**Android程序设计课程报告**

**设计题目:\_\_\_\_\_点餐平台App\_\_\_\_\_\_\_**

利用该点餐App 能实现登录、注册、注销功能，以及基本的选择店铺点餐，加入购物车和结算等功能。此外在页面上也做了一些美化。

下面我将分几个模块对该App的功能以及实现过程进行详细介绍。

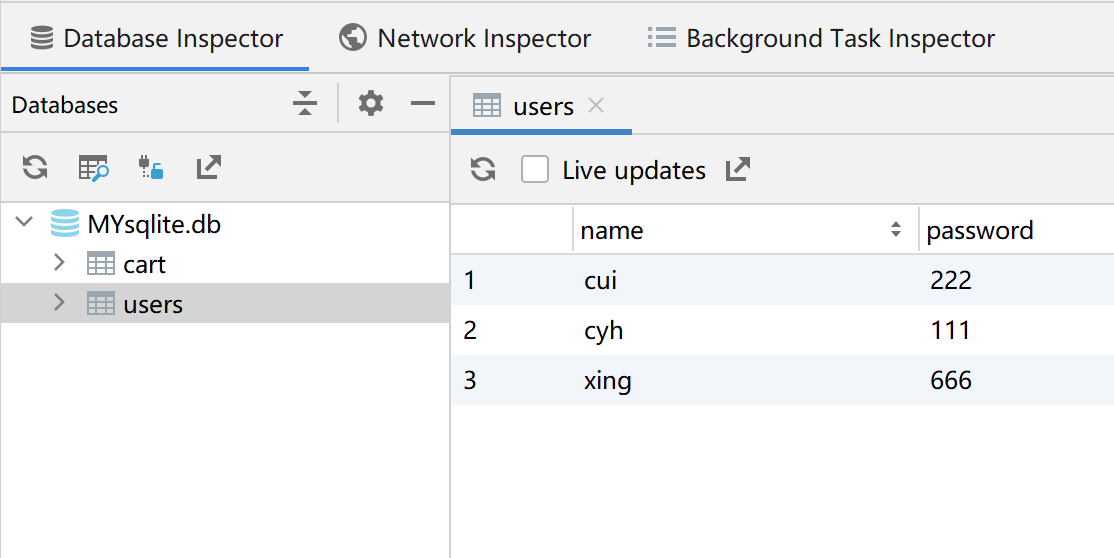
(包括文本介绍、重要代码与图片的展示，完整代码和演示效果见其他附件)

**一、登录注册与注销模块**

该模块相关演示效果如下图所示：

   ** **

（封面leadin） (登录login) (注册register) (我的page4)

** **

(注销logout) (数据库的users表)

**1. 用户类User**

该模块需要对用户的信息进行操作处理，所以首先需要定义一个用户类User，该类包括name和password两个属性，以及构造函数和相应的get、set方法。（具体代码见附件User.java）。

**2. 封面**

利用ImageView做了一个封面，对应附件leadin.xml；在相应的leadin.java代码中，利用Handler类初始化一个hander对象，调用postDelayed方法，在3秒后执行一个 Runnable 对象的 run() 方法，run()中利用Intent来跳转到登录页面login.xml。

**3.登录界面**

在login.xml中，利用线性布局制作了一个简单的登录页面，其中账号密码的输入使用了EditText的控件。在对应的login.java代码中，我对各个按钮进行了事件监听，比如点击登录，调用login()函数，点击注册，跳转到注册页面activity\_sign.xml和退出程序弹出警示框，提示是否要退出程序，如果点击确定就会退出程序，否则取消警示框。在login()函数中，需要对用户输入的信息进行相应处理，下文第6点会讲具体操作。

**4.注册页面**

编写一个activity\_sign.xml，其中的布局跟登录类似，只不过多了个确认密码的输入框，相关操作需要在对应的sign.java中编写代码来实现。首先是账号密码的合法性。这里我设置账号密码必须都不为空，且密码只能由数字构成，两个密码的输入需一致，账号不能和其他账号名重复才能注册成功。同时也对返回，注册按钮注册了点击跳转事件。

**5.数据库**

由于这里的登录，注册以及后面要实现的注销模块都涉及到对用户信息的众多处理，并且可能要管理多个账户信息，因此我使用的数据库对用户信息进行存储，使用SQLiteOpenHelper这个辅助类来管理SQLite数据库。我创建了zMySqlHelper这个类来继承SQLiteOpenHelper，并重写相关方法，从而对我数据库的信息进行管理。其中，我定义了登录函数login()和注册的函数register()，对传进来的User u用户信息进行相关处理。

5.1注册函数register()

拿到数据库db，开辟一个存键值对的空间

(1)重名情况处理：先u.getName()拿到用户名称，然后新建一个光标，利用query来扫视查询整个User表的每一行，如果发现某一行中的name与用户名称u.getName()相同的话，说明重名了，返回-1退出函数。

(2)合法情况判断：上面的重名情况不发生时，执行这个部分的代码。如果输入的账号密码都不为空，那么该情况合法，将用户的账号密码以键值对的形式，insert插入Users表中。



（register函数）

5.2登录函数login()

拿到数据库db，定义一个boolean类型的变量result，初始化为false。然后跟注册函数类似，也是利用Cursor光标扫视Users表的每一行，query查询有没有与u.getName()一样账号的行，如果有的话，对该行的password列的数值和传进来的u的password做对比，如果相同的话就说明登录成功，返回true给result。



（login函数）

5.3注销功能logout()

拿到数据库db，也是新建一个光标对Users表的没一行进行查询，拿到每一行的name和password两列的数据，分别和传进来的name、password字符串进行比较，都相等的话说明确实要删除的账号密码原本存在于数据库的Users表中，因此将该行数据在Users表中删除。返回true给result，否则返回false给result。



（logout函数）

**6.登录界面的login()代码逻辑**

在login.java中，定义函数login()，其中利用xml中的id获取输入框中的账号密码信息，将其作为参数，传入zMySqlHelper类的login()成员方法中并调用该方法，之后得到一个返回值。若返回值为true，则说明登录成功，执行第一个if分支，Intent跳转到相应的MainActivity.xml界面，若返回值为false，则说明登录失败，需要对登录失败的原因进行进一步判断，若输入框中的信息都为空，则利用Toast打印提示信息：“账号密码不能为空”，否则打印提示信息：“账号或密码错误”。

**7.注册界面的sign\_to\_login()代码逻辑**

拿到三个输入框的信息，将name和password作为User()有参构造函数的参数，新建一个用户对象。先对输入框进行信息判断，如果都为空就Toast打印提示信息：”账号密码不能为空”，如果后两个输入框的信息不一致就Toast打印：“两次密码输入不一致”。接着调用zMySqlHelper的register()方法，如果返回值是1，说明register()中由进行数据库写入，注册成功，否则就是在register()的执行过程中出现了重名情况，没有进行数据库写入，此时Toast打印：” 账号已存在，请重新输入”。

**8.page4页面 – 注销账号功能**

该页面是App的第四个分页 -- “我的”页面（前三个页面放在“首页以及店铺展示模块”进行介绍）。在这个页面中，不仅有基本的退出登录功能，退出到对应的登录界面login.xml，还对注销功能进行了具体实现。在点击注销功能后，用户会跳转到对应的界面，其中有两个输入框，用户需要输入当前的账号和密码进行注销，点击注销后，还会弹出警示框，提示用户注销功能不可恢复，是否继续选择注销，若用户点击取消则使对话框消失，点击确定则进行相应的注销流程。在相应的page4\_logout.java中，有对注销的各种情况进行不同处理。具体处理如下：

首先利用SharedPreferences获取当前账号（因为每次登录时都会使用SharedPreferences来获取，如果之前有账号，则新的账号信息会覆盖掉之前的内容，所以保证了获取到的信息永远是当前的账号信息），若用户输入的账号密码就能在数据库的Users表中查到，但是输入的账号不是当前账号，则会Toast提示“输入账号不合法”。如果输入的账号为当前账号，那么调用zMySqlHelper中的logout()方法来注销，返回一个boolean类型的值，如果该值为true，说明logout()执行时成功将账户信息从数据库中的Users表中删除，那么就提示用户“注销成功”，并跳转到登录页面；如果返回的值为false，那说明注销不成功，提示用户“账号或密码错误“。

**二、首页以及店铺展示模块**

**1.MainActivity页面**

该页面是“首页”页面。里面用了嵌套的滚动布局。外层是垂直的滚动布局。

上面区域是内层的水平滚动布局，放置3个视频，用户可以水平滚动，选择想要的美食宣传视频进行观看，每个视频都要基本的播放暂停，快进和快退的功能。

下面区域是用了网格布局，放置了精选店铺，包括店铺宣传图片和店铺名称，用户根据喜好点击相应店铺可以进行页面跳转，跳转到具体的店铺点餐页面。

（MainActivity页面） （page2页面）

**2.page2页面**

该页面是“店铺栏”页面。采用线性布局，做了一个店铺列表，里面包含所有注册到该平台的店铺。用户可以具体选择想要的店铺，点击跳转到店铺的具体点餐页面。

**3.page3页面**

该页面是“购物车”页面。若用户未进行点餐，则页面会显示购物车空空如也，让用户立即前往点餐，用户点击“立即点餐“的按钮，则会跳转到page2，用户可选择相应的店铺进行点餐。

若用户点过餐了，则跳转到该界面会显示具体的商品价格和数量等信息，具体内容会在下面的“加入购物车和结算模块”进行讲解。

（空的购物车page3） (有商品的购物车page3\_Cart)

**4.店铺具体页面**

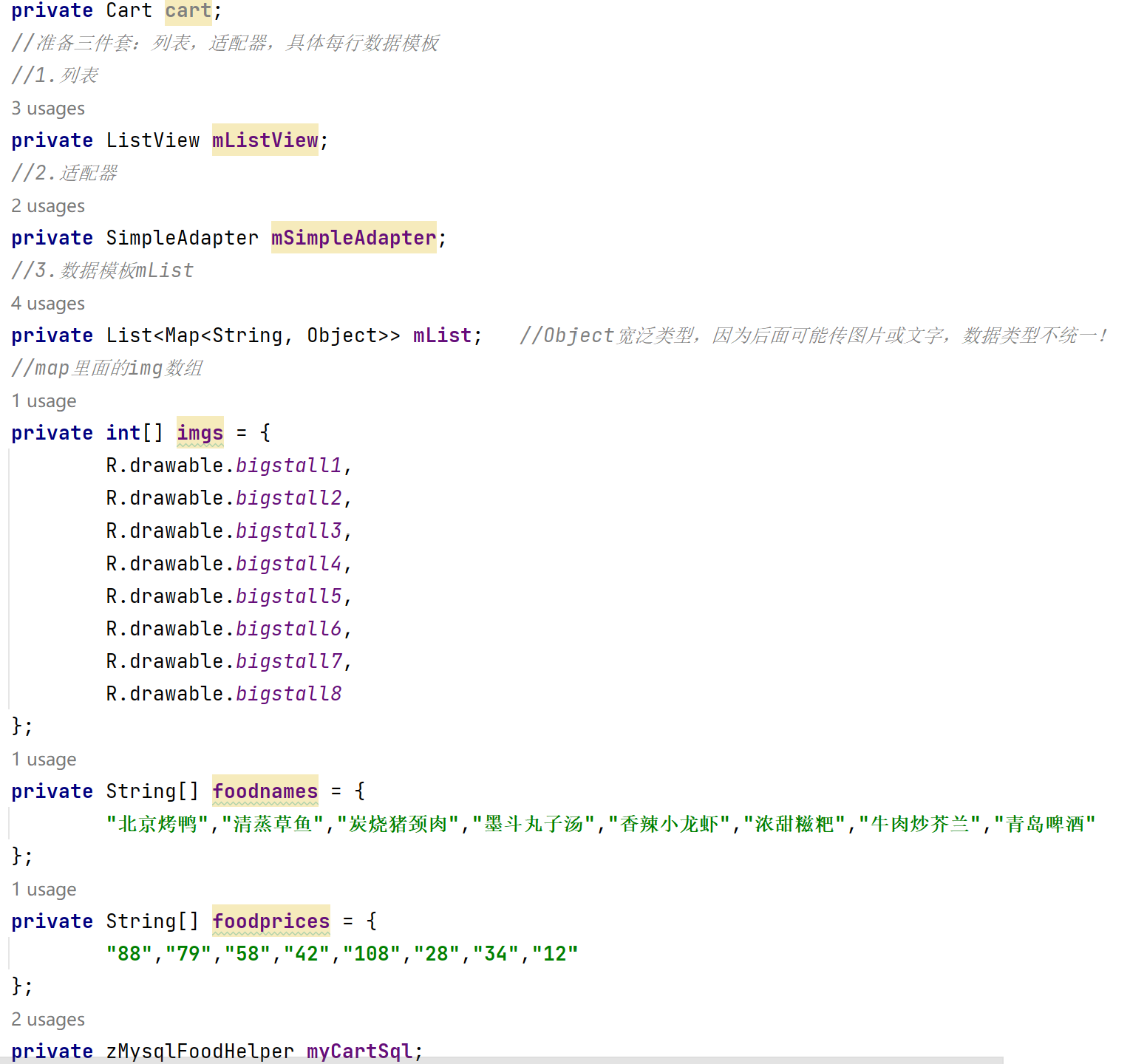
每个店铺都有各自的详情页，在详情页中，我使用的ListView来展示每道菜肴的具体信息，用户可以选择具体的每一道菜加入购物车，多次点击就会添加多份数量，点击后，购物车的商品数量也会相应增加，实时显示购物车的商品总数量。具体代码实现逻辑放到下面的“加入购物车和结算模块”进行详细讲解。

**5.店铺具体页面的信息写入**

我使用了适配器操作，将商品的图片、名称、价格分别写在三个数组中。

接着准备一个哈希表类型的列表List<Map<String, Object>> mList作为写入数据的模板，每个map都包括img、name、price三个键key和对应的值value。数据输入好后，将模板mList中的数据都写到msimpleAdapter适配器中，最后将这个适配器设置给主视图mListView，即可在页面中显示所有的商品列表信息。说得可能有点抽象，以下是具体代码：





**三、加入购物车和结算模块**

**1.商品类Product**

该类包含商品名称name、商品单价price、商品数量quantity三个属性，提供相应的get方法以及商品数量增减操作，在后面相关模块会对这些方法进行调用。

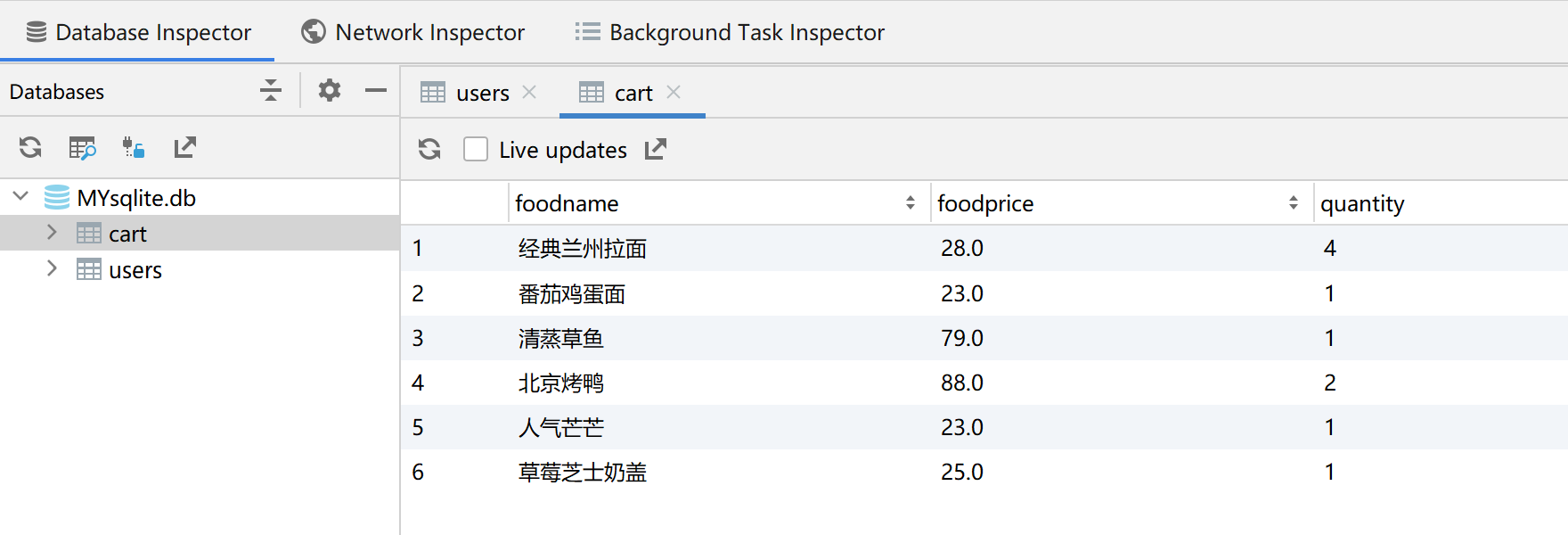
**2.购物车类Cart**

该类其实就是一个商品列表，列表的每个元素类型都为商品Product，提供相应的get方法以及计算总价的函数getTotalPrice()，在后面相关模块也会对这些方法进行调用。

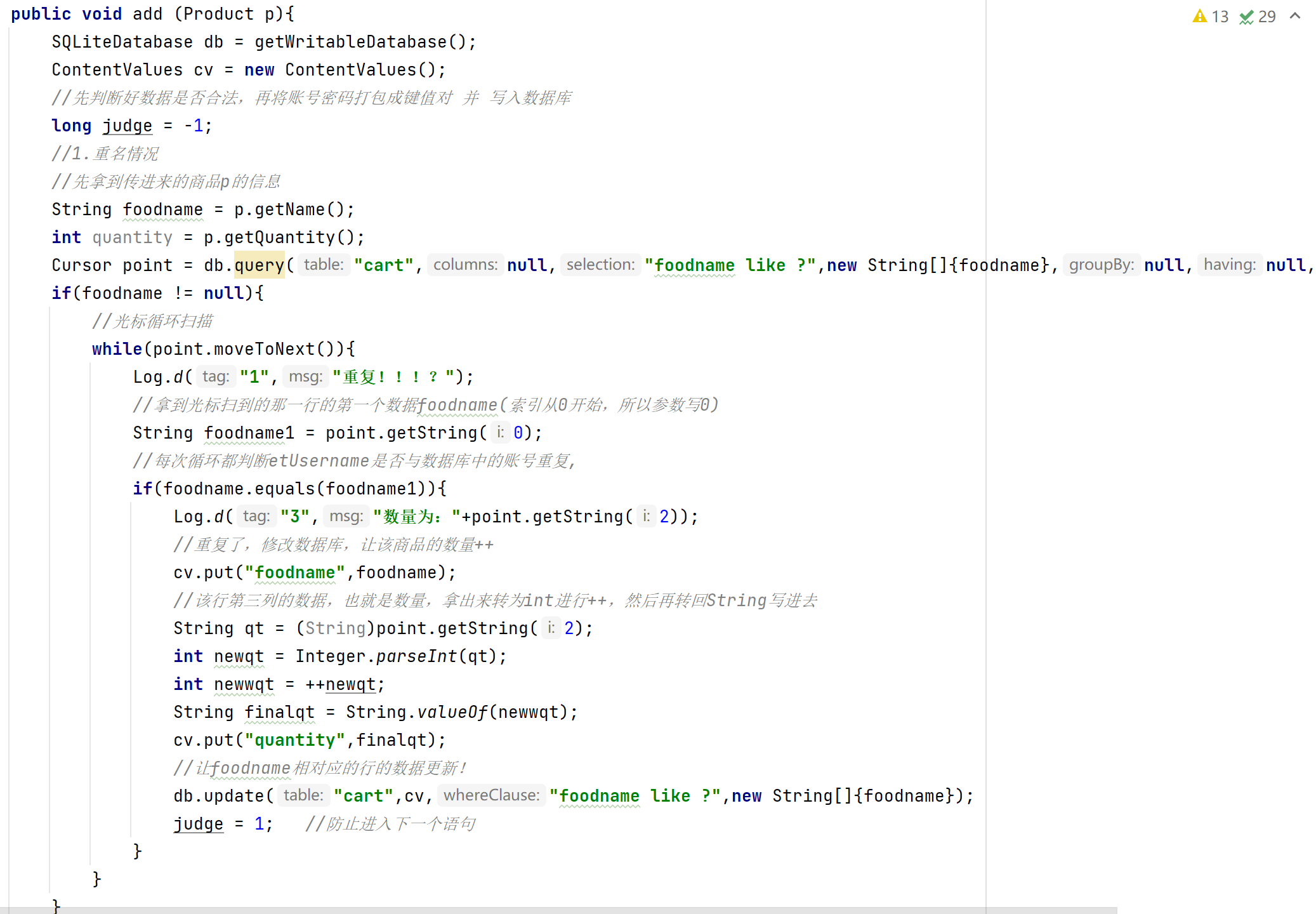
**3.数据库的Cart表**

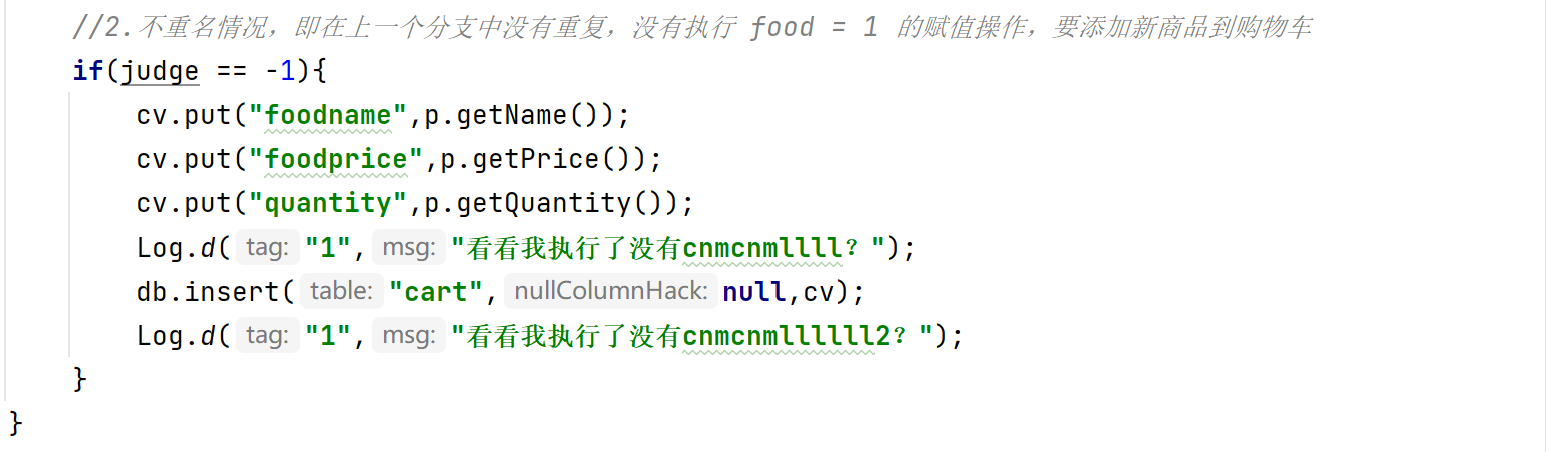
用户可以对不同店铺进行点餐，并同时加入购物车，最后一起结算，为了记录这些商品信息，以及后期对其进行相应的结算等处理，我在数据库中新建了一个Cart表，存储用户加入购物车的商品信息，其中包括商品名称foodname、商品单价foodprice和商品数量quantity。

我创建了zMysqlFoodHelper这个类来继承SQLiteOpenHelper，并重写相关方法，定义add()、dec()这两个方法处理相关信息，从而对我数据库的信息进行管理。



（数据库的cart表）

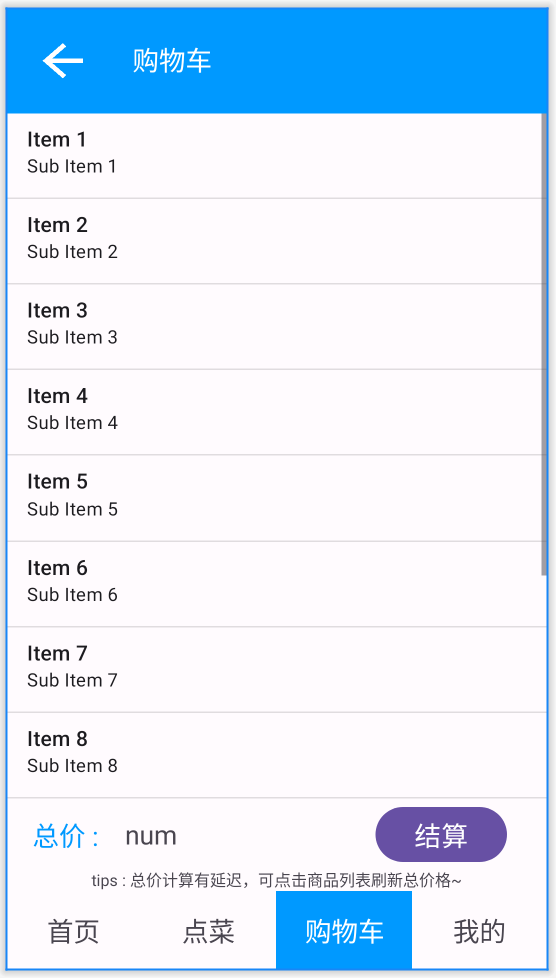
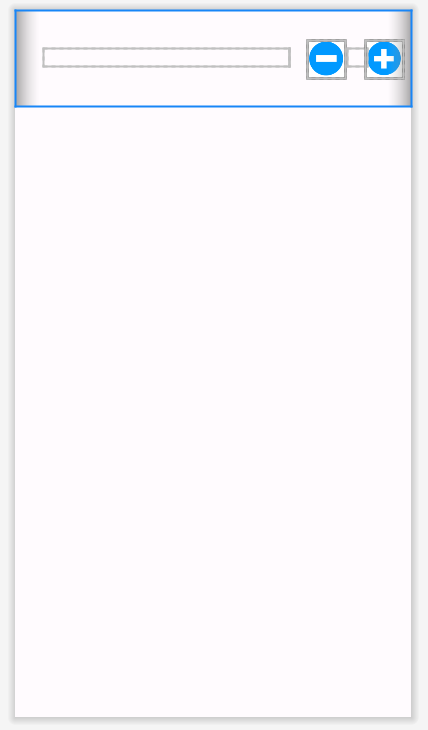




(add函数，dec函数跟这个类似)

**4.ListView布局**

在具体的购物车页面中，我也是采用了ListView布局，在每个item中，都有商品名称，商品单价，商品增加减少按钮，实时商品已点数量这些信息。

（整个ListView） (每个item)

**5.加入购物车**

当用户在店铺详情页面点击加入购物车时，add()函数会根据点击的item在ListView的location位置，拿到对应item的商品信息。若该商品在数据库的Cart表中不存在记录，则初始化该商品并加入数据库的Cart表中，记录商品名称，商品单价，商品数量初始化为1。若该商品已存在于购物车，则会对Cart表的相应商品数量进行+1的操作。

**6.购物车商品总数量**

当点击加入购物车时，还会到Cart表中进行遍历每一行，统计总共的商品数量，累加后返回，再setText写到页面上显示。

**7.增加商品数量**

当用户点击增加按钮，会调用数据库的add()方法，执行相关操作，且该操作的实现效果跟点击“加入购物车”按钮的效果是一模一样的，商品不存在就写入Cart表，存在就让数量+1。

**8.减少商品数量**

当用户点击减少按钮，会调用数据库的dec()方法，执行相关操作，逻辑跟add()是类似的，只不过如果商品数量>1时，点击后商品数量会-1，并且当商品数量为1时，用户点击减号按钮，此时则会将数据库的该商品信息删除，同时在购物车页面不显示该商品信息。

**9.计算商品总价**

当用户点击去结算时，会调用Cart类的getTotalPrice()方法，拿到所有Product商品并计算总价。之后，页面会弹出一个对话框，提示用户总价以及询问是否支付，若用户点击确定，则提示用户支付成功，同时清空购物车，数据库的Cart表的相应数据也要全部清除，页面跳转到page3，也就是购物车为空，提示用户去点餐的页面。若用户点击取消则关闭对话框，表示取消支付，购物车和数据库里的商品信息仍然保留。

(显示当前购物车内商品数量为10) (支付页面)

**10.商品数据更新**

当用户进行页面跳转，在其他店铺也进行点餐时，数据库的数据会进行更新，同时我在Cart.java中，写了数据库更新的代码，确保每次跳转到此页面时，都会先查询数据库Cart表中的数据，把最新的数据更新写在Cart.xml这个页面中。

同时，用户在点击商品增加和减少时，也应该更新商品数量和总价。数量我能通过add()、dec()方法的调用来及时更新，但是总价需要刷新页面才能更新，原因是我的总价并没有写在ListView中，而是另一个页面上在处理该信息，但是商品数量写在ListView中是可以用CartAdapter适配器来实时更新的，总价则不行。我尝试许多次，但最终仍无法实现点击+、-号即可同时更新商品数量和总价的代码，只能勉强做到点击ListView的item列表（除了+、-两个按钮之外的地方）就能实现总价刷新的效果。

**四、购物车适配器**

为了将item的数据显示到ListView上去，同时点击增加减少等按钮还能实现数据显示的动态更新，我给购物车写了个适配器CartAdapter，并在CartAdapter.java中重写getView()方法，对增加和减少等按钮设置事件监听，点击后会根据点击的position位置参数来会回调add()、dec()函数，这就是上述说“点击增加/减少按钮会根据位置调用相应方法的原因”，因为增加了适配器。同时使用notifyDataSetChanged()来监测数据集改动，确保改动后能立即更新ListView中相应item显示的商品数量。

具体代码分析：

首先在getView()传进来int position, View convertView, ViewGroup parent这三个类型的参数，主要使用前两个参数。position表示调用者的位置，通过该参数可以知道是listView中的哪个item触发了点击事件，拿到该item的Product信息。使用convertView.findViewById来找到每个item的五个控件，对其中三个控件，get商品的三个数据（商品名，单价，数量），以java的方式写入对应控件中；对另外+ - 两个控件，则设置点击事件监听，分别调用add()、dec()函数，再使用notifyDataSetChanged()来监测数据集改动。

相关代码如下：





**五、“我的”页面**

**1. “我的”页面-我的订单：**



**zMySqlHelper类实现SQLiteOpenHelper，利用onCreate方法创建订单表。**

private static final String *createOrders* = "create table MyOrder (" +  
 "uuid text primary key," +  
 "username text," +  
 "total\_price real," +  
 "time integer)";

public void onCreate(SQLiteDatabase sqLiteDatabase) {  
 //直接调用上面的String(封装好语句的字符串)来创建语句  
 sqLiteDatabase.execSQL(*create\_user*);  
 sqLiteDatabase.execSQL(*create\_cart*);  
 sqLiteDatabase.execSQL(*createFood*);  
 sqLiteDatabase.execSQL(*createOrders*);  
}

**实现Order类与OrderAdapter适配器**

public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
 View view;  
 ViewHolder viewHolder;  
 if (convertView == null) {  
 view = LayoutInflater.*from*(getContext()).inflate(R.layout.*acvitity\_my\_order\_item*,  
 parent,  
 false);  
 TextView priceText = view.findViewById(R.id.*order\_price*);  
 TextView timestampText = view.findViewById(R.id.*order\_timestamp*);  
 viewHolder = new ViewHolder(priceText, timestampText);  
 view.setTag(viewHolder);

} else {  
 view = convertView;  
 viewHolder = (ViewHolder) view.getTag();  
 }  
 Order order = getItem(position);  
 if (order != null) {  
 viewHolder.priceText.setText(String.*format*(  
 getContext().getString(R.string.*OrderPrice*),  
 order.getTotal\_price()  
 ));  
 viewHolder.timestampText.setText(String.*format*(  
 getContext().getString(R.string.*OrderTimestamp*),  
 *timestampToString*(order.getTimestamp())  
 ));  
 }  
 return view;  
}

**将订单信息读入内存**

private void initMyOrder() {  
 // 获取用户名  
 SharedPreferences preferences = getSharedPreferences("MyPrefs", *MODE\_PRIVATE*);  
 String username = preferences.getString("username", "");  
  
 SQLiteDatabase sqLiteDatabase = new zMySqlHelper(this).getReadableDatabase();  
 Cursor cursor = sqLiteDatabase.query("MyOrder",  
 null, "username = ?", new String[]{username},  
 null, null, null);  
 if (cursor.moveToFirst()) {  
 do {  
 String uuid = cursor.getString(cursor.getColumnIndexOrThrow("uuid"));  
 long timestamp = cursor.getLong(cursor.getColumnIndexOrThrow("time"));  
 double price = cursor.getDouble(cursor.getColumnIndexOrThrow("total\_price"));  
 my\_order.add(new Order(uuid, price, timestamp));  
 } while (cursor.moveToNext());  
 }  
 my\_order.sort(new Comparator<Order>() {  
 @Override  
 public int compare(Order o1, Order o2) {  
 return Long.*compare*(o2.getTimestamp(), o1.getTimestamp());  
 }  
 });  
 cursor.close();  
}

**使用ListView来显示订单信息**

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_my\_order*);  
  
 TextView order\_back = findViewById(R.id.*my\_order\_back*);  
 order\_back.setOnClickListener(view -> finish());  
  
 initMyOrder();  
 ListView my\_order\_list = findViewById(R.id.*order\_list*);  
 my\_order\_list.setAdapter(new OrderAdapter(this, R.layout.*acvitity\_my\_order\_item*,  
 my\_order));  
  
}

**2. “我的”页面-用户图标：**

**注册头像点击事件**

avatarImageView = (ImageView) findViewById(R.id.*avatarImageView*);  
Bitmap bitmap = *readImage*(this, username);  
if (bitmap != null) {  
 avatarImageView.setImageBitmap(bitmap);  
} else {  
 Log.*d*("bitmap", "null");  
}  
avatarImageView.setOnClickListener(view -> {  
 Intent intent = new Intent(Intent.*ACTION\_PICK*, MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*);  
 startActivityForResult(intent, *REQUEST\_CODE\_SELECT\_IMAGE*);  
});

**回调函数，将选择的图片转换为位图，并且保存到私有文件位置**

@Override  
public void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 switch (requestCode) {  
 case *REQUEST\_CODE\_SELECT\_IMAGE*:  
 if (resultCode == *RESULT\_OK* && data != null) {  
 Uri selectedImageUri = data.getData();  
 Bitmap bitmap = null;  
 // 将头像转为bitmap  
 try {  
 bitmap = MediaStore.Images.Media.*getBitmap*(this.getContentResolver(), selectedImageUri);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 // 保存头像到username.png  
 assert bitmap != null;  
 *saveImage*(this, bitmap, username);  
 avatarImageView.setImageBitmap(bitmap);  
 }  
  
 }  
}

**保存图片与读取图片实现**

private static boolean saveImage(Context context, Bitmap bitmap, String imageName) {  
 File directory = context.getFilesDir();  
 File file = new File(directory, imageName + ".png");  
  
 try (FileOutputStream out = new FileOutputStream(file)) {  
 bitmap.compress(Bitmap.CompressFormat.*PNG*, 100, out);  
 return true;  
 } catch (IOException e) {  
 e.printStackTrace();  
 return false;  
 }  
}  
  
// 读取图片  
private static Bitmap readImage(Context context, String imageName) {  
 File directory = context.getFilesDir();  
 File file = new File(directory, imageName + ".png");  
  
 Bitmap bitmap = null;  
  
 if (file.exists()) {  
 try {  
 bitmap = BitmapFactory.*decodeFile*(file.getPath());  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 return bitmap;  
}

**3. “我的”页面-用户名称：**在登录后，利用SharedPreferences来保存当前账号信息，然后拿到账号名称的文本，写入对应的TextView控件中。具体效果如下图，用户名称为1。

****

//获取当前登录的账号username  
SharedPreferences preferences = getSharedPreferences("MyPrefs", *MODE\_PRIVATE*);  
username = preferences.getString("username", "");

showname.setText("欢迎您，" + username + " !");

**4. “我的”页面-相关帮助：**点击进去后会显示相关的帮助文本信息。

**5. “我的”页面-相关推荐：**点击后会跳到首页，首页有进精选视频和相关美食店铺推荐。

**六、修改漏洞**

初步包装好整个App，并对可能出现的bug进行多次修改，确保点击页面上的任何一处地方都不会出现明显的逻辑漏洞，同时对一些代码进行优化。

**修改逻辑漏洞有：**

**1. 点击返回键会跳转会相应页面**

比如所有店铺在点击“去结算”都会跳转到购物车页面，但是点返回则不知道要怎么跳回上一次点餐相应的店铺页面。此时我让每个店铺在点击“去结算”时，都在跳转前打包好bundle对象，putExtra传到下一个页面，这样下一个页面getIntent便知道时是哪个页面跳转过来的。要点返回跳转回去点餐的店铺时，只需利用bundle中的信息就能判断出应该跳转到哪个店铺页面。

**2. 各种数据类型转化报错**

比如String中的数字字符与其他地方的整数混淆，页面上的数字要进行计算得先转为int再转回String等，这些我经过了大量的Log.d()调试打印，才把它弄得完善，并能正确运行起来。

**3. 变量作用域问题**

一般情况下，获取的元素都设置为局部变量，但是如果在onCreate()方法外设置别的方法且又要用到元素，则要设置全局变量。一开始我为了方便，全部设置为全局变量，但是考虑到可能产生命名冲突，难以测试，安全性和可维护性都下降等问题，我最后还是尽可能使用局部变量，有些必须使用全局再使用全局。

**代码优化有：**

1. 由原来写固定的LinearLayout布局转为使用ListView列表布局，并使用适配器，提供更灵活的功能实现方式。

2. 用lambda表达式来写事件监听代码，使代码更简洁。（不过整个项目的几千行代码中，还是有大量重复的代码。我目前能力有限，没能想到好的设计模式来进行优化）