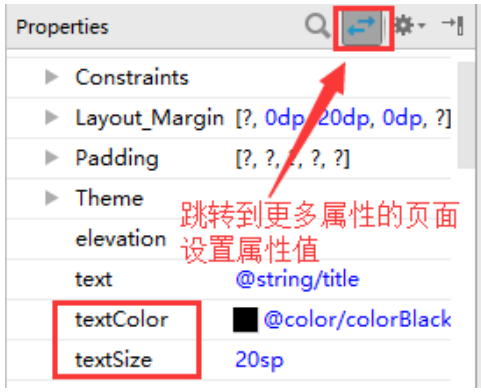


在 TextView 的顶部控键和父控件（ConstraintLayout）的顶部控键创建约束，在 TextView 的左右控键分别和父控件左右控键创建约束（从而实现居中）。

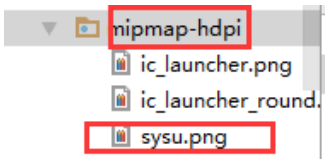


然后在 UI 编辑器右侧的 Inspector 中设置控件的相关属性。

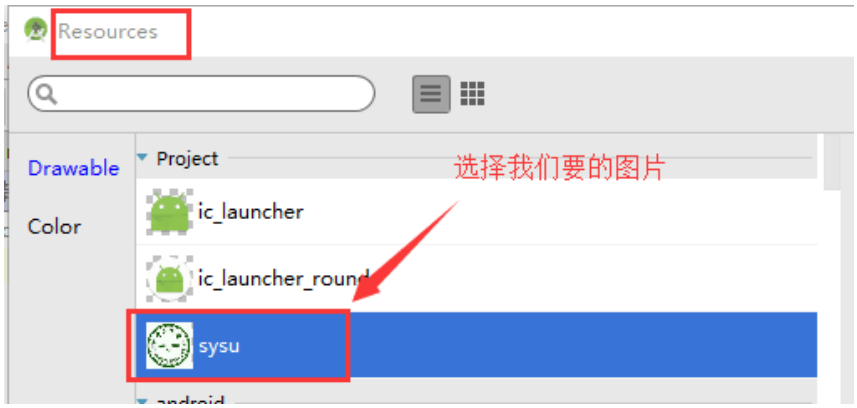




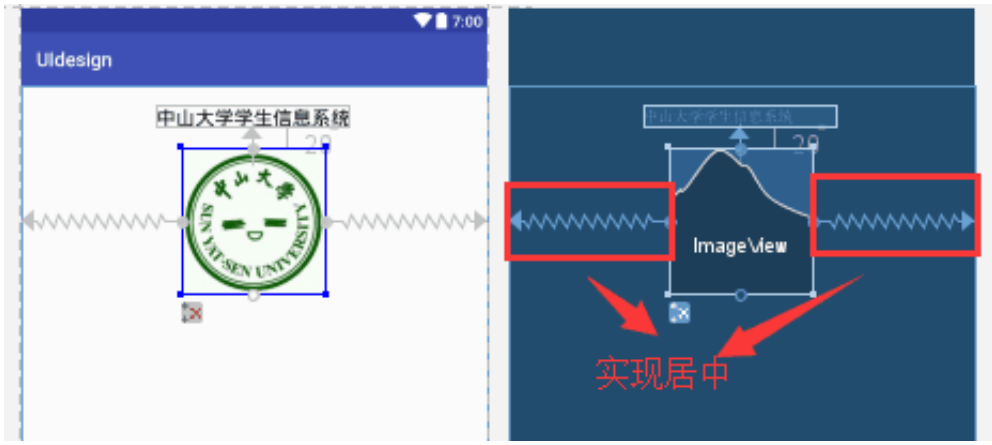
④ 将图片 sysu.png 复制到 app\src\main\res\mipmap-hdpi 中。



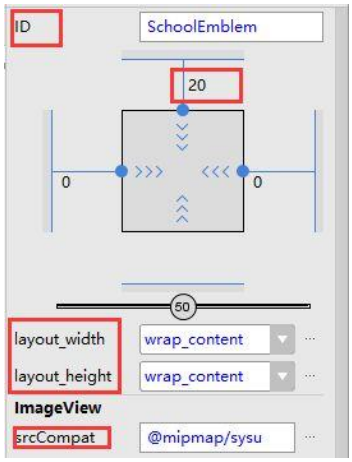
将 ImageView 控件拖进 layout 中，出现 resource，选择我们需要的图片。



创建约束：



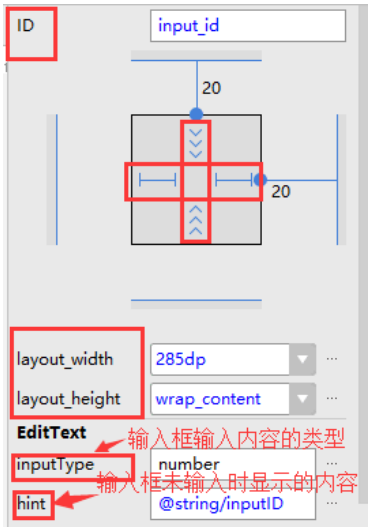
在 Inspector 设置相关属性：



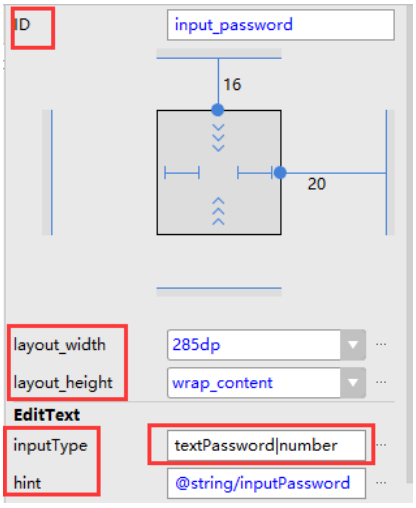
⑤ 因为我在 UI 编辑界面中没有找到 EditText 这个控件，所以通过在 layout\activity_main.xml 中编写代码让这个控件出现在 layout 里面。

学号输入框：创建约束：

设置相关属性：



密码输入框：设置相关属性：

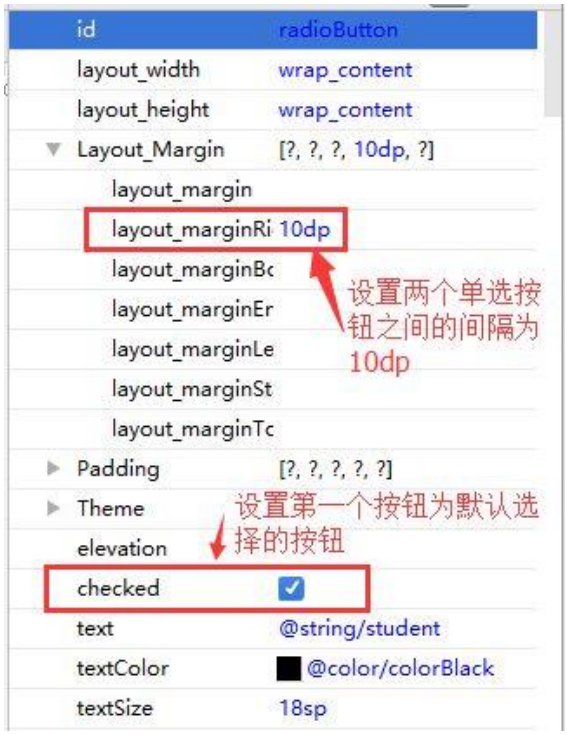


⑥ 将 RadioGroup 控件和 RadioButton 控件拖进 layout 中。

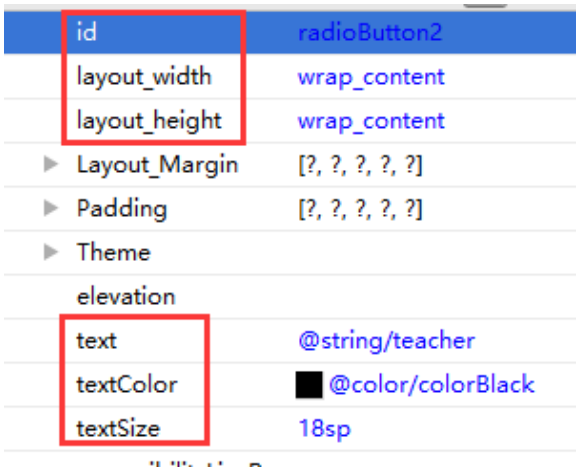
创建 RadioGroup 约束：

设置 RadioGroup 相关属性：

设置学生按钮的相关属性：



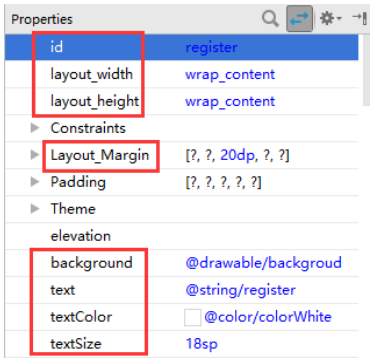
教职工按钮属性设置：



⑦ 将 Button 控件拖到 layout 中。在两个按钮之间形成链条 Chain。

设置登录按钮的属性：

设置注册按钮的属性：



（2）实验结果：

（3）相关内容：

① 控件类型：

调整尺寸控键 ：该控键允许你调整 widget 尺寸。 

侧约束控键 ：该控键让你指定 widget 的位置。 

基线约束控键 ： 帮助你对齐任意两个 widget 的文字部分。 

② 创建约束：

在 widget 的某个控键上点击并按住，然后拖到另一个 widget 的约束控键内。一旦显示绿色就松手，约束就会被创建。

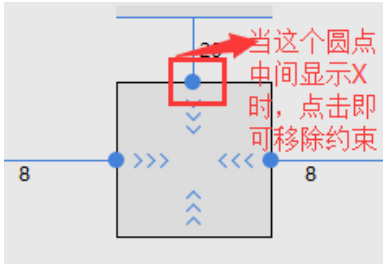
移除约束：

移除某个约束只需点击指定约束的控键，当它显示为红色时点击即可。

移除全部约束需要点击如右按钮： 

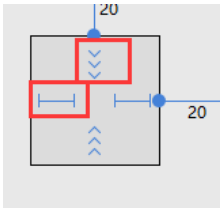
③ Inspector：

移除 constraint ：



相对于约束来放置 widget ：当在一个 widget 有至少两个相对的连接，比如说顶部和底部，或者左侧和右侧，然后就可以使用滑动条来调节 widget 在链接中的位置。你还可以改变屏幕方向来进一步调整方位。

控制 widget 内部尺寸 ： Inspector 内部的线让你可以控制 widget 内部尺寸。



Fixed ：可以调整 widget 的宽度和高度。 

AnySize ： 使得 widget 占据所有可用的控键来满足约束。 

五、 实验思考及感想

经过这次实验，我对于 Androidstudio 的 ConstaintLayout 布局和一些控件例如 button、TextView 等有了更加深入的理解，能够灵活地运用它们来进行基本的 UI 界面设计。通过这次实验，我知道了 ConstraintLayout 是一个为了解决布局嵌套和模仿前端 flexible 布局的一个新布局。ConstraintLayout 即为约束布局，通过控件之间以及控件和 Guideline 之间的相互约束来调整控件在整个布局中的位置，从而实现减少多层嵌套布局的目的。因此当我们在使用 ConstraintLayout 布局来进行 UI 界面设计时，我们要理清楚每个控件之间的约束关系。而且当我们在约束控件时，我们要选定一个思考方向，例如对于每个控件，我们都只考虑它被什么控件约束，不要考虑它要约束什么控件。当我们在学习新知识时，我们要多尝试。在我们感觉某个知识点难以理解时，我们不妨先动手尝试一下，借助实践来帮助我们理解知识。当我们遇到问题时，我们要善于自己寻找答案。