移动互联网应用开发课程设计个人报告

2020-2021 学年 第二学期，2019 级本科，计算机科学与技术

项目名称：MOTA 学习魔塔
姓名：陈语奇
学号：1906010232
承担任务：UI 制作
评分：

1.任务概述

本人主要负责这个 APP 的 UI 实现，实现整个 APP 的 UI 界面以及部分页面跳转的效果，根据 UI 界面设计，优化并完成所给设计的所有 UI 组件及界面。UI 的实现为后端的实现提供了可视化的界面，为后端提供了方便，同时也是整个 APP 的门面。

2.设计思路

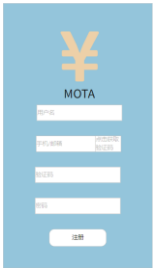
初期界面设计由其他组员提供，根据其他组员提供的设计，实现并完善 UI 界面。

2.1 标题界面、登陆界面、注册界面

这三个界面的实现较为简单，如下图 2-1，2-2，2-3 所示，考虑采用 LinearLayout 嵌套布局实现。

2-1 标题界面

2-2 登陆界面

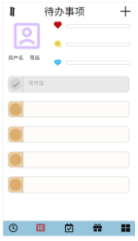


2-3 注册界面

2.2 APP 主界面

主界面一共有五个，如下图 2-4 至 2-8 所示，由于界面的上端标题栏、用户信息以及下方底部的按钮栏目变动较小，考虑使用一个 Activity，采取在当前 Activity 上才去切换 Fragment 的方式来实现页面切换的功能。顶部标题工具栏可以使用自定义的 toolbar 实现。每个页面的条目显示由于会有很多条，需要通过滑动显示全部，我考虑使用 ListView 控件来实现。

2-4 习惯页面



2-5 待办事项页面

2-6 每日任务页面

2-7 奖励页面

2-8 菜单页面

2.3 成就页面、账单页面

成就和账单页面较为类似，如下图 2-9 和 2-10 所示考虑使用 LinearLayout，滑动条目仍然使用 ListView 控件实现。

2-9 成就页面

2-10 账单页面

2.4 新建条目页面

四个新建条目界面都比较类似，由于是在 Activity 上方显示，所以使用 Dialog 类来实现显示操作，布局仍然使用 LinearLayout 嵌套布局。

2-11 新建习惯

2-12 新建待办事项

2-13 新建每日任务

2-14 新建奖励

3.技术路线与方法

由于本人主要负责 UI 实现，所以下面介绍每个页面具体的布局设计。其中的图片文件由阿里巴巴矢量图标库以及好友友情提供。

3.1 标题界面、登陆界面、注册界面

3.1.1 标题界面

标题界面主要是由一个图标，APP 的标题，以及登录和注册按钮构成，如下左图 3-1 所示。LinearLayout 主要布局如下面右侧代码所示。每个控件的 layout_gravity 值设置的均为 center 使得控件在整个布局居中显示。

3-1 标题界面

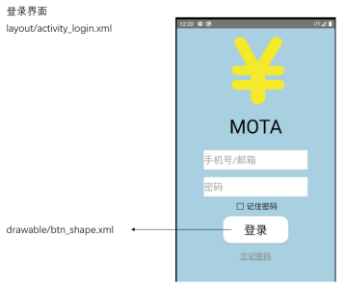
```
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/main_background"
    android:orientation="vertical"
    tools:context=".Login">
    <ImageView ... >
    <TextView ... >
    <TextView ... >
    <Button ... >
```

```
    <Button ... >
</LinearLayout>
```

3.1.2 登陆界面

登陆界面和标题界面布局类似，如下左图 3-2 所示，同样是使用 LinearLayout，登录按钮使用了圆角，实际上是自己重新写了一个 btn_shape.xml 的样式，如下右侧代码所示，设置了背景色，圆角以及文字与按钮四周的间隙。后面的注册界面也同样运用了这个样式。

记住密码采用的是 CheckBox 控件，下面的忘记密码也是一个按钮，不过我将背景更换为了透明色（android:background="@android:color/transparent"），并且在 string.xml 中增加了下划线（<string name="forget_psd"><u>忘记密码</u></string>）呈现出这种效果（遗憾的是这个忘记密码暂未实现其功能）。

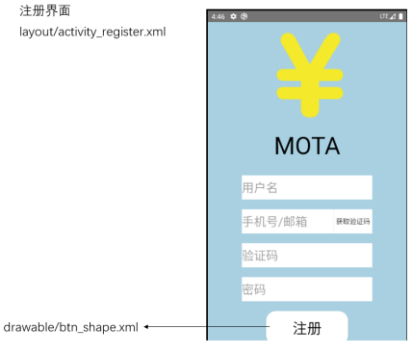


3-2 登陆界面

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/andro
id" android:shape="rectangle">
    <!-- 设置背景颜色 -->
    <solid android:color="@color/white" />
    <!--  设置圆角  -->
    <corners android:radius="20dp"/>
    <!-- 设置文字与按钮的四周间隙 -->
    <padding
        android:left="10dp"
        android:right="10dp"
        android:top="10dp"
        android:bottom="10dp" />
</shape>
```

3.1.3 注册界面

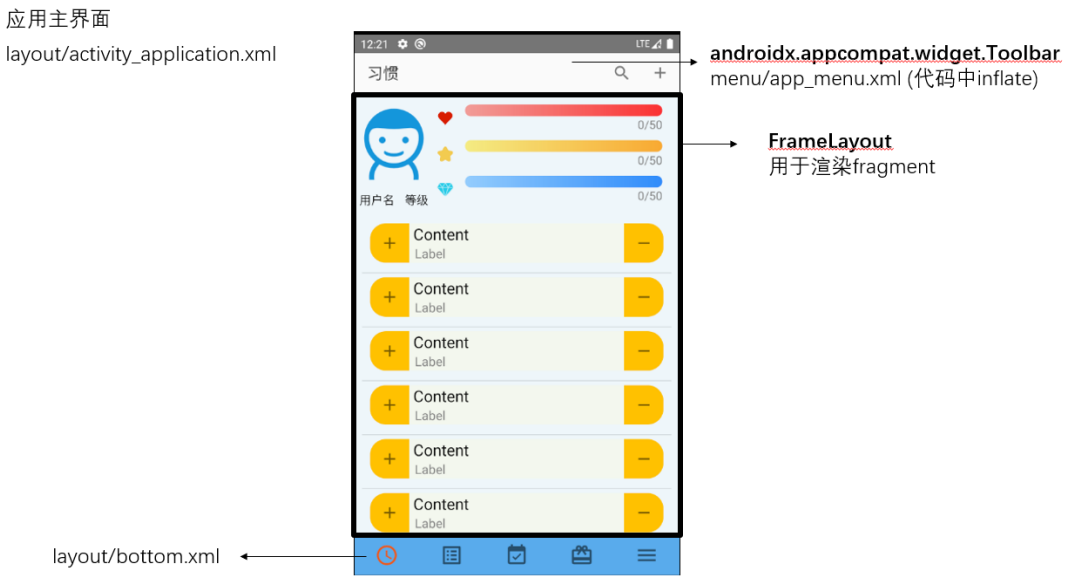
同样的，与前两个界面类似，仍然使用的是 LinearLayout 布局，如下图 3-3 所示，不同的是在手机号/邮箱那一处使用了布局的嵌套，使得 Button 与 EditText 组成一行。布局嵌套是之后经常要使用到的一种方法，后面不再过多描述。



3-3 注册界面

3.2APP 主界面

主界面只由一个 Activity 构成基本结构，如下图和代码所示。



3-4 主界面结构

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/app_toolbar"
        android:layout_width="match_parent"
        android:layout_height="50dp">
    </androidx.appcompat.widget.Toolbar>
    <FrameLayout
        android:id="@+id/frame_content"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/app_background"
        android:layout_below="@+id/app_toolbar"
        android:layout_above="@+id/app_bottom">
    </FrameLayout>
    <include
        layout="@layout/bottom"
        android:id="@+id/app_bottom"
        android:layout_height="50dp"
        android:layout_width="match_parent"
        android:layout_alignParentBottom="true">
    </include>
</RelativeLayout>
```

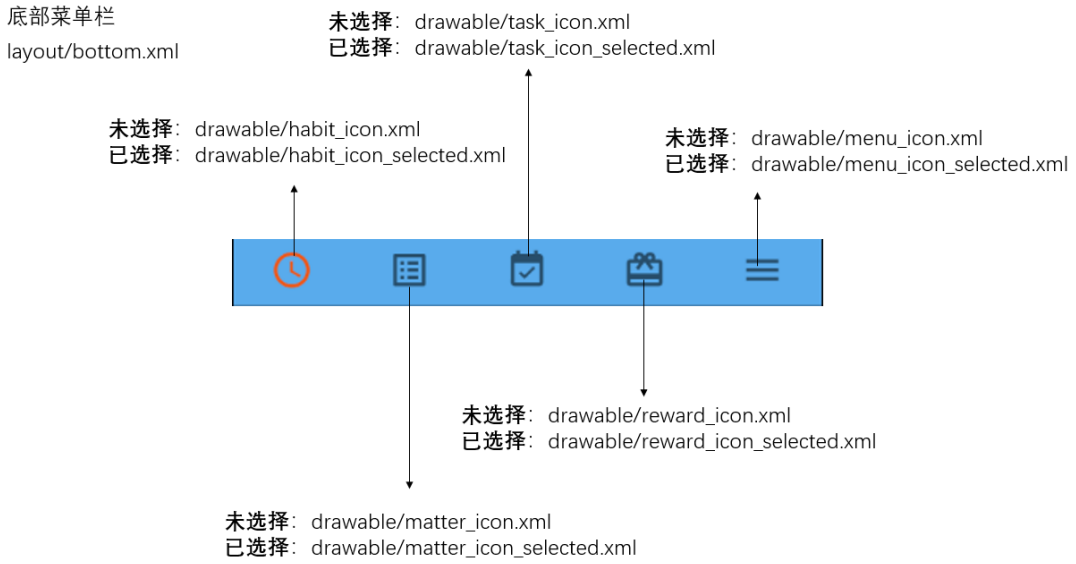
在主界面 Activity 中，使用了相对布局，即 RelativeLayout。最上方用了 Toolbar 控件，用来自定义

顶部工具栏。最下方用了自己写的 bottom.xml 布局文件，再使用 include 导入。由于之前使用 LinearLayout 布局时，在输入文字的时候会导致底部菜单栏被键盘挤压到上方，所以采用了相对布局，使得底部菜单栏能够保持在最下方。中间部分嵌套了一个 FrameLayout，以便于后期在五个界面使用 Fragment 的来回切换，只需要在这个给定的 FrameLayout 上进行渲染就可以，十分方便且快捷。

3.2.1 主界面的自定义布局

1. 底部菜单栏

底部菜单栏保存在 bottom.xml 文件中，具体结构如下图 3-5 所示。

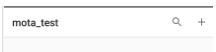


3-5 底部菜单栏结构

该底部菜单栏实现较为简单，由五个 ImageView 控件水平线性布局排列而成。由于之后要设置监听所以必不可少的代码是要对这五个 ImageView 设置可获取焦点（android:focusable="true"），否则无法设置监听。

2. 顶部菜单栏

顶部菜单栏定义在 menu 文件下 app_menu.xml 当中，实现的效果如下左图 3-6 所示，代码如下右所示



3-6 顶部菜单栏

```xml
<?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/action_search"
        android:icon="@drawable/search_icon"
        app:showAsAction="always"
app:actionViewClass="androidx.appcompat.widget.SearchView"
        android:title="Search" />
    <item
        android:id="@+id/menu_add"
        android:icon="@drawable/add_icon"
        android:title="Add"
        app:showAsAction="always" />
</menu>
```

两个 item 分别为查找按钮及增加按钮，其中 showAsAction=”always”属性，表示一直显示。运用自定义的 menu 可以在后端代码中使用自带的类更加方便的调用，使得 UI 更加美观。

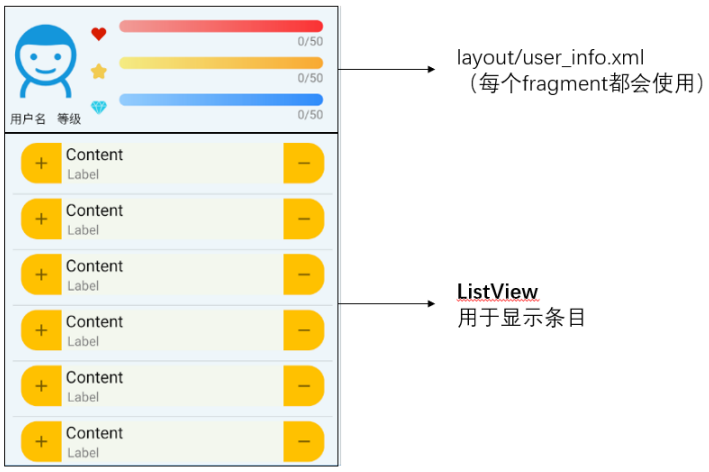
下面是后端中对自定义 menu 的初始化以及按钮点击功能的实现，使用 setSupportActionBar()方法实现对 toolbar 的初始化。然后重写 onCreateOptionsMenu(Menu menu)方法实现渲染 toolbar 样式以及对 toolbar 按钮上的监听。后端的成员只需要再重写监听方法就可以实现效果了。

```
//toolbar 初始化
toolbar = (Toolbar) findViewById(R.id.app_toolbar);
setSupportActionBar(toolbar);
getSupportActionBar().setHomeButtonEnabled(false);
getSupportActionBar().setDisplayHomeAsUpEnabled(false);

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    //toolbarOptionmenu 监听 inflate
    getMenuInflater().inflate(R.menu.app_menu,menu);
    mySearch = menu.findItem(R.id.action_search);
    add_btn = menu.findItem(R.id.menu_add);
    SearchView mySearchView = (SearchView) mySearch.getActionView();
    mySearchView.setQueryHint("Search");
    // toolbar 查找
    mySearchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) { return false; }
        @Override
        public boolean onQueryTextChange(String newText) {return false; }
    });
    //toobar 增加
    add_btn.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem item) {return true; }
    });
    return super.onCreateOptionsMenu(menu);
}
```

3.2.2fragment 布局定义

本 APP 一共有五个需要切换的页面，分别为习惯 fragment、待办事项 fragment、每日任务 fragment、奖励 fragment 和菜单 fragment，每个 fragment 基本组成相差无几，以习惯 fragment 为例，如下图 3-7 所示。

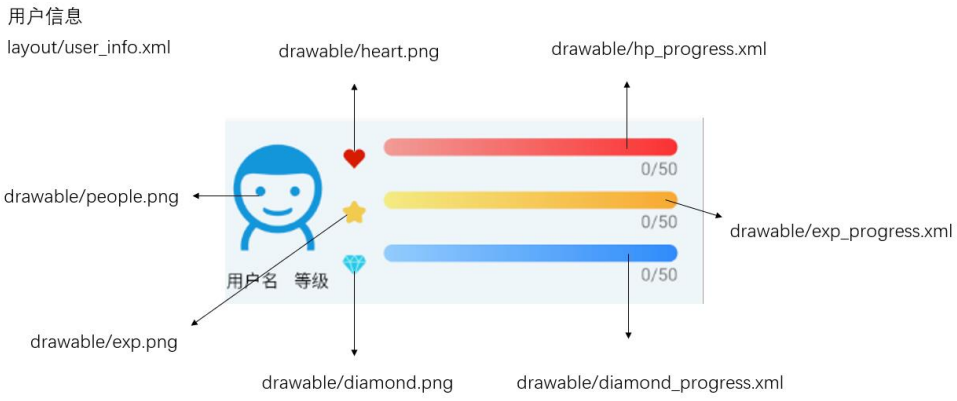


3-7 习惯 fragment 结构

每个 fragment 都是用帧布局嵌套线性布局，即外部是 FrameLayout 嵌套内部的 LinearLayout，其中上面是一个自定义的布局文件 user_info.xml，用来显示用户的个人信息下面是一个 ListView 控件，用来显示各种各样的条目（菜单 fragment 除外）。代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".habit">
    <!-- TODO: Update blank fragment layout -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:paddingTop="5dp">
        <include layout="@layout/user_info"></include>
        <ListView
            android:id="@+id/habit_lv"
            android:layout_marginTop="10dp"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp">
        </ListView>
    </LinearLayout>
</FrameLayout>
```

user_info.xml 布局文件用于显示用户信息，我们需要显示用户的头像、用户名、等级、HP（血量）、EXP（经验值）以及所拥有的钻石数量。如下图 3-8 所示。



3-8 用户信息（user_info.xml）

再这个布局文件中由于需要显示 HP、EXP 等的占比值，所以使用了进度条 ProgressBar 控件，下面具体介绍进度条这一控件，以 HP 值为例，布局文件如下所示。

```
<ProgressBar
    android:id="@+id/hp_progressBar"
    android:layout_width="210dp"
    android:layout_height="15dp"
    android:layout_marginLeft="10dp"
    android:max="100"
    android:progress="100"
    android:progressDrawable="@drawable/hp_progress"
    style="@android:style/Widget.ProgressBar.Horizontal"></ProgressBar>
```

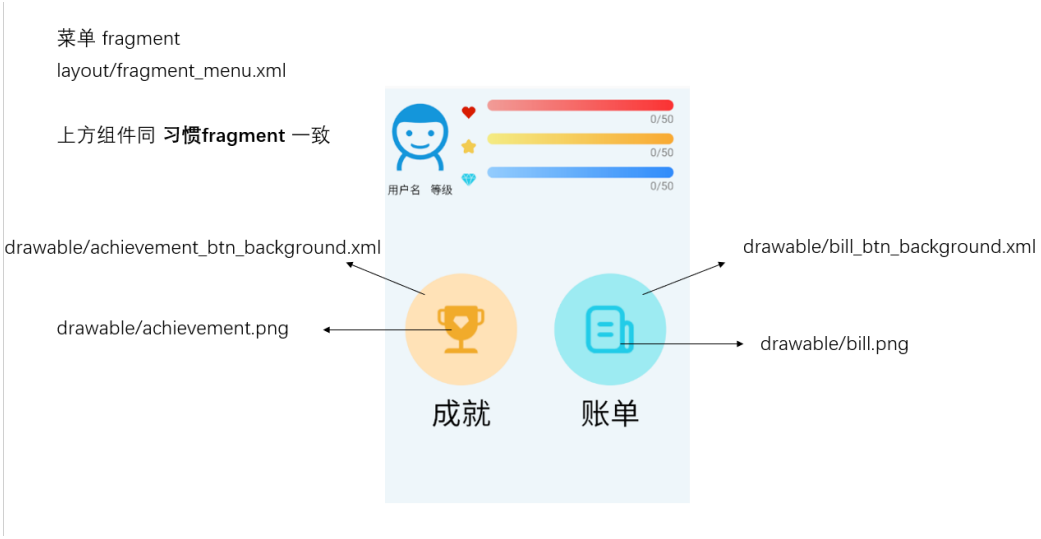
这是进度条的基本属性展示，其中 android:max 属性指的是进度条的最大值，android:progress 属性指的是当前进度条的值，然后 style 属性这里我使用的是横向的进度条所以是 android:style/Widget.ProgressBar.Horizontal，android:progressDrawable 属性则是我自定义了一个 xml 文件用于渲染进度条的渐变色，使得整体 UI 更加的美观，下面具体介绍自定义的渲染 xml 文件。代码如下所示。

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@android:id/background">
        <!--形状-->
        <shape>
            <corners android:radius="20dp" />
            <solid android:color="@color/progress_bar_background"></solid>
        </shape>
    </item>
    <item android:id="@android:id/progress">
        <clip>
            <shape>
                <corners android:radius="20dp" />
                <gradient
                    android:angle="0"
                    android:startColor="#80F44336"
                    android:endColor="#CBFF0000"/>
            </shape>
```

```
        </clip>
    </item>
</layer-list>
```

首先是最外层的 layer-list，内部有两个 item 一个定义形状，我设置了进度条的圆角为 20dp，另一个定义渐变色。<gradiet>下的属性 startColor 为起始的颜色 endColor 属性为终止的颜色。

最后单独介绍菜单 fragment，菜单 fragment 的下方控件与其他的不同，仅仅由两个 ImageButton 和 TextView 组成，如下图所示 3-9 所示。



3-9 菜单 fragment

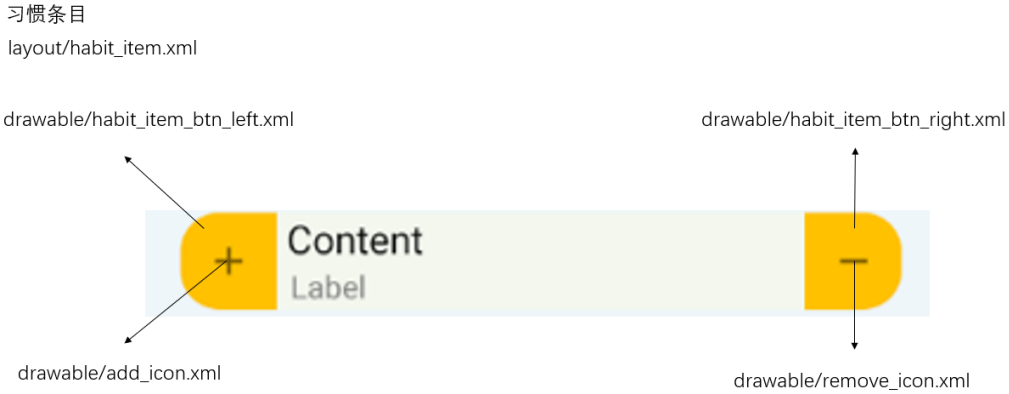
如何将按钮显示为原型只需要在其按钮背景布局中将其圆角设置为原先按钮长度的一半，就可以达到圆形得到效果。

3.2.3ListView 中的 item 设计

由于前四个 fragment 的结构类似，这里不做过多的赘述。这部分将着重介绍对于每个 ListView 中显示的条目即 item 样式。

1.习惯 item

习惯 item 由三个部分组成左右两边是两个 ImageButton,用于点击记录完成次数。中间是一个 TextView，用于显示习惯的内容和标签。如下图 3-10 所示。



3-10 习惯 item

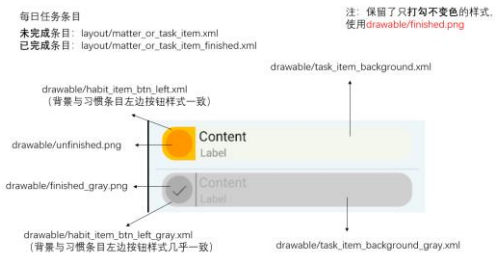
习惯 item 布局文件依旧采取了最简单的水平线性布局，并且额外设置了分割线属性（android:outlineAmbientShadowColor="@color/black"）使得界面更加美观，中间的 Content 和 Label 则是又嵌套了纵向线性布局，并对其颜色大小进行了适当的修改，需要注意的是，由于 ImageButton 要显示在两侧，所以外部的线性布局的宽度必须要设置为 match_parent 。

（android:layout_width="match_parent"）。再对每个控件设置 layout_weight 达到这样的效果。部分代码如下所示。

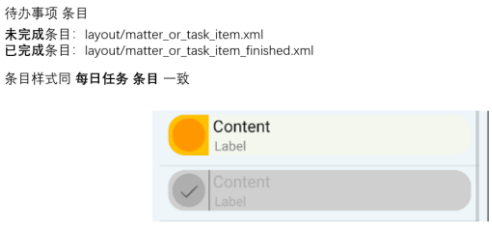
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="5dp"
    android:paddingBottom="12dp"
    android:outlineAmbientShadowColor="@color/black"
    android:gravity="center_vertical">
    <ImageButton ...>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:orientation="vertical"
        android:layout_weight="1"
        android:background="@color/rice_white">
        <TextView ...>
        <TextView ...>
    </LinearLayout>
    <ImageButton ...>
</LinearLayout>
```

2.待办事项、每日任务 item

这两个 item 由于样式相同，所以只写了一个 xml 文件。由于存在完成任务会变灰的需求，所以写了一个未完成的样式以及一个已完成的样式，由于版本更新的缘故，仍然保留了只打勾不变色的版本，可以随时更换。如下图 3-11、3-12 所示。



3-11 每日任务 item



3-12 待办事项 item

这个 item 的布局文件与上面所写的类似，只是缺少了右侧的 ImageButton，所以左侧的背景仍然沿用选择了 habit_item_btn_left.xml 这一布局文件（android:background="@drawable/habit_item_btn_left"），只对其中的图片文件进行了修改（android:src="@drawable/unfinished"），由于找到的图片会过大或过小，我在对其属性设置了适应大小（android:scaleType="fitCenter"）。

在灰色 item 中你可以看到，左侧的 Button 与右侧的 TextView 中存在一条竖线，如果仅仅修背景颜色，即 background 属性的值，是没有办法做到的，所以我对其背景又重写了一个 xml 文件并调用。代码如

下。

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item><shape >
            <solid android:color="@color/light_gray" />
            <corners
                android:topLeftRadius="0dp"
                android:topRightRadius="20dp"
                android:bottomRightRadius="20dp"
                android:bottomLeftRadius="0dp"/>
        </shape></item>
    <item android:bottom="-4dp" android:right="-4dp" android:top="-4dp">
        <shape><stroke android:width="1dp" android:color="@color/dark_gray"/></shape>
    </item>
</layer-list>
```

仍然是外部一个 layer-list，第一个 item 设置的圆角属性（<coners>标签），我只需要右侧的两个为圆角，所以 android:topRightRadius，android:bottomRightRadius 设置为 20dp。下面一个 item 我需要只显示左边框，我就必须将上下右边先设置为-4dp（item android:bottom="-4dp" android:right="-4dp" android:top="-4dp"），然后对左边设置边框线条（<stroke>标签）。

3.奖励 item

这个 item 与上面的也比较类似，原先打算沿用左侧一个 TextView，右侧一个 ImageButton 的格式，但是后来发现，右侧还需要显示文字，于是放弃使用 ImageButton，考虑使用最基础的 Button 控件。效果图如下图 3-13 所示。

3-13 奖励 item

右侧的按钮背景由于与习惯 item 右侧按钮背景一致，仍然沿用了其 xml 文件。左侧的显示界面背景仍然是自定的一个 xml 文件，由于这次是左侧需要圆角，所以只需稍微修改一些属性的值便可以实现效果。

关于右侧如何既显示图片又显示文字，代码如下所示。

```
<Button
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:background="@drawable/habit_item_btn_right"
    android:paddingLeft="5dp"
    android:drawableLeft="@drawable/diamond_small"
```

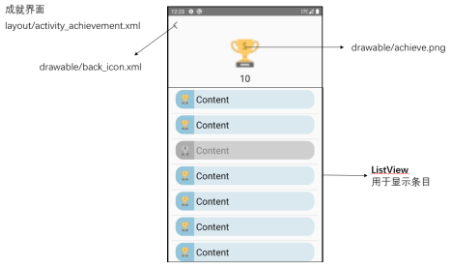
```
    android:text="10"
    android:layout_gravity="right">
</Button>
```

从中我们可以看出，只需要增加 android:drawableLeft 这一属性，便可以使图片添加在文字的左侧。

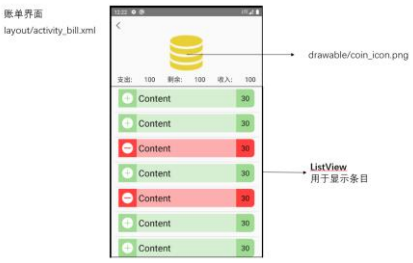
在 user_info.xml 中的钻石图标与这个相同，我没有使用 diamond.png 而是又重新下载了一个更小的 png 图片，是由于过大的 png 图片文件显示在按钮中会显示的很大，由于本人能力原因没有办法像 ImageView 中调节其 scale 属性。所以我选择了另一个方法找到一个更小的图片。

3.3 成就界面、账单界面

成就界面和账单界面布局类似，也是由上下两个部分构成，最上方是一个 ImageButton 构成的返回按钮，中上方显示图片，数据等，下方是 ListView 显示条目。如下图 3-14，3-15 所示。



3-14 成就界面

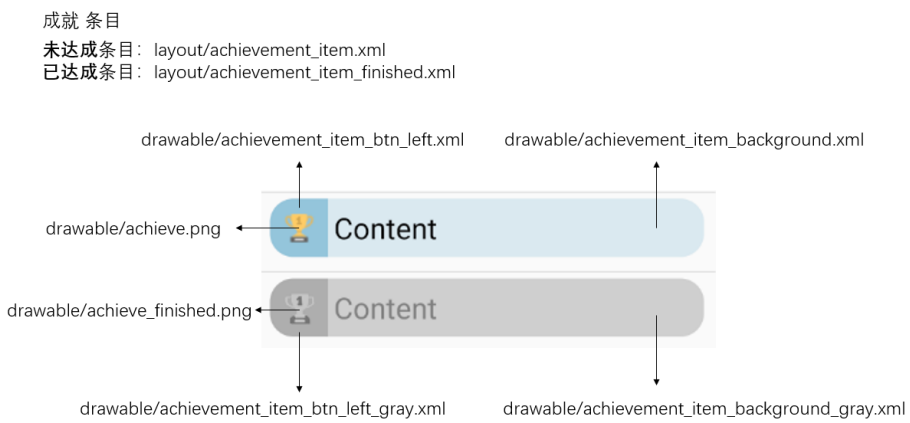


3-15 账单界面

由于实现起来较为容易，且与之前布局类似，这里不再过多赘述。主要将介绍两个部分的 item 样式。

3.3.1 成就 item 样式

与之前的待办事项 item 类似，成就 item 也拥有原色与灰色两种形式。如下图 3-16 所示。



3-16 成就 item

成就 item 的结构与之前的待办事项 item 结构极其类似，只需要更换背景颜色，图片文件即可。

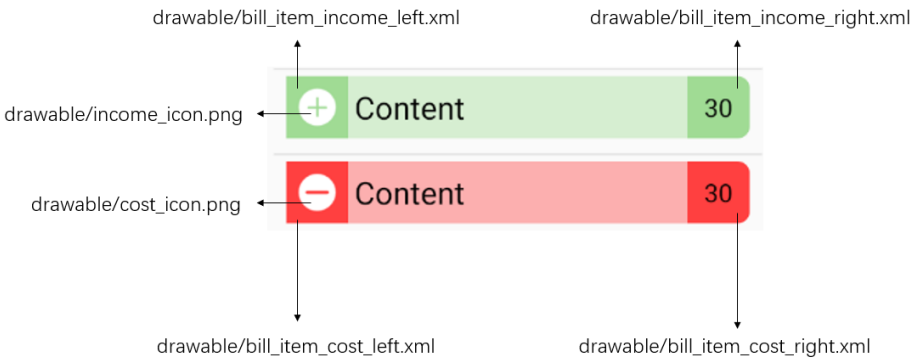
3.3.2 账单 item

账单 item 与习惯 item 相类似，左侧是 ImageButton，右侧更换为了 TextView。由于存在收入和支出，需要写两个不同的 xml 文件。如下图 3-17 所示。

账单 条目

收入条目：layout/bill_item_earn.xml

支出条目：layout/bill_item_cost.xml



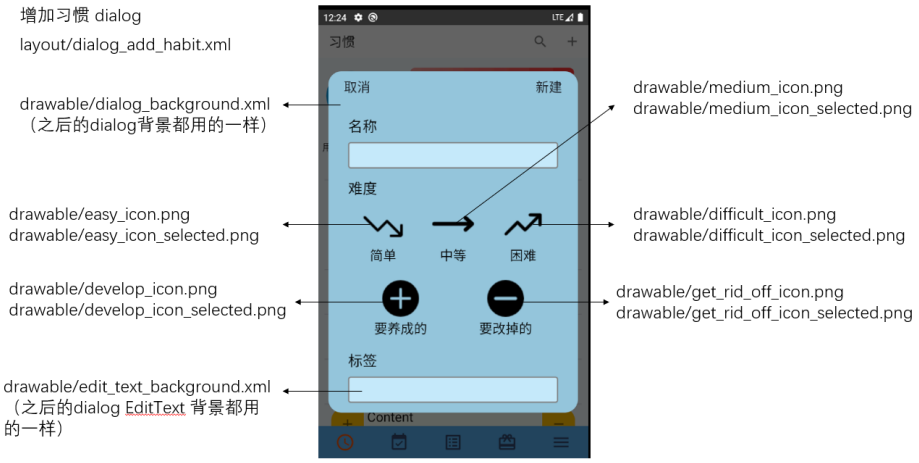
3-17 账单 item

对于背景，都是自定义的 xml 文件，在 xml 文件中直接修改，方便且快捷。前面已经写过很多类似的，到这里已经是轻车熟路。

3.4 用来增加条目的 dialog

在习惯、每日任务、待办事项、奖励中都需要增加新的条目（点击右上角的加号），屏幕上会弹出一个对话框用来填写增加的信息。对于每一个增加的界面，都需要一个 xml 来自定义 dialog 样式。

增加的页面基本都比较相似，这里以增加 dialog 为例具体说明，效果如下图 3-18 所示。



3-18 增加习惯 dialog

整体上仍然是才去垂直的线性布局，这个页面对布局的嵌套要求较高，考虑到美观性与和谐性的统一，需要对每个控件的位置进行排版，其中 padding、margin、layout_weight 等属性需要熟练使用。难度不大，但是需要细心和耐心。

剩余的 dialog 页面如下图 3-19、3-20、3-21 所示，其本质与上面所示没有差别。结构也都基本类似，在这里不再做过多赘述。

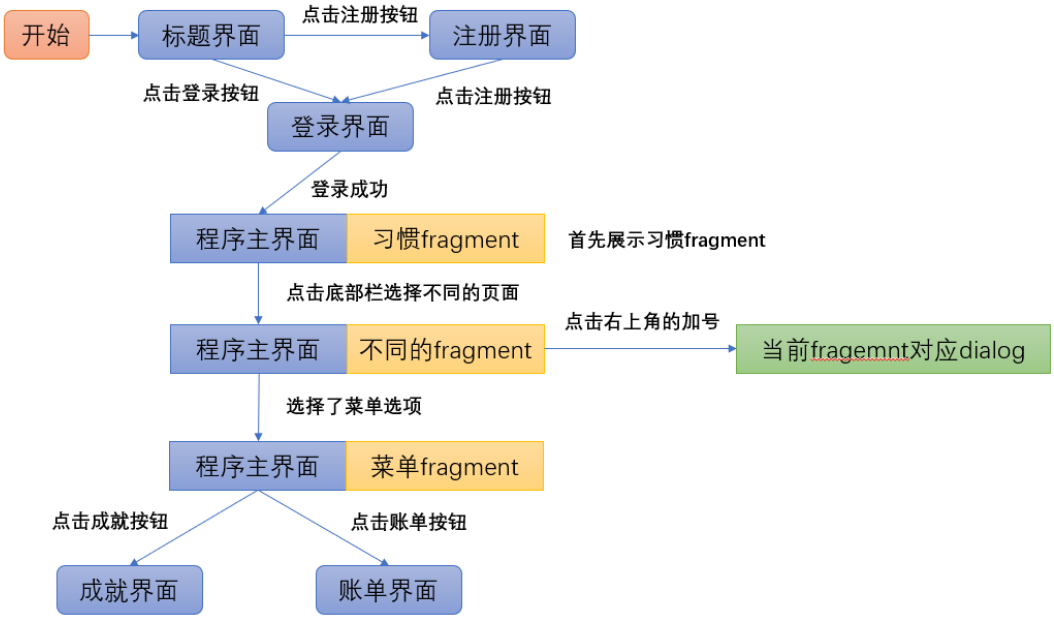
3-19 增加每日任务 dialog

3-20 增加待办事项 dialog

3-21 增加奖励 dialog

与其他的不同的是在增加待办事项 dialog 中最下方的选择实时间应当是点击之后出现日期选择器然后用户选择日期后显示在 EditText 中，这一点将会在后文说道。

3.5 页面之间的转换



3-22 页面之间的转换

页面间的转换如上图所示，目前采取 Intent 执行页面之间的相互转化，后期可以使用 putExtra()方法增加页面之间传递的参数。

4.实现效果及测试用例

在虚拟机上运行时 UI 界面较为完好，在手机上运行时，会出现尺寸不匹配显示不全的问题，在对代码进行尺寸修改后，最终在手机 APP 上显示的效果如下所示。

标题界面

登陆界面

注册界面

当 APP 打开时，显示的是标题界面，此时点击登录后，出现登陆界面，登陆界面可以选择记住密码，下次登陆时可以不用输入密码（后端，小组成员工作）。忘记密码按钮此版本暂未实现。在标题界面如果点

击注册按钮，则会进入注册界面，输入用户名、手机号/邮箱、验证码和密码。所有的密码设置了 inputText 属性为 textPassword，显示时为圆点。

页面的切换采用 intent，代码如下所示。

```
// 从标题界面跳转至登陆界面
Intent intent = new Intent(Main.this,Login.class);
startActivity(intent);
```

登陆成功后，首先显示的是习惯的界面，点击下方底部栏的按钮可以更改页面中的 fragment，此时点击的按钮会变为橙色，原先的按钮变为黑色，实现页面切换的效果。

习惯界面

每日任务界面

待办事项界面

奖励界面

菜单界面

切换 fragment 的代码如下。

```
//fragment 切换 index 为切换的界面
private void InitFragment(int index){
    //使用 getSupportFragmentManager 获取
    FragmentManager fragmentManager = getSupportFragmentManager();
    //启动事务
    FragmentTransaction transaction = fragmentManager.beginTransaction();
    //将所有的 Fragment 隐藏
    hideAllFragment(transaction);
    switch (index){
        // 习惯界面
        case 1:
            if (habit_f == null){
                habit_f = new habit();
                transaction.add(R.id.frame_content,habit_f);
            }else{
                transaction.show(habit_f);
            }
            // 修改图标以及标题
            habit_iv.setImageResource(R.drawable.habit_icon_selected);
            setTitle("习惯");
            break;
        //待办事项界面
        case 2:
            if (matter_f== null){
```

```
                matter_f = new matter();
                transaction.add(R.id.frame_content,matter_f);
            }
            else{
                transaction.show(matter_f);
            }
            break;
        //每日任务界面
        case 3:
            if (task_f== null){
                task_f = new task();
                transaction.add(R.id.frame_content,task_f);
            }
            else{
                transaction.show(task_f);
            }
            break;
        //奖励界面
        case 4:
            if (reward_f== null){
                reward_f = new reward();
                transaction.add(R.id.frame_content,reward_f);
            }
            else{
                transaction.show(reward_f);
            }
            break;
        //菜单界面
        case 5:
            if (menu_f== null){
                menu_f = new menu();
                transaction.add(R.id.frame_content,menu_f);
            }
            else{
                transaction.show(menu_f);
            }
            break;
    }
    //提交事务
    transaction.commit();
}
```

底部按钮的切换 fragment 监听，由于代码类似，只展示其中一部分。

```
//fragment 切换监听
@Override
```

```
public void onClick(View v) {
    switch(v.getId()){
        case R.id.habit_iv:
            habit_iv.setImageResource(R.drawable.habit_icon_selected);
            // 切换 fragment
            InitFragment(1);
            setTitle("习惯");
            mySearch.setVisible(true);
            add_btn.setVisible(true);
            // fragment 标记
            fragment_flag = 1;
            break;
        // 剩余的 case 都类似，这里省略
        case ...:
}
```

切换时，需要修改顶部的按钮是否显示，底部的标题，以及一个 flag 变量记录当前 fragment 是哪一个。

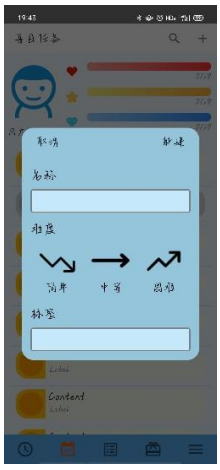
每个页面中存在 ListView 的 item 显示，这里我使用了自定义的 ItemAdapter 继承自 BaseAdapter，这里以待办事项 ItemAdapter 为例，代码如下。

```
public class MatterItemAdapter extends BaseAdapter {
    //测试代码 list
    private List<String> data;
    //测试代码 list
    private LayoutInflater layoutInflater;
    private Context context;
    public MatterItemAdapter(Context context, List<String> data) {
        //传入的 data，就是我们要在 listview 中显示的信息
        this.context = context;
        this.data = data;
        this.layoutInflater = LayoutInflater.from(context);
    }
    @Override
    public int getCount() {return data.size();}
    @Override
    public Object getItem(int position) {return null; }
    @Override
    public long getItemId(int position) {return 0; }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        //测试代码 listview inflate
        if(position == 2){
            View v =
LayoutInflater.from(context).inflate(R.layout.matter_or_task_item_finished, null);
            return v;
```

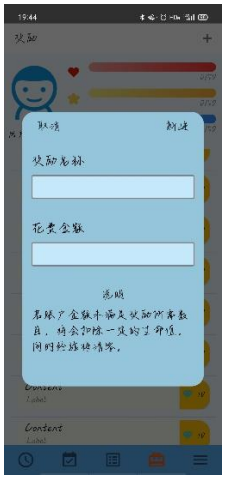
```
        }
        //测试代码 listview inflate
        View v = layoutInflater.inflate(R.layout.matter_or_task_item, null);
        return v;
    }
}
```

继承 BaseAdpater，需要重写其中的五个方法，在 getView(int position, View convertView, ViewGroup parent)方法中渲染 ListView 的 Item 样式，同时通过控制 position 的值，可以修改不同的样式。其余的 ItemAdapter 与这个类似，这里不再赘述。


增加习惯


增加每日任务


增加待办事项


增加奖励

点击加号后显示增加 item 的 dialog，如上图所示，不同的 fragment 会显示不同的增加样式，是由于上面提到的 fragment_flag 标记记录当前的 fragment。具体代码如下。

```
//增加按钮的监听
add_btn.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {
    @Override
    public boolean onMenuItemClick(MenuItem item) {
        switch (fragment_flag){
            case 1:
                showAddHabitDialog();
                break;
            case 2:
                showAddMatterDialog();
                break;
            case 3:
                showAddTaskDialog();
                break;
            case 4:
                showAddRewardDialog();
                break;
        }
        return true;
```

```
    }
});
```

对于这四个 dialog，每一个都需要写一个类继承 Dialog 类，由于四个类的实现都比较类似，这里用增加习惯 dialog 举例。代码如下。

```
public class AddHabitDialog extends Dialog {
    public AddHabitDialog(Context context) {super(context);}
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //界面初始化
        this.setContentView(R.layout.dialog_add_habit);
        //设置背景透明
        this.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
    }
}
```

setContentView()方法用来渲染 dialog，setBackgroundDrawable()方法用来使背景变为透明的。

特别的在增加待办事项的时候，最下方的选择日期 EditText 需要实现点击后让用户选择日期然后显示的效果，如下图所示。

这个功能在 java 代码中实现。需要注意的是在 dialog 类中调用 findViewById()方法时，一定是使用 this. findViewById()，否则没有办法获取到该控件，也没有办法设置监听。代码如下。

```
//日期选择
date_select_et = this.findViewById(R.id.matter_date_et);
date_select_et.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {getDate();}
});
```

在这之中的 getDate()方法是我自定义的方法，用于获取用户选择的日期。具体代码如下。

```
private void getDate(){
    //editview 显示选择日期
    Calendar calendar = Calendar.getInstance();
```

```
    //显示日历的 dialog
    DatePickerDialog dialog = new DatePickerDialog(context, new
DatePickerDialog.OnDateSetListener() {
        //选择日期的监听
        @Override
        public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
            int mYear = year;
            int mMonth = month;
            int mDay = dayOfMonth;
            // 更新 EditText 控件日期 小于 10 加前面 0
            date_select_et.setText(new StringBuilder()
                    .append(mYear)
                    .append("-")
                    .append((mMonth + 1) < 10 ? "0"
                            + (mMonth + 1) : (mMonth + 1))
                    .append("-")
                    .append((mDay < 10) ? "0" + mDay : mDay));
        }
    },calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH));
    dialog.show();
}
```

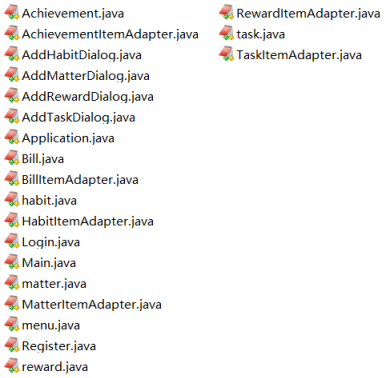
使用系统自带的类 Calender 以及 DatePickerDialog 实现此效果，通过用户的日期选择，EditText 自动显示所选择对应的日期格式。

成就界面与菜单界面的效果如下图所示,仍然是使用自定义的 ItemAdapter 渲染 ListView 中的 item，左上角返回按钮可以回到之前的程序主界面（不是标题界面）。

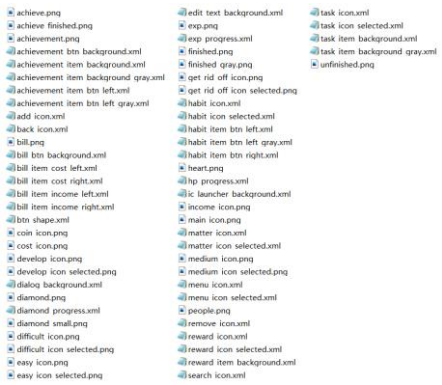
成就界面

账单界面

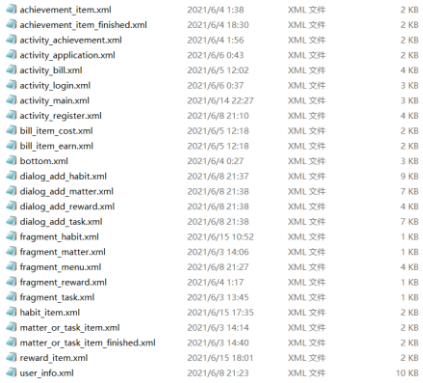
以下是这次 UI 基本文件结构。



java 文件



drawable 文件



layout 文件

menu 文件

5.个人小结

在这次的 Android 开发过程中，遇到了许多困难，也学习了很多新知识。在开始做 UI 之前，对其了解仅仅局限于每个控件的样式以及部分布局。在制作过程中，为了达到其他组员给我的设计需求，我查阅了许多网上的资料，阅读别人的布局代码，有的借鉴了，有的也提炼出自己想要的结果。一开始的时候，连最简单的两个控件在两边一个控件放在中间都做不好。经过不断的摸索，对每个布局的属性了解更加深刻了。有的时候为了贴合组员给的设计，仅仅是很小的一部分，比如只显示左侧的边框，我也在网上找了各种代码，并进行不断地尝试。同时，改 bug 也是不可或缺的一部分，在写 dialog 的监听时，由于之前从来没有用过 dialog 类，所以对于在 dialog 类中通过 findViewById()方法寻找控件时，尝试了各种各样的方法,直接找报空指针错误,通过 view 找报运行错误,后来经过不懈的尝试,发现了要用 this.findViewById(),这小小的四个字母的 bug 改了我一个晚上加一个早晨。不过最终还是成功的做完了所有的 UI，虽然 UI 制作看起来简单，但是当你真正去做的时候，你会发现其实这也是一门学问，如何做的好看，美观，十分重要。如果以后我还有机会参加 Android 开发项目，我希望我可以尝试一下后端的开发，毕竟什么方面都需要勇敢的去尝试一下嘛。

本人承诺该课程报告中不存在抄袭、剽窃或其他学术不端行为，并愿意承担因此类行为可能导致的一切后果。

承诺人（签名）：