3 系统需求分析和可行性分析

本系统是电子拍卖的手机客户端应用，客户可以通过手机客户端进行拍品查询与竞拍，管理员也可以通过手机客户端进行管理操作。具体实现目标：

（1） 拍卖预展，用户可以浏览正在或者即将拍卖的物品。

（2） 信息发布，管理员可以发布拍卖物品。

（3） 竞价交易， 用户可以对拍卖物品竞价并交易。

（4） 查看竞标信息，用户可以查看正在拍卖物品的竞标信息。

（5） 查看公告、财务管理、系统设置等。

3.1 系统功能概述

基于 Android 的电子拍卖系统主要功能有 Android 客户端功能以及服务器端功能。这两套功能分别应用于 Android 前端（客户端）以及后端服务器。系统架构图如图 3-1 所示：

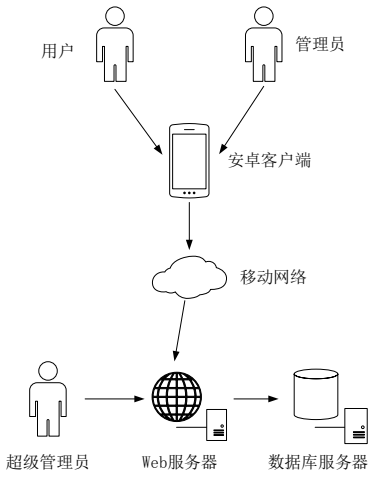


图 3-1 系统架构图

如上图所示，用户与管理员均可以在任意时间地点通过客户端的电子拍卖应用进行竞拍、浏览信息以及管理操作。用户或管理员使用安卓客户端通过移动网络向 Web 服务器发送请求，服务器收到请求后根据请求信息对数据库进行增、删、改、查等操

作并响应客户端的请求。超级管理员可以通过 Web 服务器，对管理员信息进行管理[14]。

3.1.1 用户客户端功能简介

基于 Android 的电子拍卖系统的客户端应用分为两个部分，分别分为面向用户部分以及面相管理员部分。面向用户部分功能[15-17]有：

（1） 用户注册功能 :用户首次进入应用, 需要根据自身信息准确填写注册账户。

（2） 用户登录功能：用户进入应用后，根据已有的账户进行登录，账户与密码信息无误即可登录客户端应用。

（3） 拍卖预展功能：用户登录客户端应用后，可以浏览查看到正在拍卖或者即将拍卖的物品, 点击任意物品项后, 即可查看拍卖物品的具体详细信息或者参与竞拍。

（4） 拍卖物品搜索功能：用户可以通过关键字搜索到自己想要的拍卖物品。

（5） 竞价交易功能：用户可以对正在拍卖的物品进行竞拍交易，支付竞拍保证金。

（6） 查看竞标信息功能，用户可以查看到正在拍卖物品的竞标信息。

（7） 订单功能，用户可以查看竞得物品，并支付剩余款项。

（8） 系统设置功能：用户可以更新自己账号详细信息和退出账户。

3.1.2 管理员客户端功能简介

基于 Android 的电子拍卖系统的客户端应用的面向管理员部分可以进行信息发布等操作，满足管理员需求。具体功能有：

登录功能：使用管理员账号登录，自动判断，进入管理员界面。

(2) 拍卖物品发布功能：管理员可以对拍卖物品进行编辑并发布。

(3) 拍卖物品查看功能：管理员能够查看到正在拍卖或者即将拍卖的物品。

(4) 用户管理功能：管理员可以对用户账户进行管理。

(5) 财务管理功能：管理员可以对平台拍卖量进行概览。

(6) 系统功能：管理员可以退出账户。

3.2 可行性分析

基于 Android 的电子拍卖系统，客户端选用 Android 系统，Android 采用 Java 为其主要开发语言，Java 存在大量的方法库可以为开发提供支持。使用官方的开发工具 Android Studio，比使用 Eclipse 等开发工具更加方便和具有效率。Flask 是一个轻量级 Web 应用框架，其开发语言为 Python，Python 开发效率比 Java 更高，Flask 具有灵活、轻便、扩展性强等特点，更加适合部署轻量级 App Web 服务器，因此选择 Flask 为本系统后台端。本系统数据库选择为目前流行并且免费的 MySQL 数据库，使用 PyMySQL 进行后台服务器和数据库的交互。使用 Json 为安卓客户端与后台服务器交互的载体。技术实现上几乎不存在问题，在技术上是可行的。

本系统开发工具为 Android Studio 和 Pycharm，数据库选用 MySQL，经济付出几乎为 0，在经济上是可行的。

本系统仅仅是一个简单的电子拍卖系统，资源销毁较少，对服务器和安卓客户端的性能要求较低，环境搭建简单，运行是可行的。

5 系统开发与实现

5.1 异步通信

使用开源的 AsyncHttpclient 网络请求框架替换 Android 默认的网络请求框架向后台服务器发送 Post 请求，以 Json 作为服务器与 Android 客户端之间数据交互的数据传输格式。

Android 端主要伪代码如下：

```
//创建 AsyncHttpClient 实例
AsyncHttpClient client = new AsyncHttpClient();
//发送 post 请求
client.post(this, "请求地址路径", 数据实体, "传输类型(application/json)", new JsonHttpResponseHandler() {
                    //请求成功
                @Override
                public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
                    super.onSuccess(statusCode, headers, response);
                    // response 为服务器响应回的数据
                }
```

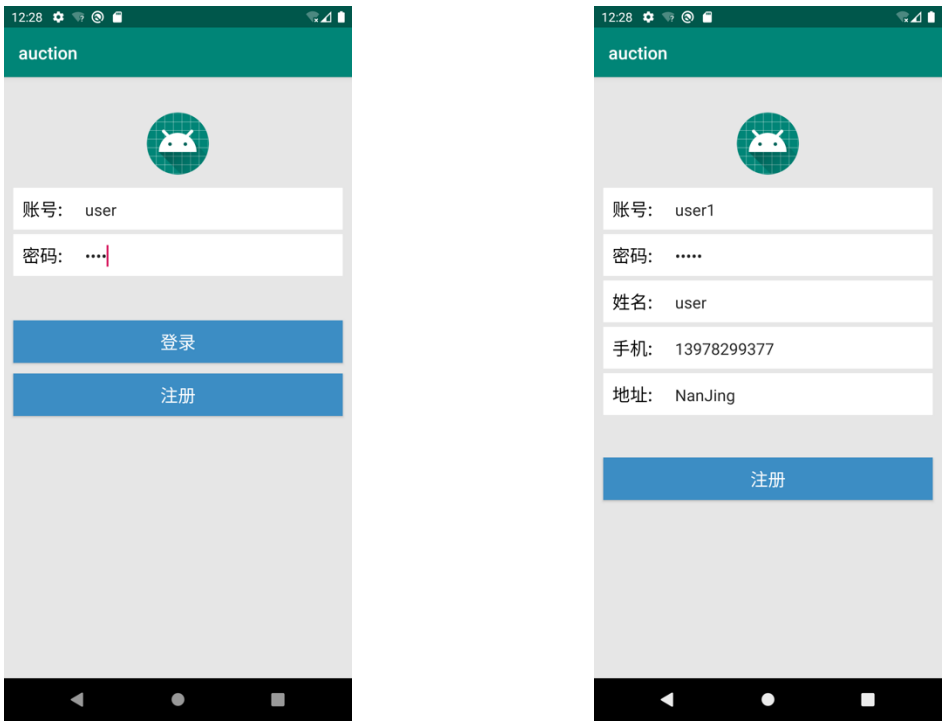
服务器端主要伪代码如下：

```
#定义路由
@app.route('请求路径', methods=['POST'])
def  函数名():
```

```
#获取请求数据
 req_data = request.get_data()
 req_data = json.loads(req_data)  #  将 json 转换为字典
 '''
 对数据进行相关逻辑操作
 '''
 return jsonify(res_data) #返回想要数据 jsonify 将字典转化为 json
```

5.2 安卓客户端的登录与注册界面的实现

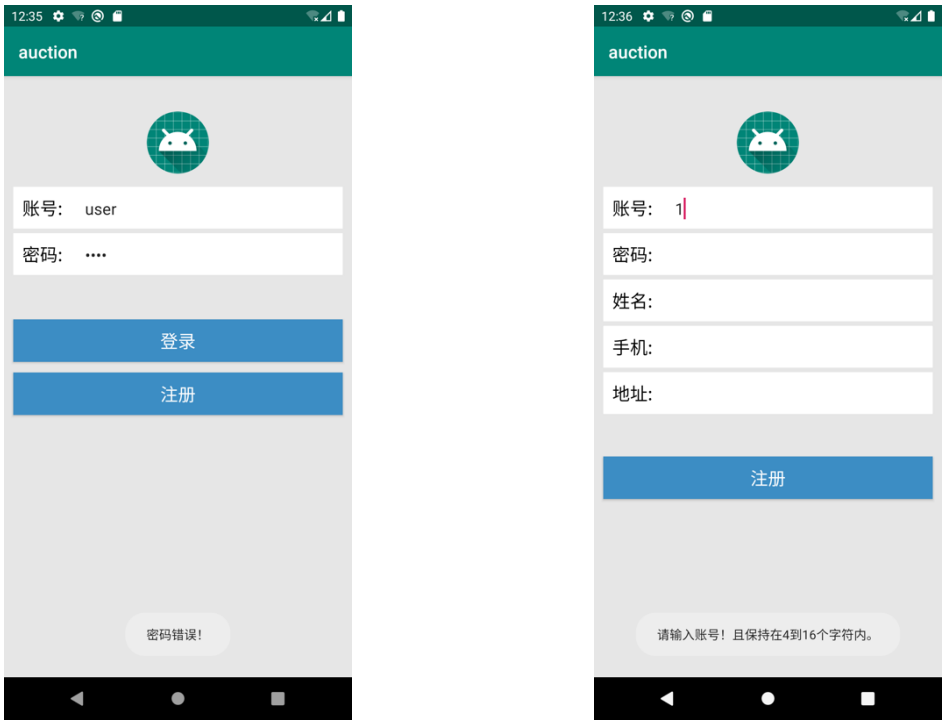
在安卓客户端打开应用时，首先显示的界面就是登录与注册的界面。用户使用用户账户登录，即可进入拍卖的主页，如何没有注册需要先注册，客户端的注册功能为注册用户账户。当管理员使用管理员账户登录时，即可进入管理界面，管理员账户需要超级管理员在后台添加。登录与注册界面分别如图 5-1 所示。



(a)　　(b)

图 5-1 登录注册示例图

在登录界面时，使用账户登录，如果密码与数据库密码不符合，会提示密码错误。在注册界面，注册账户时如果信息提供的不正确，比如账户填 1，其他不填，也会提示相应的信息。如图 5-2 所示。



(a) (b)

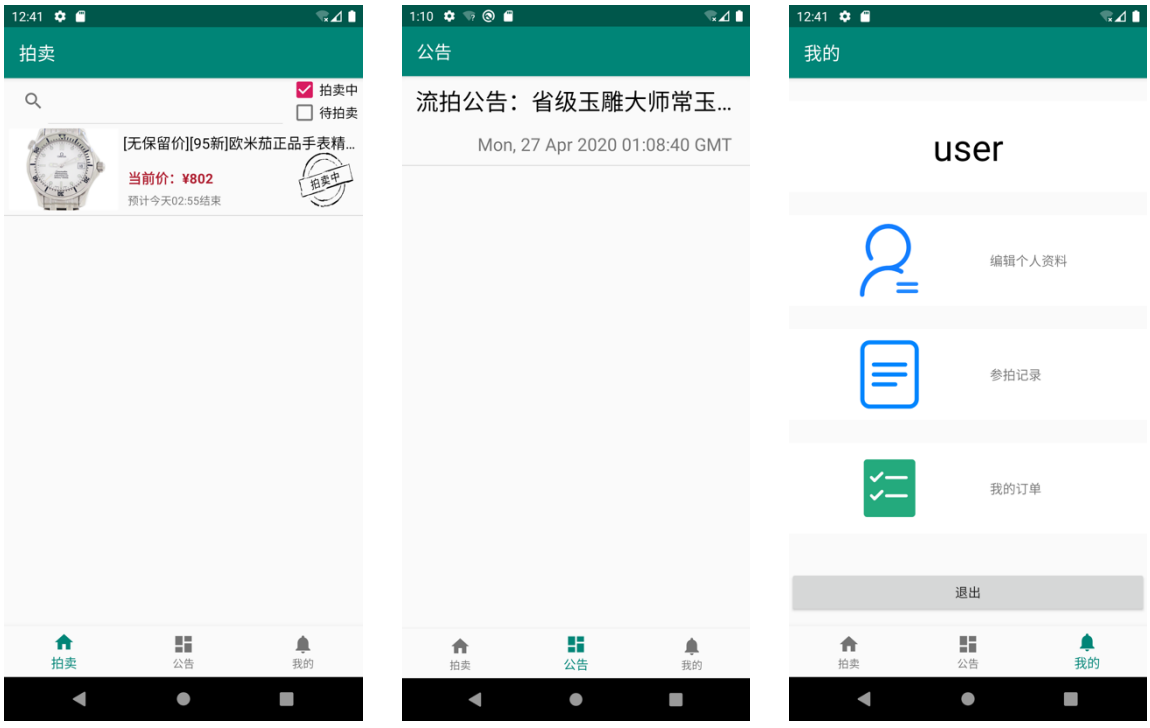
图 5-2 登录注册失败图

登录界面与注册页面的实现原理基本相同，基本由 EditText 和 Button 组成，EditText 负责收集信息。当点击 Button 时例如登录界面，将 EditText 信息加入 json 通过 AsyncHttpClient 网络框架向后台发出请求，后台接收到 json，解析出 json，向数据库查询是否有相同账户并比较密码是否相同，并响应相应的 json，AsyncHttpClient 接收到 json 解析出相应信息判断是否登录成功，如果登录成功则写入本地数据库 user 表，然后通过 Intent 转跳到下一个 Activity，然后销毁掉当前 Activity。其中密码的选项为了安全性，在输入后通过 MD5 加密封装进 json，因为在注册时也是同样 MD5 加密并写入数据库，登录时也要进行 MD5 加密直接查询匹配。因为 MD5 加密较容易难解密的特性，即便被得知 MD5 的密码，也登录不了。注册界面当点击注册 Button 后

会匹配 EditText 的信息长度是否达到相应的长度，如果满足条件将进行下一步骤。之后流程与登录界面类似，通过 AsyncHttpClient 网络框架向后台发出请求，得到响应后解析并判断，进行下一步骤。

5.3 拍卖界面的实现

安卓端用户拍卖的主界面，是一个 Bottom Navigation Activity，底部有三个按钮，拍卖、公告和我的。如图 5-3 所示。此界面默认的 Fragment 是拍卖 Fragment，点击其他按钮，就会转到其他 Fragment。拍卖 Fragment 由 ListView，SearchView，CheckBox 组成。



(a) (b) (c)

图 5-3 用户主页面示例图

其中 CheckBox 有两个分别为拍卖中和待拍卖，拍卖中是默认选中的，用于筛选 ListView 显示的内容。对于其不同的状态，使用不同参数向后台发送请求，或者相应的 ListView 数据。CheckBox 主要代码如下：

```
boolean isCd1 = true;
```

```
boolean isCd2 = false;
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    switch (buttonView.getId()) {
        //点击 checkBox1，拍卖中，如果被上勾 isChecked 为 true，否则为 false
        case R.id.checkBox1:
            isCd1 = isChecked;
            break;
            ////点击 checkBox2，拍卖中，如果被上勾 isChecked 为 true，否则为
false
        case R.id.checkBox2:
            isCd2 = isChecked;
            break;

    }
    if (isCd1) {
        if (isCd2) {
            //都有，拍卖中 and 拍卖中
            mViewModel.getdata("3");
        } else {
            //拍卖中
            mViewModel.getdata("1");
        }
    } else {
```

```
            if (isCd2) {
                //待拍卖
                mViewModel.getdata("2");
            } else {
                //null
                mViewModel.getdata("0");
            }
        }
        //加载资源动画可见
        loading.setVisibility(View.VISIBLE);
}
```

总共有 4 种状态，即只选中拍卖中，只选中待拍卖，拍卖中与待拍卖都选中，和都不选中。4 中状态对应不同的参数，向后台请求相关数据。拍卖中与待拍卖都选中如图 5-4 所示。

图 5-4 拍卖中与待拍卖选中示例图

点击 SearchView，输入关键字，可以模糊查询到相关的数据，在 CheckBox 的基础上进一步加参数。安卓端代码如下：

```
public boolean onQueryTextSubmit(String query) {
    //query 为搜索的参数
    if (isCd1) {
        if (isCd2) {
            //都有，待拍卖+拍卖中
            mViewModel.Sgetdata("3+" + query);
        } else {
            //拍卖中
            mViewModel.Sgetdata("1+" + query);
        }
    } else {
        if (isCd2) {
            //待拍卖
            mViewModel.Sgetdata("2+" + query);
        } else {
            //null
            mViewModel.Sgetdata("0+" + query);
        }
    }
    //加载资源动画可见
```

```
        loading.setVisibility(View.VISIBLE);


        return true;
}
```

当搜索时，安卓端向后台发出请求，例如搜索关键字 t，后台收到请求："POST /auction/SgetAuction/3+t HTTP/1.1" 200 -，参数为 3+t。后台端接收到参数后的代码如下：

```
        # 切片，例如 args==1+s，str=s，args=1
        str = args.split('+')[1]
        args = args.split('+')[0]
        num = 0
        s = ""
        #如果 str 有空格，继续切片，s 为模糊查询，加下面的 sql 语句中
        for i in str.split():
            num = num + 1
            if (num > 1):
                s = s + ' or '
            s = s + 'commodity_name like ' + '\'' + "%" + i + "%" + '\''
        #数据库连接
        connection = pymysql.connect("localhost", "root", "123456", "Auction")
        #根据 args 的参数，args 为 CheckBox 的状态标准，进行 sql 查询，使用 pandas 库的 read_sql 进行查询，获得查询结果
        try:
```

```
        if args == "1":
            sql = 'select * from commodity where now() > start_time and now() < end_time and (' + s + ')'
            data = pd.read_sql(sql=sql, con=connection)
            data['is_cd'] = '1'
        #data 就是查询的结果，这里只举例了 args 为 1 的一种情况
```

当搜索关键字为 A，待拍卖与拍卖中均勾上，搜索的结果如图 5-5 所示。

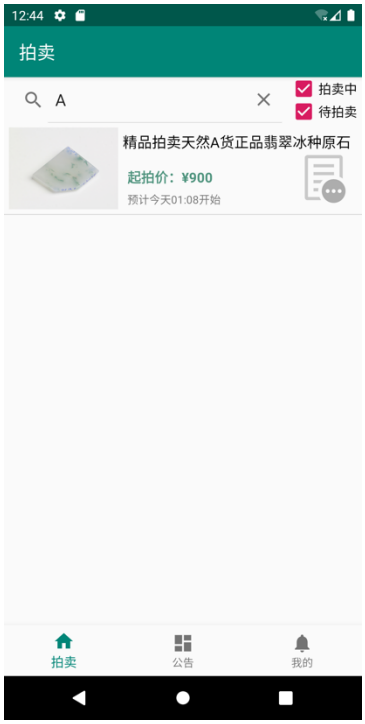


图 5-5 搜索结果示例图

5.4 拍卖物品详情页的实现

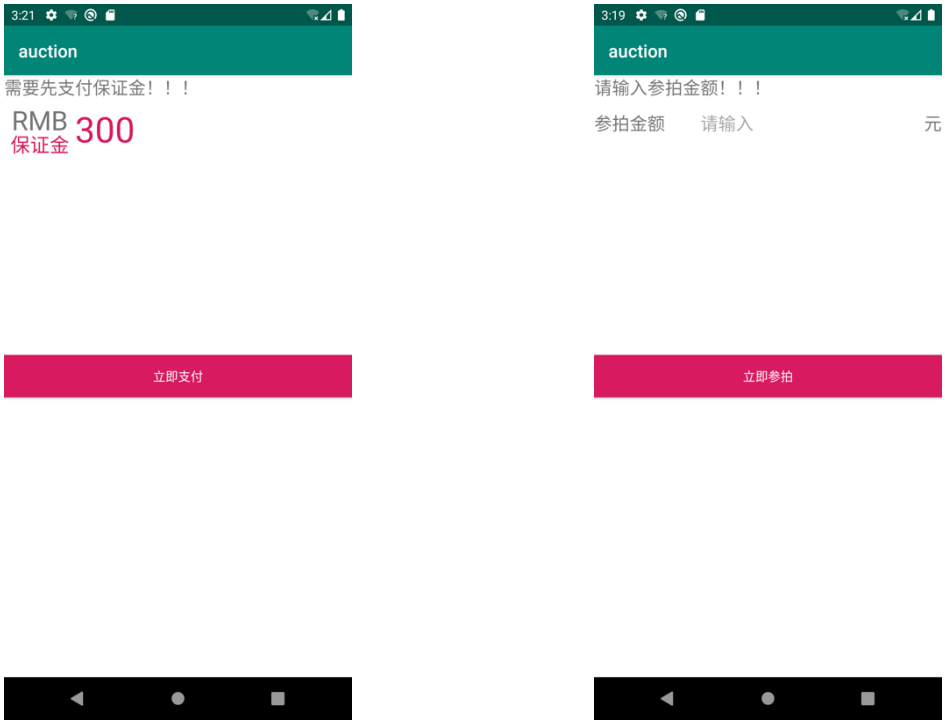
此页面采用相对布局进行排班，从上到下为由 SmartImageView、TextView、WebView 和 Button 组成，在此界面中，用户可以查看拍卖物品的图片、标题、竞标详情以及商品详情，用户也可以参与竞拍并且可以实时看到竞拍价和倒计时如图 5-6 所示。

图 5-6 拍品详情页面示例图

当中的拍卖记录模块，在当前页面只默认显示价格最高的 3 个，实现原理是向数据库查询本拍卖品的参拍记录并倒序输出，点击拍卖记录，可以看到全部的记录，是使用 listview 实现。拍品的详情展示是直接使用 webview 读取服务器的 html 文件，因为管理员创建拍卖品编辑的时候是使用富文本进行编辑的，直接保存为 html 文件，方便取读。

5.5 竞价交易功能的实现

用户在详情页面点击立即参拍的时候并且此拍卖物品是处于拍卖中的状态时，即可跳转到竞拍支付的页面，如果用户没有支付保证金，显示支付保证金，如图 5-7-a 所示。支付完保证金后，即可显示竞拍界面，参拍的金额需要大于此拍品的当前最高价格，如图 5-7-b 所示。



(a) (b)

图 5-7 保证金与参拍界面示例图

当此拍卖物品到终拍时间后，系统会自动生成竞拍成功的用户的订单或者是否流拍，并发布相应的公告。其核心代码如下所示：

```
while True:
    #先将成功拍卖的和未拍卖的商品 id 查询并放入 list 与 list1 列表中，这边略
        if temp[i]['commodity_id'] not in list1 and temp[i]['commodity_id'] not in list:
            # 倒序查询流拍的拍品在拍品拍卖记录订单表中的数据
            sql = 'select * from commodity_order where commodity_id = \'' + temp[i][
                'commodity_id'] + '\'' + ' order by id desc'
            data = pd.read_sql(sql=sql, con=connection)
            # 转为字典
```

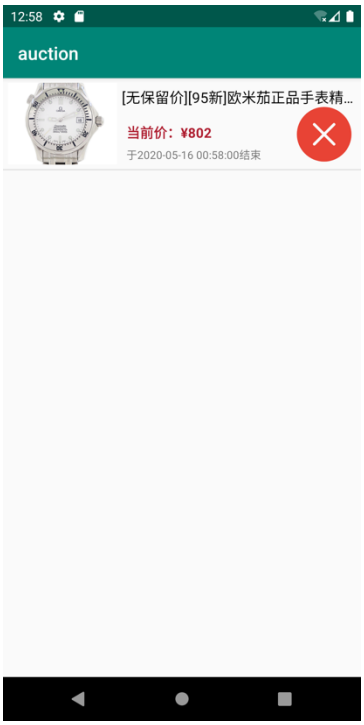
```
                temp2 = data.to_dict(orient='index')
                # 结果大于等于 1
                if len(temp2) >= 1:
                    try:
                        # 将出价最高的 userid 和 price 插入用户订单表
                        with connection.cursor() as cursor:
                            sql= 'insert into user_order(order_id,commodity_id,
                                user_id,price,pay) values(\'' + str(uuid.uuid1())
                                + '\',\'' + temp2[0]['commodity_id'] + '\',\''+
                                temp2[0]['user_id'] + '\',' + str(temp2[0]['price'])
                                + ',0)'
                            cursor.execute(sql)

                        connection.commit()
                        # 成交公告
                        notice(1,                    temp[i]['commodity_name'],
temp2[0]['user_name'])
                    except:
                        print("插入用户订单表出错")
                else:
                    try:
                        # 将出价最高的 userid 和 price 插入用户订单表
                        with connection.cursor() as cursor:
```

```
                    sql = 'insert into
unsuccessful_order(commodity_id,commodity_name,time) values(\'' + temp[i][
                        'commodity_id'] + '\',\'' + temp[i][
                            'commodity_name'] + '\'' + ',now())'
                    cursor.execute(sql)
                connection.commit()
                # 流拍公告
                notice(0, temp[i]['commodity_name'], '')
            except:
                print("插入流拍表出错")
            print(temp[i]['commodity_name'] + "-->流拍")
```

当到达拍卖的终止时间后，用户在客户端也可以看到拍卖的结果如图所示，并且在参拍记录里也可以看到拍卖的商品如图 5-8-b 所示。

(a)



(b)

图 5-8 拍卖结束示例图

5.6 管理员界面的实现

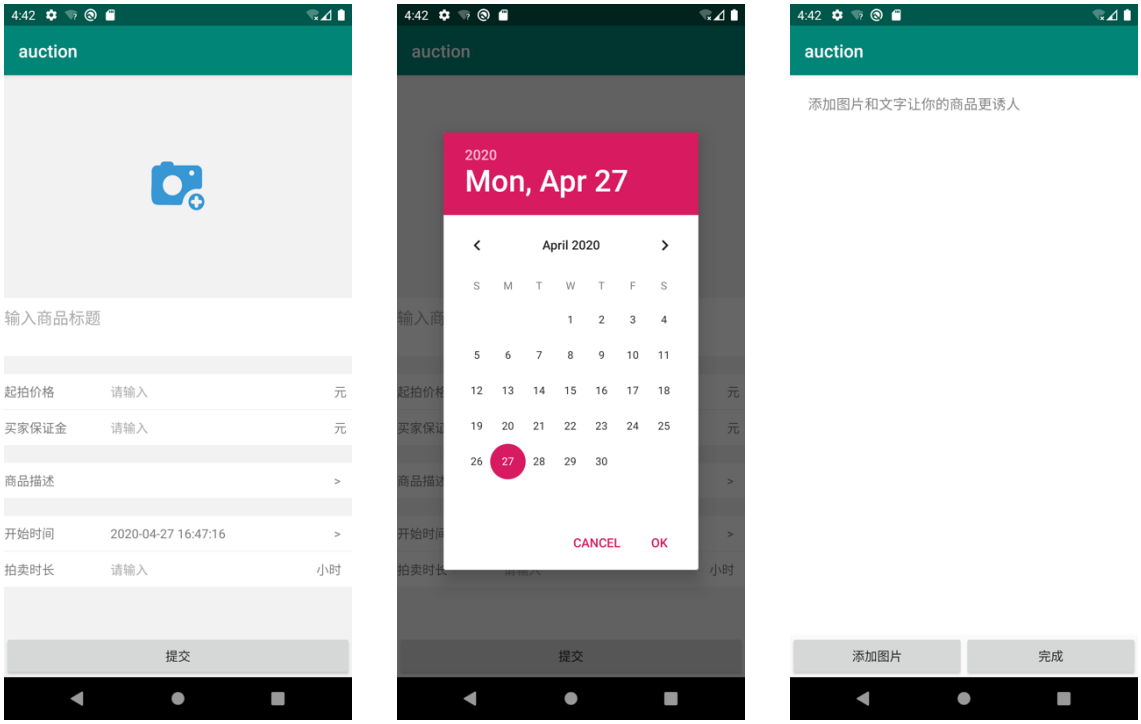
管理员使用管理员账户登录后，进入管理员界面，此界面可以显示拍卖中与待拍卖的商品数目，以及可以进入创建拍品、用户管理和财务管理等功能界面。如图 5-9 所示。

图 5-9 管理员界面示例图

5.7 创建拍品功能的实现

此功能是此系统的重要功能，管理员通过此功能创建拍卖物品并发布，主要有上传图片，编辑标题，设置起拍价、保证金，商品描述，开始时间，拍卖时长等如图 5-10-a 所示。开始时间设置功能使用 Dialog 实现，且设置的时间只能为 5 分钟后，如图 5-10-b 所示。商品描述是使用富文本实现，利用第三方开源库 QRichText 开发，基本实现思路为使用创建拍品活动界面生成的商品 id 传输到富文本编辑界面，富文本界面完成相关文本编辑，利用此 id 在后台生成文件夹，上传富文本 html 文件以及相关资源，完毕后返回结果码销毁此界面回到创建拍品界面，编辑界面如

图 5-10-c 所示。

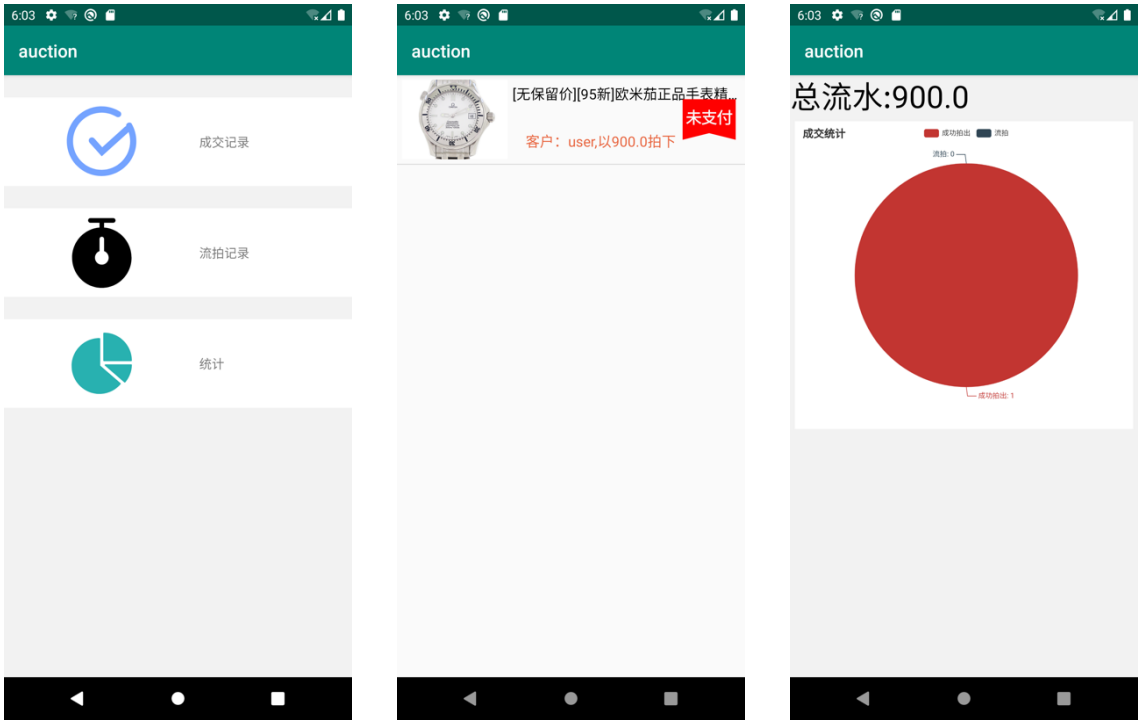


(a) (b) (c)

图 5-10 创建拍品界面示例图

5.8 财务管理功能的实现

此功能由 3 个子功能实现，分别为成交记录、流拍记录和统计如图 5-11-a 所示。其中成交记录中为成功拍出的物品包含用户已支付订单和未支付订单，流拍记录中为流拍物品的列表，统计中包括总流水数以及一个统计饼图。成交记录和流拍记录都是利用 listview 展示向后台请求响应相应的数据，成交记录如图 5-11-b 所示。统计中包括一个 webview 和一个 textview，textview 展示的是后台统计的总流水数，webview 展示的是一个 html 界面，由后台利用 pyecharts（Python 的一个数据可视化第三方库）生成，如图 5-11-c 所示。



(a) (b) (c)

图 5-11 财务管理示例图

5.9 拍卖监控的实现

因为本系统中对拍卖情况的监控是一个循环，需要另外的运行，即后台服务器需要 flask 服务与此循环同时运行。这里采用 threading 将两个服务分为两个线程，同时独立运行。主要代码如下：

```
import threading

import tools

import os

#falsk 启动

def job1():

    os.system('python app.py')

#监控拍卖启动

def job2():
```

```
    tools.UserOrder()
#线程 1
thread1 = threading.Thread(target=job1, name='job1')
#线程 1 启动
thread1.start()
#线程 2
thread2 = threading.Thread(target=job2, name='job2')
#线程 2 启动
thread2.start()
```

后台服务器的监控主要实现原理是对拍卖物品的拍卖时间区间与当前时间进行对比。当当前的时间不在这个区间内时会读取此物品的 ID，首先在拍卖成功表以及流拍表中查询，当未查询到时，查询拍卖记录表，如果未查到将此物品的商品信息插入流拍表，如果查询到，将排名第一的竞拍者信息与此商品的信息插入用户订单表。

安卓客户端的监控主要是在拍卖信息界面中，主要分为两个监控，即待拍卖时的和正在拍卖时的。实现原理主要就是开一个线程对此拍卖物品的开拍时间或终拍时间与当前时间的时间差进行查询，每秒查询一次，并且附带对当前价以及参拍记录的查询，当当前价发生变化时，进行下一步的界面更新。如果当前时间与开始时间的时间差小于等于零时，界面进入拍卖状态。当当前时间与终拍时间的时间差小于等于零，界面进入结束状态，并且根据用户信息与最高价竞拍者信息进行对比，显示成功与识别。如果用户在参拍记录里，点击某个拍卖物品，如果当前时间大于终拍时间，进入结束状态，显示竞拍成功与否，如果当前时间大于开始时间小于终拍时间，进入拍卖状态，用户可以继续竞拍。