中原工学院信息商务学院

中工信商		

本科毕业论文（设计）

基于人证核验的共享酒店住宿信息管理系统

——用户端 APP

系　（部）	信息技术系
专　　业	网络工程
学　　号	201401024234
学生姓名	李金刚
指导教师	吴　颖
提交日期	2018 年 5 月 26 日

摘要

随着互联网的高速发展，与其它产业的结合越来越密切。酒店预订 APP 就是互联网和酒店预订快速发展、相互融合的产物。随着互联网和移动 APP 的发展，越来越多的人喜欢在 APP 上预订酒店。人们急切需求一款可以足不出户、随时随地通过自己的手机来浏览并预订快捷酒店的住宿信息管理系统。然而在当前市场上的一些应用及酒店入住流程太复杂，用户体验和隐私效果较差，为此开发一款简单、实用的基于人证核验的共享酒店住宿信息管理系统具有实际意义。

本文设计并实现了一款基于人证核验的共享酒店住宿信息管理系统，该系统共分为三部分，用户端 APP、酒店端 APP 和后台系统。本文主要介绍用户端 APP，其基于 Android 平台，使用 Java 语言编写并采用 MVP 结构设计。本文分别介绍了相关技术、需求分析、总体设计，系统设计、系统实现和系统测试等。通过系统的需求分析，以及用户的特征分析，把系统的需求分为六大模块。其中包括登录模块、注册模块、人证核验模块、酒店模块、订单模块和用户信息模块。登录模块和注册模块为用户提供注册和登录功能。人证核验是对比身份证照片和实时人脸。酒店模块主要实现了用户搜索酒店，对酒店进行排序和查看酒店、房间详情等。订单模块提供了预订功能、查看订单功能和在线退房等。用户模块主要提供查看用户信息、获取帮助和退出登录等操作。

关键词：Android；酒店预订；人脸识别；身份证识别；共享酒店

Abstracts

With the rapid development of the Internet, the integration with other industries is becoming more and more close. Hotel Reservation app is the product of the rapid development and integration of Internet and hotel bookings. With the development of Internet and mobile app, more and more people like to book hotels on app. People are eager to be able to stay at home, anytime, anywhere through their own mobile phone to browse and book a quick hotel accommodation information management system. However, in the current market of some applications and hotel check-in process is too complex, user experience and privacy effect is poor, to develop a simple, practical based on witness verification of shared hotel accommodation information management system has practical significance.

This paper designs and implements a shared hotel accommodation information management system based on witness verification, which is divided into three parts, the client app, the hotel app and the backend system. This article mainly introduces the client app, which is based on the Android platform, and uses the Java language to write and use the MVP structure design. In this paper, the related technology, requirements analysis, overall design, system design, system implementation and system testing are introduced separately. By analyzing the requirement of the system and analyzing the characteristics of the users, the requirements of the system are divided into six modules. These include login module, registration module, witness verification module, Hotel module, order module and user Information module. The login module and registration module provide users with registration and login functions. Witness verification is a comparison of ID photos and real-time human faces. Hotel module mainly realizes the user searches the hotel, the hotel carries on the sorting and the View hotel, the room details and so on. The order module provides booking capabilities, viewing order features and online check-out. User module mainly provides the view user information, obtains the help and exits the login and so on the operation.

Keywords:Android; hotel reservation; face recognition; identity card recognition; shared hotels

目录

1 引言

1.1 研究背景及意义

在这个移动互联的时代，人们急切需求一款可以足不出户、随时随地通过自己的手机来浏览并预订快捷酒店的住宿信息管理系统。然而在当前市场上的一些应用及酒店入住流程太复杂，用户体验和隐私效果较差，为此开发一款简单、实用的基于人证核验的共享酒店住宿信息管理系统具有实际意义。

该系统是基于共享经济和人脸识别，利用共享经济的优势，让用户可以快速，方便入住酒店。然后结合人工智能的人脸识别算法，验证人脸和身份证信息来保证安全，可靠。最终实现方便用户的人证核验的共享酒店住宿信息管理系统。

1.2 研究状况及发展趋势

共享经济本质是整合线下的闲散物品、人力、资源等，是一种在互联网行业新型起来的新的经济模式。现在已经存在的共享经济有共享单车、共享汽车、共享电动车等，并不断涌入更多新的共享经济形式。以共享充电宝为例，在短短的40 天就能获得了近 12 亿元的融资金额，有将近 35 家的金融机构介入。因此共享经济作为未来经济发展的新模式有着很大的发展空间。

基于共享经济的启发，现提出共享酒店的新课题，面向的用户既包括传统的酒店行业，也包括那些出门在外，长期不在家但是又不想把房子长期租出去的人。传统的酒店行业使用我们的模式时，可以完全不用酒店前台，直接通过系统就可以提供用户入住，退房等过程。对于那些长期不在家的人，可以选择把房子短期出租出去，系统配套的会有保洁人员打扫。这样可以更好的达成共享酒店的模式。

人脸识别是检测和辨认人的面部特征，运用摄像头先采集视频流再检测。最早开始于 20 世纪 60 年代，随着技术的发展直到 90 年代才开始应用。到目前为止，已经有非常成熟的人脸识别技术了。人脸识别技术是否可靠有效，最关键是在与其核心算法。而核心算法的好坏直接影响到识别率和识别技术。该系统采用的是虹软的人脸识别技术，虹软是视觉人工智能技术应用的领军企业。虹软提供的有多个 SDK，其中包括人脸检测、人脸追踪和人脸对比。在该系统中主要进行人证核验，所以首先要对身份证中的人脸和摄像头的人脸进行检测，然后把检测到结果进行对比。对比成功就保存人脸信息到服务器，方便以后进行对比。

将人脸识别和共享酒店结合起来的状况并不多见，所以这是一个新的课题。共享酒店和传统酒店对比，是一个新的开始，它既保留了传统酒店的模式，又利

用互联网和人工智能的优势，不断开创新大陆。将共享经济和人脸识别结合起来，将会擦出新时代不一样的火花。随着时代的进步，市场竞争推动，各行各业的竞争越来越激烈，只有酒店赶上新形势，只有更新改造、升级换代，才能不落伍。所以将人脸识别和共享经济结合起来的基于人证核验的共享酒店住宿信息管理系统具有很的发展前景和优势。

2 相关技术介绍

基于人证核验的共享酒店住宿信息管理系统用户端APP的开发利用Android Studio为开发工具，使用Java语言开发和实现。APP采用开源框架MVPArms，使用Dagger2+Rxjava+Retrofit搭建。采用人脸识别和身份证识别技术进行人证核验。

2.1 Android Studio简介

Android Studio是基于Intellij IDEA，然后Google公司对其进行了改造，成为专门开发Android的开发工具，首先在2013年的Google I/O交流大会推出。相比之前Eclipse更加智能方便，现在很多人都在用，并且Google也停止了对Eclipse插件的更新Google对Android Studio更新速度也很快，所以Android Studio将成为Android开发的主要工具。除此之外，它还具有速度更快、内存占用很平稳、编译更快的优点。UI漂亮扁平化的设计风格、多色多样的主题更美观。自动保存，每次修改都会自动保存，再也不会忘了保存，而狂按Ctrl+S了。整合了新的构建工具Gradle，它是集配置、编译、打包为一体的构建工具，集Ant和Maven的优点于一身。还具有强大的UI编辑器，能够实时预览多设备，对于Android开发者来说就是一个神器。

2.2 MVPArms框架简介

MVPArms是github社区JessYanCoding提供的MVP开源框架，地址为https://github.com/JessYanCoding/MVPArms。在该框架中集成了Android的大部分主流框架，全部使用Dagger2管理。可以快速搭建Dagger2+Rxjava+Retrofit的Android应用。作者还提供了详细的Wiki中文和英文文档，还有一个简单的Demo。上面有各种各样的用法，作者也提供了详细的注释，非常方便使用。除此之外，框架还有很高的扩展性和自定义性，可以满足开发者各色各样的需求，能够适应各种场景。

2.3 人证核验

该系统通过文字识别功能对身份证中的姓名和身份证号进行了识别，使用虹软的人脸识别SDK完成了人证核验。虹软是视觉人工智能技术应用的领军企业。虹软提供的有多个SDK，其中包括人脸检测、人脸追踪和人脸对比等。还免费提供给开发者使用。

3 需求分析

3.1 系统可行性分析

3.1.1 技术可行性

基于人证核验的共享酒店住宿信息管理系统——用户端 APP 的开发利用 Android Studio 为工具，使用 Java 语言设计和实现，采用 MVP 模式。之前经常做一些基于 MVC 设计模式的项目，MVP 又是从 MVC 演化而来。所以对 MVP 了解也很快，并且 Android 完全是 Java 开发，本人对 Java 非常熟悉，再加上使用了 Android Studio 和 MVPArms 框架，简化了开发，所以技术上是完全可行的。

3.1.2 实用性分析

开发该系统的初衷就是为了简化酒店预订和入住的流程，整个流程都可以通过该系统完成，不再需要酒店前台，减少了管理酒店的成本，并且该系统配套有酒店管理端的 APP，可以随时随地查看和管理酒店状态。用户也可以在用户 APP 完成从预订到入住再到离店的全部流程，配合系统的人证核验，不再需要去登记入住，完全做到了一键入住。

用户端 APP 本着实用简洁的原则，界面美观大方，操作也非简单方便，只需几个简单的步骤就能找到满意的酒店。

3.1.3 安全性分析

用户注册必须提供手机号，并且通过验证码验证手机号的正确性。用户密码等信息全部通过加密处理，对用户信息绝对保密，管理员也无法看到。进行预订时用户必须进行注册和通过实名认证。后台通过 Spring Security 管理分为不同的权限，每个用户的权限不同，能够操作的东西也不同。

3.2 业务需求分析

本文设计并实现了一款基于人证核验的共享酒店住宿信息管理系统，该系统共分为 3 部分，用户端 APP、酒店端 APP 和后台系统。本文主要介绍用户端 APP，其基于 Android 平台，使用 Java 语言编写并采用 MVP 结构设计。通过系统的需求分析，以及用户的特征分析，把系统的需求分为六大模块：

注册模块：用户需要提供手机号，验证手机号的正确性，然后提供用户名和密码，对密码复杂性进行校验，最后上传到服务器。

登录模块：用户通过手机号和密码或者手机号和验证码进行登录。

酒店模块：首先获取用户的位置，用户可以选择直接搜索当前位置附近的酒店。用户还可以通过关键字、入住日期、目的地进行酒店搜索。然后可以通过评分，价格，距离等进行排序。可以查看该酒店的详细信息和可预订房间号等。

订单模块：用户选择对应的房间号进行下单，生成订单。然后可以获取全部订单、待入住订单、代付款订单、待评价订单等。还可以进行取消订单，在线退房等。

人证核验：通过用户 APP 与后台进行通信，用手机对用户本人的身份证进行拍照，然后对身份证进行文字识别，获得用户身份信息，然后通过人脸识别对比身份证和本人人脸信息进行人证核验，最后保存到服务器。

我的模块：可获取个人信息、客服中心、退出账户等。

4 总体设计

4.1 业务设计

通过对业务需求的详细分析，把系统的需求分为六大模块。其中包括登录模块、注册模块、人证核验模块、酒店模块、订单模块和用户信息模块。登录模块和注册模块为用户提供注册和登录功能。人证核验就是对比身份证照片和实时人脸。酒店模块主要实现了用户搜索酒店，对酒店进行排序和查看酒店、房间详情等。订单模块提供了预订功能、查看订单功能和在线退房等。用户模块主要提供查看用户信息、获取帮助和退出登录等操作。模块分析如下图 4-1 所示。

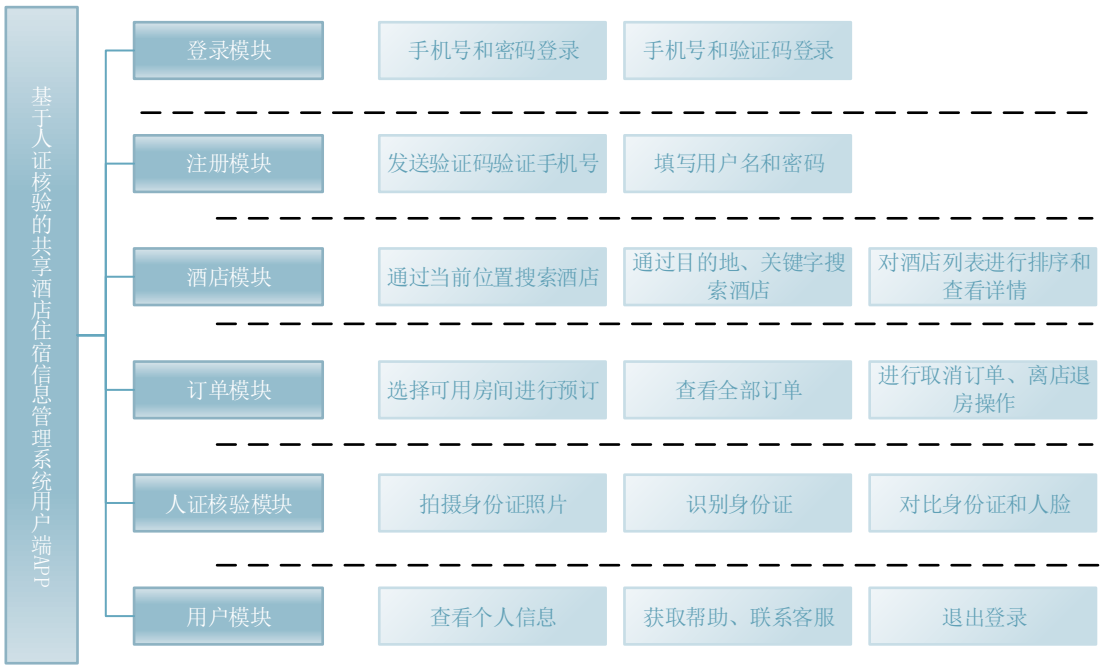

图 4-1 系统业务模块图

4.1.1 登录模块

登录模块是一个系统的基本模块，在该系统中，用户只有登录之后才能预订酒店。没有登录的用户只能浏览酒店信息和房间类型信息。用户一共有两种登录方式，一种通过手机号和密码进行登录，另一种是通过手机号和验证码登录。登录之后会保存登录状态，用户可以进行预订酒店和查看个人信息。对应流程图如下图 4-2 所示。

4.1.2 注册模块

注册模块需要用户首先需要填写手机号，对输入的手机号进行验证，验证手机号的正确性，并且一个手机号只能注册一次。该模块流程如下图 4-2 所示。

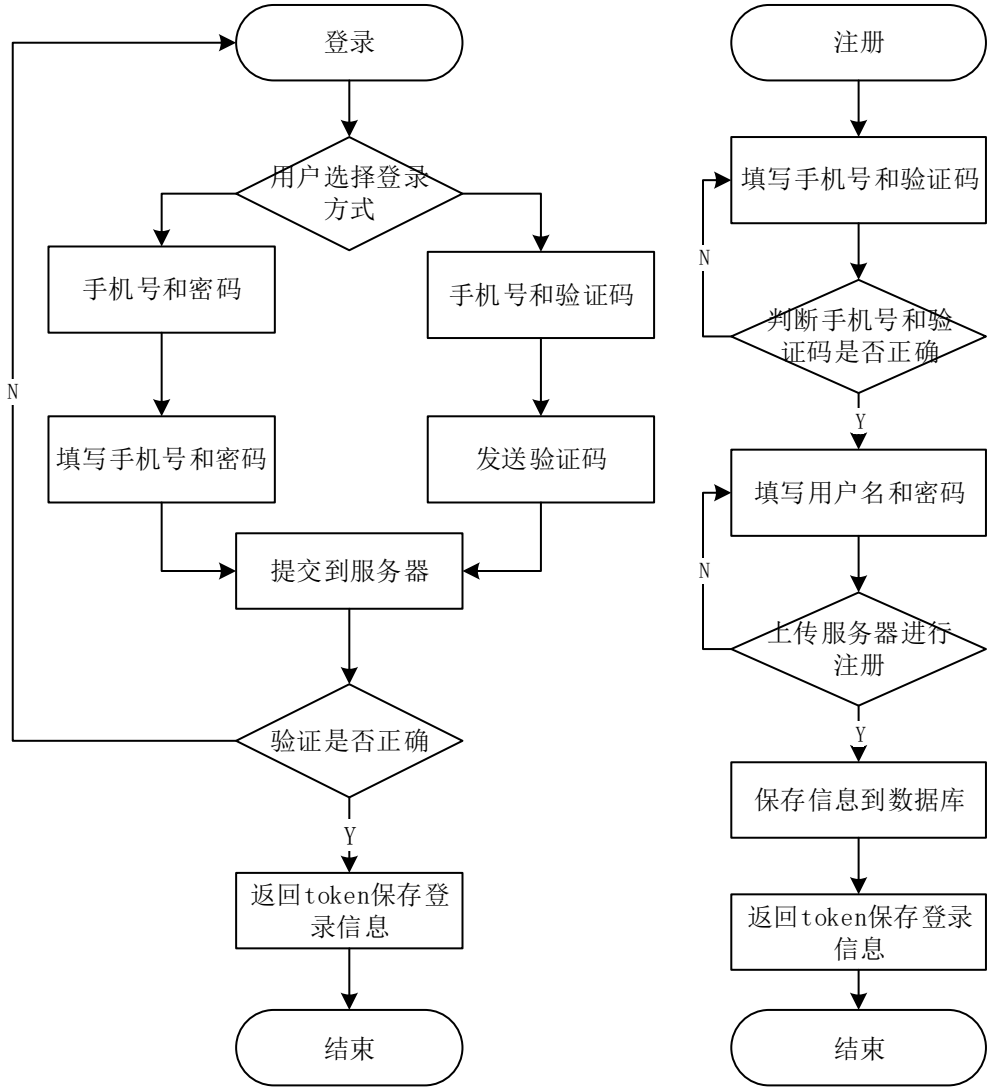

图 4-2 登录和注册流程图

4.1.3 人证核验模块

人证核验一共分为两部分，第一步就是对身份证的进行识别，首先进入到拍摄身份证界面，接着要调用系统相机进行拍照，拍照以后把身份证返回，然后对身份证进行识别。首先是对身份证中的文字进行识别，识别其中的姓名和身份证号，识别之后对其进行校验，成功就保存起来。接着对身份证中的人脸进行识别，把识别到的人脸保存起来。然后进入到人脸识别界面，打开手机的前摄像头，进行实时人脸识别，识别到人脸之后就会进行和身份证中的照片进行对比，如果对比成功之后就会保存当前人脸信息。用户选择下一步就会把这些东西发送到服务器。如果用户不点击下一步按钮，就不会保存实名认证数据。该模块流程如下图 4-3 所示。

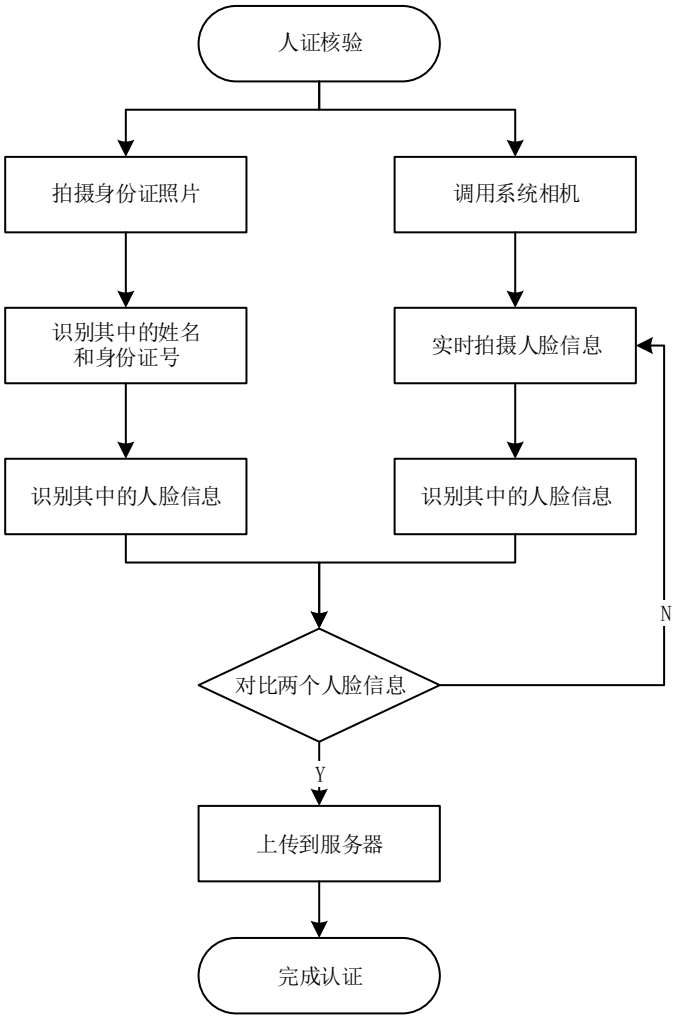

图 4-3 人证核验流程图

4.1.4 酒店模块

酒店模块是该系统的最主要模块，该模块又分为搜索酒店、获取酒店列表、获取酒店详细信息等。

酒店搜索：

酒店搜索一共分为两种，第一种就是获取经纬度，第二种就是通过目的地进行搜索。第一种就是通过经纬度来来获取用户附近的酒店，首先要请求位置权限，然后使用定位功能获取用户的地理位置和经纬度，查询高德地图的地标库，获取现在地理位置的描述，显示到界面。当用户搜索的时候就把经纬度传递给后台，后台会通过经纬度来获取附近的酒店，然后发送到客户端。第二种就是通过目的来搜索酒店，用户需要在目的地这一栏输入目的地，然后下面有一个入住时间，默认的是今天，离店时间是明天，用户可以进行修改。最后还有一个关键字，用户也可以输入关键字来进行搜索指定酒店。这一部分的流程如下 4-4 所示。

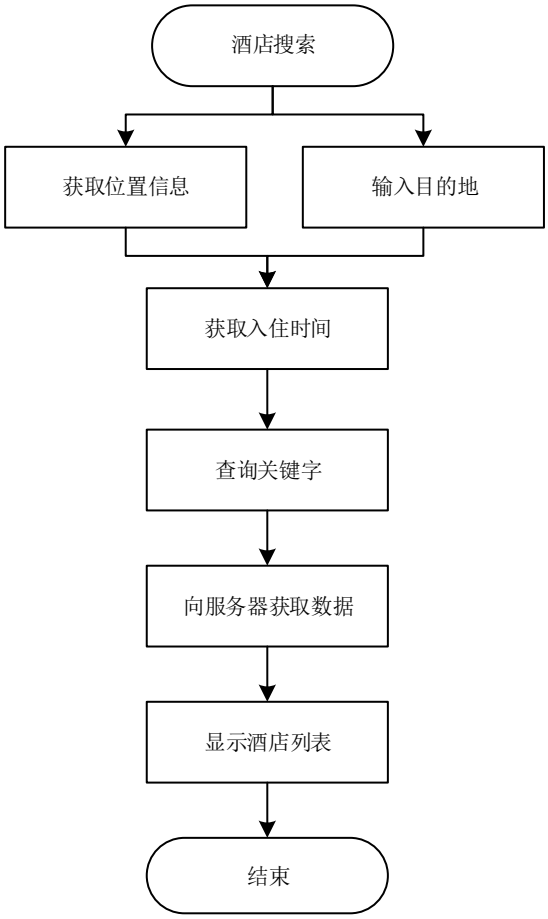

图 4-4 酒店搜索流程图

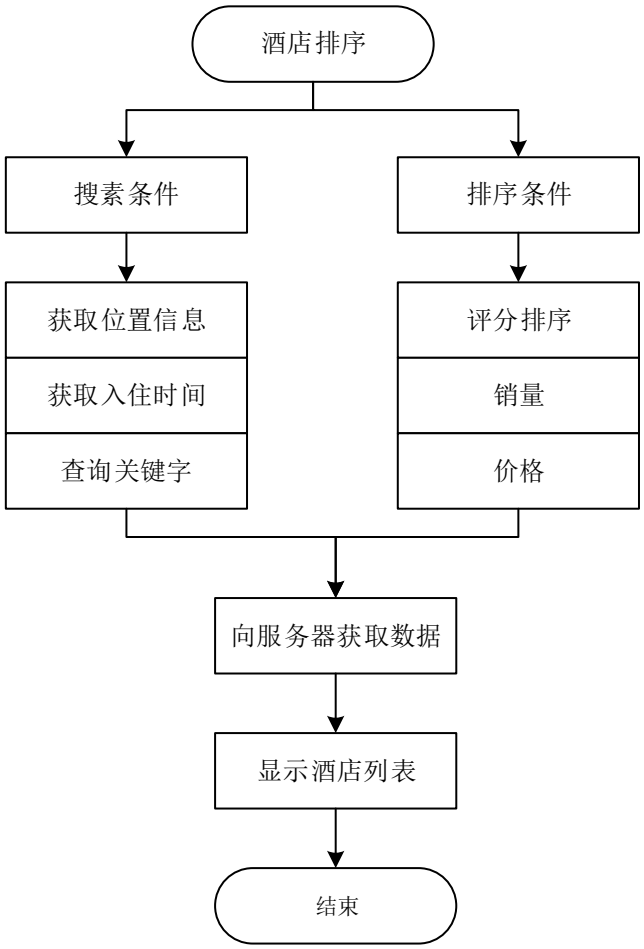

图 4-5 酒店列表排序流程图

酒店列表：

酒店列表界面主要是显示用户搜索到的酒店列表，并且可以在该列表进行排序操作。提供综合排序、距离优先、好评优先、销量优先、价格从高到低和价格从低到高几种排序方式。用户每次选择后，都会重新设置条件，然后重新请求服务器，再把获取到的数据显示到界面。该模块的流程如图 4-5 所示。

酒店详情：

酒店详情一共分为三个部分，第一部分就是获取酒店的详细信息，主要包括酒店的图片、评分、详细地址等。第二部分就是获取该酒店的所有房型，会把返回的数据通过列表的形式展示到界面，用户可以选择房型进行预订。第三部分是点击房型会显示该房型的详细信息，点击其中的预订按钮就可以进行预订。

4.1.5 订单模块

订单模块主要分为生成订单和查询订单，生成订单是用户选择对应的房间后，服务器会生成订单信息，发送到客户端进行确认，如果用户确认订单后，会生成订单，并进入付款界面，用户选择付款后，预订成功。查询订单是用户查询自己

的订单，可以查看全部订单、待入住订单、代付款订单和待退房订单。订单模块流程如图 4-6 所示。

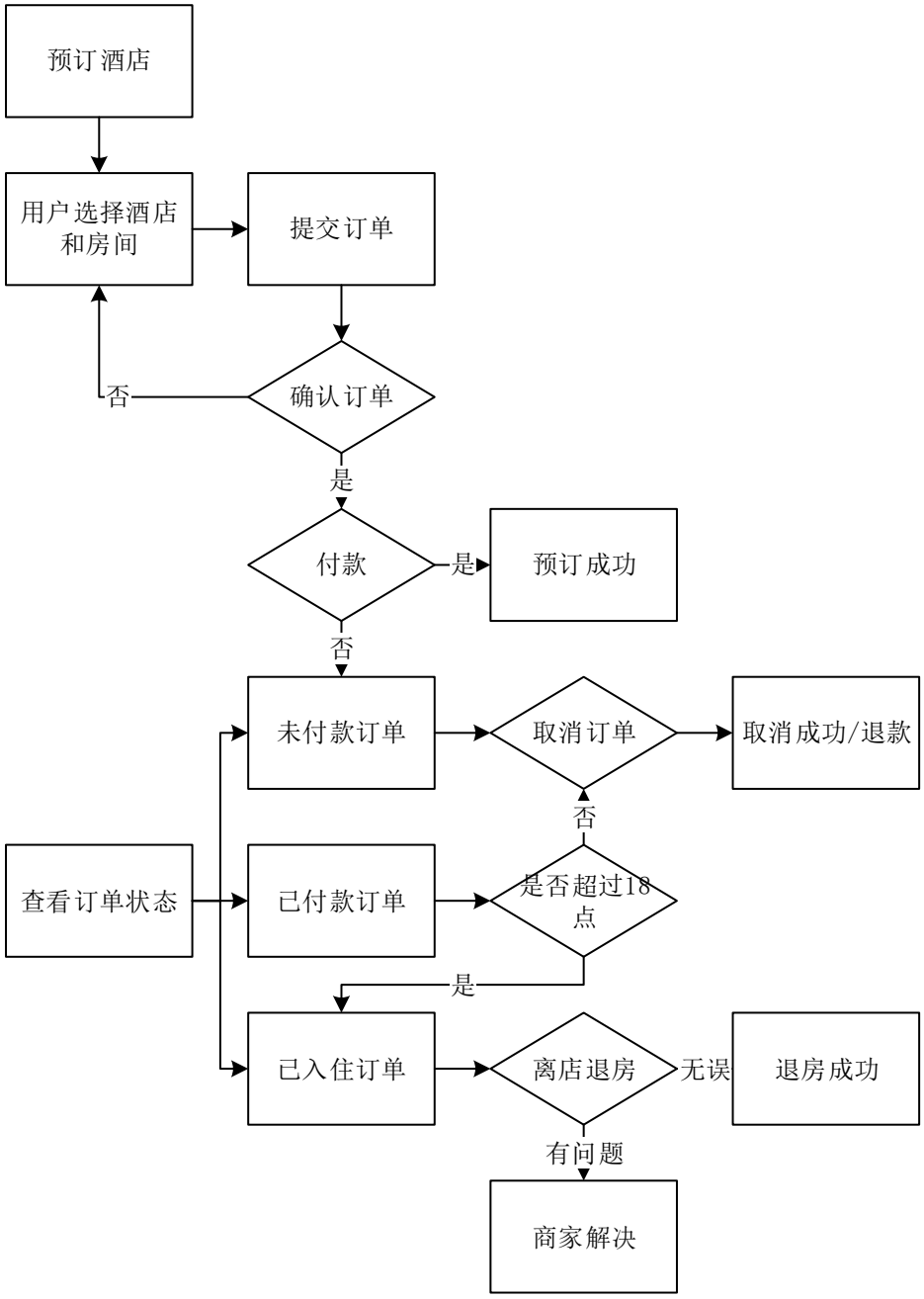

图 4-6 订单模块流程图

4.1.6 用户信息模块

用户信息模块是用来查看用户的个人的信息，主要包括用户名、手机号和是否实名认证等信息。如果没有实名认证可以进行实名认证。

4.2 数据库设计

数据库是程序必不可少的一部分，数据库设计的好坏直接影响到系统后续的开发，所要首先设计好数据库。在该系统中，数据库的设计不仅是数据表，还要用于以后的实体类，更要用于 APP 和服务器通信的 JSON，所以下面的表也代表了通信的 JSON 字段。

通过功能模块分析可知，用户端 APP 大致分为用户、酒店、房间、订单等表。

（1）用户表：用户通过注册后，系统将用户的电话号码、登录密码、权限、用户头像存入到数据库，如表 4.1 所示：

表 4.1 user 表

字段	说明	类型	字段大小	不为空
uid	用户编号（主键）	int	11	是
phoneNumber	手机号	varchar	11	是
username	用户名	varchar	100	是
password	密码	varchar	100	是
roles	用户角色权限	varchar	100	是
avatar	用户头像	varchar	100	是

（2）实名认证表：用户注册后需要通过身份证和人脸对比进行实名认证，实名认证过的用户系统将身份证上的信息读取出来存入到数据库，其中 uid 对应的是 user 表中的用户，如表 4.2 所示：

表 4.2 verifyuser 表

字段	说明	类型	字段大小	不为空
verifyId	实名编号（主键）	int	11	是
uid	用户编号	int	11	是
idNumber	身份证号	varchar	50	是
realName	真实姓名	varchar	100	是
postive	身份证证面	text	0	是
opposite	身份证反面	text	0	是
face	人脸图像	blob	0	是

（3）酒店表：主要储存酒店详细信息，拥护搜索酒店、预定酒店都需要 hotel 表，管理员管理酒店也需要用到，如表 4.3 表所示：

表 4.3 hotel 表

字段	说明	类型	字段大小	不为空
hotelId	酒店编号（主键）	varchar	100	是
hotelName	酒店名称	varchar	20	是
imageUrl	酒店图片	varchar	100	否
addressDes	酒店地址	varchar	200	是
longitude	酒店所在经度	double	10	是
latitude	酒店所在纬度	double	10	是
score	酒店评分	double	2	是
sales	酒店总销量	int	11	是
minPrice	最低消费	decimal	10	否

（4）房间表：记录的是所有酒店的房间，需要有房间编号、房间号、房间类型编号关联的房间类型实体、房间状态（1.使用中，2.未使用，3.待打扫，4.可交付，5.待维修）、入住时间和离店时间，如表 4.4 表所示：

表 4.4 room 表

字段	说明	类型	字段大小	不为空
roomId	房间编号（主键）	int	11	是
roomNum	房间号	varchar	100	是
typeId	房间类型编号	int	11	是
status	房间状态	int	1	是
startTime	入住时间	datetime	0	否
endtime	离店时间	datetime	0	否

（5）房间类型表：记录的是所有房间的类型，room 表中的 typeId 对应的是每个房间的房间类型，hotelId 对应的是此房间类型所属的酒店，如表 4.5 表所示：

表 4.5 roomtype 表

字段	说明	类型	字段大小	不为空

typeId	房间类型编号（主键）	int	11	是
hotelId	酒店编号	varchar	100	是
typeName	房间类型名称	varchar	100	是
typeDes	房间类型描述	varchar	100	是
typePrice	房间单价	decimal	7	是
cancal	是否可取消	tinyint	1	是

（6）房间类型详情表：记录房间的详细情况，所属哪一个房间详情，如表 4.6 表所示：

表 4.6 roomdetail 表

字段	说明	类型	字段大小	不为空
detail	房间类型详情编号（主键）	int	11	是
typeId	房间类型编号	int	11	是
network	上网方式	varchar	100	是
bathroom	卫生间类型	varchar	50	是
window	是否有窗	varchar	50	是
capacity	入住人数	int	11	是
bedDes	床规格	varchar	50	是
typeDes	发票类型	varchar	50	是
typePhoto	房间类型图片	varchar	500	是
liveDes	入住说明	varchar	400	是

（7）用户订单表：用户下单后，记录存储用户入住的哪一个酒店的哪一个房间，房间信息和入住离店时间，以及订单不同时间的不同订单状态（0.待付款，1.已支付，待入住，2.已入住，3.退房待确认，4.待评价，5 已完成，6.订单取消中，7.已取消,8.待处理），如表 4.7 表所示：

表 4.7 userorder 表

字段	说明	类型	字段大小	不为空
orderId	订单编号（主键）	varchar	100	是
uid	用户编号	bigint	20	是

hotelId	酒店编号	varchar	100	是
roomId	房间编号	int	11	是
hotelName	酒店名称	varchar	100	是
roomNum	房间号码	varchar	100	是
typeName	房间类型名称	varchar	100	是
typePhoto	房间类型图片	varchar	100	是
startTime	入住时间	datetime	0	是
endTime	离店时间	datetime	0	是
price	订单总价	decimal	7	是
status	订单状态	int	11	是

（8）订单详情表：确认订单的用户将记录用户信息到订单详情表，并关联订单表，如表 4.8 表所示：

表 4.8 orderdetail 表

字段	说明	类型	字段大小	不为空
orderdetailId	订单详情编号（主键）	int	11	是
orderId	订单编号	varchar	100	是
addressDes	酒店地址	varchar	100	是
realName	用户真实姓名	varchar	100	是
phoneNumber	用户手机号	varchar	11	是
idNumber	用户身份证号	varchar	20	是
orderTime	下单时间	datetime	0	是
payMoney	支付金额	decimal	7	是
typeDes	发票类型	varchar	50	是
liveDes	入住说明	varchar	400	是

4.3 界面设计

界面作为一个系统的门户，是直接和用户体验息息相关的，所以一个 APP 的最重要的就是界面设计，也就是 UI 设计。系统本着简洁大方、实用的原则，大致分为主界面、酒店列表、酒店详情界面、登录注册界面和实名认证界面。

4.3.1 主界面

主界面是用户看到的第一个界面，也是用户使用和点击最多的界面。所以主界面采用底部导航的方式展现给用户，主要分为三个模块，分别是首页、订单和我的。考虑到该系统主要是进行预订酒店，所以查找酒店的操作放在了首页。可以方便用户查找和预订酒店。首页主要包括了五个控件，目的地和关键字可以输入作为搜索条件。点击我的位置可以获取当前位置，点击入住时间会弹出日历控件来选择入住时间。点击搜索按钮则会根据当前条件进行酒店搜索。订单界面主要用来体现用户的订单信息，能够让用户更方便的查看订单信息。在此界面又采用了顶部导航，分为 4 个部分，分别是全部订单、待入住订单、代付款订单和待退房订单。用户可以点击或者滑动来查看对应的订单信息。我的界面主要用来显示用户的个人信息和进行其它操作。主界面如图 4-7 所示。

图 4-7 主界面

图 4-8 酒店列表界面

4.3.2 酒店列表

酒店列表界面主要有显示搜索到的酒店和对结果进行排序的功能。所以该界

面主要有两部分组成，上面是搜索框和下拉选择框，下面是一个显示列表的视图 RecyclerView，每当上面的条件改变的时候，就要更新下面的 RecyclerView。可以综合排序、距离优先、好评优先、进行销量优先、价格从高到低和价格从低到高等。效果如图 4-8 所示。

4.3.3 酒店详情界面

酒店详情界面主要展示三部分内容，酒店的详细信息、酒店的房型信息列表和可用房间列表。该界面上部分是该酒店的详细介绍，在该界面做了一个滑动特效，就是首先实现酒店图片沉浸到状态栏。然后用户向上滑动改变状态颜色和显示 ActionBar 和酒店名字。该特效是先自定义一个继承 ScrollView 的布局，再监听其中的滑动事件，进而改变状态栏和 ActionBar 的状态和颜色。下面的部分是显示该酒店的所有的房间类型，点击该房间类型列表会弹出一个 PopWindow 显示该房间类型的详细信息。如果用户点击预订按钮就会显示该房间类型可用的房间号。效果如图 4-9、4-10 所示。

图 4-9 酒店详细信息

图 4-10 房间详细信息

4.3.4 登录注册界面

登录和注册是系统的基本功能之一，所以登录注册界面必不可少，登录和注册界面采用统一的风格，一共包括四个界面，手机号密码登录界面、验证码登录

界面、验证手机号和填写用户名密码。其中是两个 Activity，分别是 LoginActivity 和 RegisterActivity，然后每个 Activity 包括两个 Fragment，登录界面分别是手机号密码登录界面和验证码登录界面，可以在其中一个界面切换到另一种登录方式，页面布局大致一样。注册界面首先是验证手机号发送验证码，点击下一步进入到填写用户名和密码界面。用户填写完成后就可以进行注册了。

4.3.5 人证核验界面

人证核验界面也是系统重要的界面之一，该界面分为三个部分，选择身份证界面、拍照界面和人脸拍摄界面。选择身份证界面分两个部分，身份证正面和反面，用户点击可以进入到拍照界面，然后把拍照片回显到当前界面，点击下一步进入到人脸识别界面，在此界面实时显示人脸进行对比，成功的话就显示下一步按钮。效果如下图 4-11 所示。

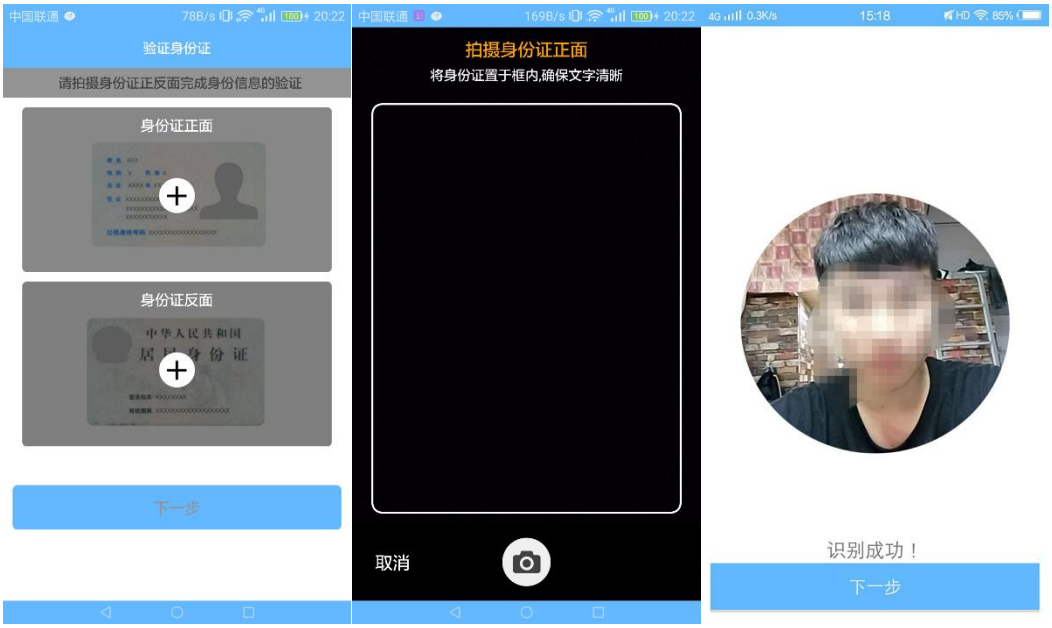

图 4-11 人证核验界面图

5 系统实现

5.1 登录模块

用户登录是一个系统不可或缺的一部分，用户只有进行登录之后才可以下单预订酒店。该系统是使用 token 的方式进行认证，就是当用户登录之后会生成一个 token，以后的验证都是通过 token 进行验证。为了方便以后登录，需要在 APP 端保存 token 信息，保存 token 最后的地方就是 SharePreference 中了。首先创建一个仅能自己 APP 读取的 SharePreference，然后在里面保存 key 为 token，value 为具体的 token 字符。这样就可以在以后的请求中从 SharePreference 中读取 token，并在请求中加入 token。以后检验身份下单等操作就可以使用该 token 进行操作，不仅避免了频繁传输用户名密码带来的安全问题，而且减少了频繁查询数据库造成的压力。但是有一个问题就是，这样需要每次请求都要手动添加 token，这样不仅麻烦，而且代码冗余比较多。为了解决这个问题，可以实现一个全局的 HTTP 网络请求拦截器，可以在改拦截器里面实现读取 SharePreference 中的 token 信息，然后判断 SharePreference 中的 token 是否为空，如果不为空就把 token 添加到 request 的 header 中。这样就可以实现每次请求都带会带上已经存在的 token。如果 token 信息过期或者丢失，拦截器会让 APP 跳转到登录界面，提示用户重新登录。设计说明请参照下面的图 5-1。

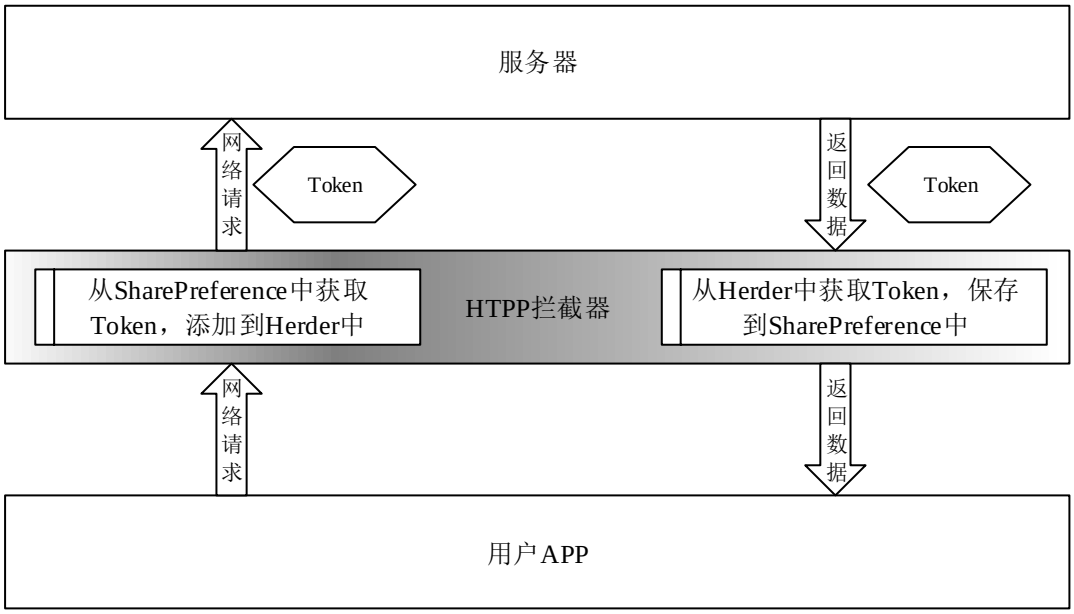

图 5-1 登录模块设计说明图

在该系统中提供两种登录方式，第一种就是通过手机号和密码进行登录，第二种就是通过手机号和短信验证码进行登录。这两种登录方式是在一个 Activity 实现的，默认为手机号和密码登录，点击切换成用户名的验证码登录。下面分别

介绍这两种登录方式。相关界面如下所示。

图 5-2 手机号密码登录

图 5-3 手机号验证码登录

5.1.1 用户名和密码登录

手机号和密码是默认的登录方式，该模块一共有三个控件，手机号输入框、密码输入框和登录按钮。首先限制手机号输入框只能输入数字，并且只能输入 11 位使用 android:inputType="phone" android:maxLength="11"两个属性就可以完成。然后要判断输入密码的复杂度，在这里使用正则表达式进行判断，只能输入 6-15 位数字或字母的密码。然后监听登录按钮的点击事件，首先要判断手机号和密码是否为空，然后是否负责校验规则。如果全部符合的话，就会获取输入的值，然后传递给 P 层，P 层会调用 M 层方法请求服务器，服务器返回一个 Result 对象，其中判断其中的 success 属性是否为 true，如果为 true 就说明登录成功，跳转到主界面。如果为 false，就提示用户 message 属性中的错误信息。其中错误信息有两种，当用户名不存在的时候提示用户名不存在，当密码不正确的时候提示密码不正确，相关代码如下。

```
if (result.getSuccess()){
    mRootView.startMain();
}else{
    mRootView.showMessage(result.getMessage());
```

}

5.1.2 手机号和验证码登录

当用户点击验证码登录的时候，就会切换到手机号和验证码登录方式，该界面一共有四个控件，首先是两个输入框，一个获取验证码按钮和一个登录按钮。首先判断输入的手机号是否正确，然后监听获取验证码按钮，点击该按钮会启动一个计时器，然在 60 秒内不能再次点击。首先要启动计时器，然后进行倒计时，把按钮致为不能点击的状态，每过一秒钟减一，然后通知主线程更新 UI，就更新按钮上显示的秒数。这是会请求服务器给手机号发送一个验证码，发送验证码使用的是百度的短信接口，服务器会把发送的手机号和验证码保存下来。当用户输入验证码之后，会再次请求服务器，服务器会对比手机号和密码是否正确，正确的话就会登录成功，返回 token，然后跳转到主界面，如果错误提示用户错误信息。

5.2 注册模块

注册模块一共分为两步，第一步就是验证用户输入的手机号是否正确，第二步就是输入用户名和密码。完成这两步才算注册成功，保存到数据库。相关界面如下。

图 5-4 验证手机号界面

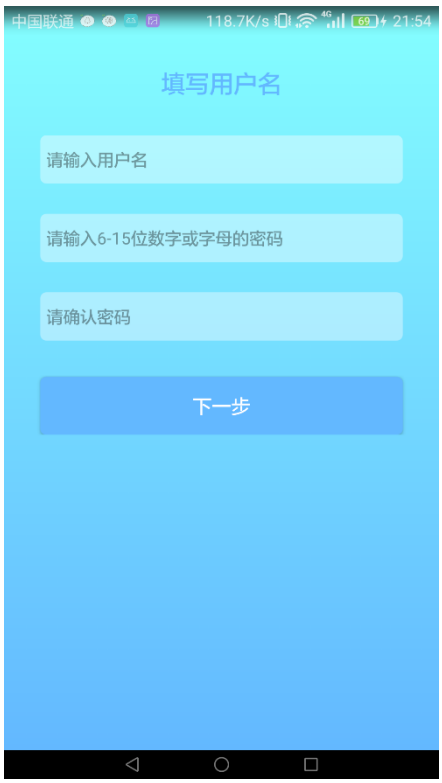

图 5-5 填写用户名密码界面

5.2.1 验证手机号

验证手机号是使用验证码的方式进行验证，获取验证码的方式和验证码登录的方式一样，在这里就不在叙述。比较重要的步骤就是要验证手机号是否已经注册过了。这就需要在发送验证码之前，服务器首先去数据库查询一下用户表，里面有没有该手机号，如果有就代表已经注册过了，提示用户该手机号已经注册过。如果手机号和验证码都填写正确，则进入到下一步，填写用户名和密码界面。由于服务器需要用户填写完用户名和密码才保存到数据库，所以服务器没有保存该手机号，需要 APP 端在该界面传递到下个界面，然后在下个界面提交给服务器。在两个 Fragment 之间传递参数如下：

```
Bundle bundle = new Bundle();
phone = phoneNumber.getText().toString();
bundle.putString("phone", phone);
regNameFragment.setArguments(bundle);
start(regNameFragment);
```

5.2.2 填写用户名和密码

通过手机号的验证之后会进入到填写用户名和密码界面，该界面需要输入用户名、密码和确认密码。用户名只允许输入除特殊字符外的其他字符，不允许输入特殊字符，输入特殊字符就不显示。首先定义一个 InputFilter，这个就是监听 EditText 的输入的过滤器，然后重写其中的 filter 方法，利用正则表达式，如果符合特殊字符的话就返回空字符串，这样在输入框就不会显示。其中主要代码如下：

```
InputFilter filter_speChat = new InputFilter() {
@Override
public CharSequence filter(CharSequence charSequence, int i, int i1, Spanned spanned, int i2, int i3) {
String speChat = "[`~!@#_$%^&*()+=|{}':;',\\[\\].<>/?~！@#￥%……&*（）——+|{}【】‘；：”“’。，、？]";
Pattern pattern = Pattern.compile(speChat);
Matcher matcher = pattern.matcher(charSequence.toString());
if (matcher.find()) return "";
else return null;
    }
};
```

验证完用户名之后，接下来就要验证密码，密码是在点击注册按钮的时候进行验证，首先要验证用户名和密码不能为空，然后要验证密码是否符合复杂度的要求，该部分是使用正则表达式进行验证，最后还要验证两次输入的密码是否一致。如果全部通过，就获取上个界面传递过来的手机号发送给服务器，注册成功就返回 token，默认用户已经登录，注册失败提示用户错误信息。

5.3 人证核验

人证核验是一个非常重要的阶段，在该系统中人证核验分为两步，第一步就是拍摄身份证照片信息，第二步就是对比身份证的信息和人脸信息是否一致。

5.3.1 拍摄身份证

首先进入到验证身份证界面，在该界面需要用户拍摄身份证的正反面的照片。界面一共分为 3 个控件，两个 ImageView 控件，显示的是默认的照片。下面是一个按钮，为不可点击状态，只有用户拍摄了身份证才会变为可点击状态。用户点击预览图片就会进入到自定义相机界面 ShootIdCardActivity，该界面包含了一个 SurfaceView，这个 View 会实时显示相机的内容。还自定义了一个 CustomCameraPreview，该界面实现了 SurfaceHolder.Callback, AutoFocusCallback 这两个接口，SurfaceHolder.Callback 是相机预览的回调接口，当相机创建、改变和销毁的时候会执行对应的方法。Camera.AutoFocusCallback 是自动对焦的接口，实现该接口就可以实现自动对焦。然后还有一个拍照按钮，当用户点击该按钮的时候就调用相机的 takePicture 方法，进行拍照。实现其中的 PictureCallback 接口就可以拿到其中的 data 数据，然后调用保存图片的方法，把图片保存的手机的存储空间中，并返回保存的路径。最后再把路径返回到验证身份证界面。验证身份证界面拿到路径之后，把该照片显示到开始的预览图上。保存图片的代码如下：

```
Camera.PictureCallback jpegCallback = new Camera.PictureCallback() {
        public void onPictureTaken(byte[] data, Camera camera) {
            new SaveImageTask(data).execute();
        }
    };
```

5.3.2 识别身份证

识别身份证分为两部分，第一部分就是对身份证进行文字识别，识别其中的姓名和身份证号。第二部分就是对身份证中的人脸进行识别。

对身份证的文字识别使用的是文字识别技术，就是对一个图片上的文字进行识别。首先创建一个 IDCardParams 对象，把请求的图片路径和其它参数设置好，然后调用 OCR.getInstance().recognizeIDCard 方法进行识别，该方法有两个回调方法，一个是成功的，一个是错误的方法。在成功的方法里会有 IDCardResult 对象，该对象是身份证信息，包括姓名、身份证号、出生日期、家庭住址等信息。然后保存该对象，如果信息识别成功，就可以点击下一步按钮。

第二步就是对该图片的人脸信息进行识别，人脸识别技术主要使用的是虹软(ArcSoft)公司的 SDK，主要使用了两种技术，第一种就是人脸检测，从视频中校测其中的人脸。第二种是人脸识别，就是对比两张人脸的相似度。首先初始化人脸检测引擎 AFD_FSDKEngine，然后使用该人脸检测引擎调用其中的 AFD_FSDK_StillImageFaceDetection 方法，其中需要传递 5 个参数，第一个参数就是要识别的视频流数据，是一个字节数组，第二个和第三个为传入数据的宽和高。第四个参数为视频格式默认为 NV21 格式。第五个为人脸检测的结果，是 AFD_FSDKFace 对象的 List 集合。其中包含了所有的人脸角度和识别出来的人脸位置坐标。只需把该信息保存下来就可以用于下个界面的人脸识别。

图 5-6 验证身份证界面

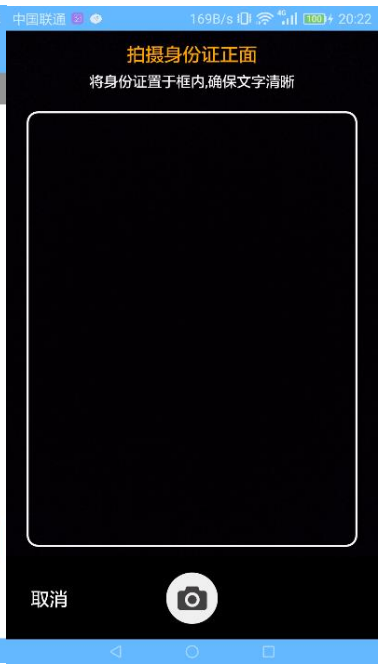

图 5-7 拍摄身份证界面

图 5-8 身份证识别成功界面

5.3.3 人脸对比

通过身份证识别之后，就可以进行人脸对比。进入到人脸对比界面，在该界面首先要实现 Camera.PreviewCallback，在 onPreviewFrame 回调方法中可以获得相机视频的数据流 bytes。然后首先初始化人脸检测引擎，把该视频流传过去，

会得到一个 AFD_FSDKFace 对象，然后初始化人脸识别引擎 AFR_FSDKEngine，调用其 AFR_FSDK_ExtractFRFeature 方法，此方法需要七个参数，第一个参数就是视频流，然后是宽和高。接着是视频流的格式为 NV21，第五个参数为刚才人脸检测的人脸位置坐标，第六个参数为刚才人脸检测的人脸角度。最后一个参数为返回的 AFR_FSDKFace 对象。该对象是一个包含人脸特征信息的对象，对象中包含了一个 mFeatureData 属性，里面是一个字符数组，里面存放的是人脸的特征信息。对应方法如下

engine.AFR_FSDK_ExtractFRFeature(data, width, height, AFR_FSDKEngine.CP_PAF_NV21, resultFace.getRect(), resultFace.getDegree(), face2);

识别到相机中的人脸信息之后，要对上个界面传递过来的身份证上的人脸进行识别，识别方法和上面相似，会得到一个另 AFR_FSDKFace 对象，然后调用其中 AFR_FSDK_FacePairMatching 人脸对比方法。该方法需要 3 个参数，一个前两个参数是要对比的人脸信息，第三个参数是返回的对比分数，对比相关的代码如下：

```
private boolean compare(AFR_FSDKFace face1, AFR_FSDKFace face2) {
float grade;
//score 用于存放人脸对比的相似度值
AFR_FSDKMatching score = new AFR_FSDKMatching();
//调用人脸识别的对比方法
afr_fsdkError = engine.AFR_FSDK_FacePairMatching(face1, face2, score);
grade = score.getScore();
if (grade >= 0.5) {
        return true;
    }
     return false;
}
```

经过多人的测试，在本系统中设置对比分数大于 0.5 为对比成功，对比失败就重复上面的步骤，如果对比成功就会保存该人脸信息。停止相机预览，并显示下一步按钮。当用户点击下一步按钮的话，就调用 P 层的方法把身份证正反面照片、姓名、身份证号和人脸信息发送到服务器，服务器保存成功之后返回客户端成功信息，人证核验完成。当识别成功之后，要在 Activity 的 onDestroy 方法中销毁对应的人脸识别引擎。只有用户点击下一步按钮才会向服务器发送数据，如果用户中途离开则不会保存信息，也对此进行了提示。相关的截图如下所示。

图 5-9 初始界面

图 5-10 提示用户界面

图 5-11 对比成功界面

5.4 酒店模块

酒店模块是该系统中最主要的模块，很多业务都在该模块。该模块又分为五个部分。分别是酒店搜索、酒店列表和排序、酒店详细信息、酒店房型列表和可用房间列表。

5.4.1 酒店搜索

酒店搜索部分又是酒店模块的主要部分，关系用户是否能够找到满意的酒店，是否能找到合适的酒店。所以这一部分非常复杂，当初在设计的时候也耗费了很长时间。由于传输数据多而复杂，所以我们制定了两个实体类，SearchHotel 和 Hotel，其中 SearchHotel 包含了九个属性，如下所示：

private String destination; // 目的地

private Coordinate coordinate; //当前位置信息

private String startTime; // 入住时间

private String endTime; // 截止时间

private String key; // 酒店名或地址关键字

private String sort; // 排序方式

private String sortKey; // 查询方式

private Integer currentPage; // 当前页

private T data; // 查询结果

Hotel 里面是酒店的一些信息，在此就不再列举。进入到首页，会调用高德的定位服务来获取当前的位置信息，然后把市、县和街道信息显示在目的地的 EditTextView 中，把经纬度信息存到 Coordinate 对象中。然后用户可以编辑目的地信息来修改目的地。下面是入住时间，默认时间为今天入住明天离店。用户点击会弹出一个 PopWindow，里面包含一个日历选择控件，该控件是在 github 上面网友分享的一个控件 com.henry.calendarview.DayPickerView，通过修改实现了需要的功能。当用户选择好入住日期的话，就会显示确定按钮，用户点击确定按钮会给选择的日期回显到界面，并关闭 PopWindow。下面就是一个关键字的 EditTextView，用户可以输入需要搜索的关键字。当用户点击查找酒店按钮，就会给这些条件封装到 SearchHotel 中。其它没有的条件系统会给出默认值。封装以后把该 SearchHotel 对象传到下个界面。

图 5-12 获取到位置信息

图 5-13 选择日期界面

5.4.2 酒店列表

用户点击搜索酒店就会跳到酒店列表界面，并传递了一个 SearchHotel 对象，然后通过调用 P 层方法去请求服务器，把 SearchHotel 传服务器，服务器会返回带有结果的 SearchHotel 对象，然后取出其中的 List<Hotel>，创建一个 Adapter 用来解析 List<Hotel>，然后把 Adapter 设置到 RecyclerView 中，就会显示酒店列

表到界面。如果其中酒店列表为空，就在下面显示暂时没有符合要求的酒店。列表页顶部搜索框使用谷歌原生的 SearchView 控件，设置监听器，然后当用户点击的时候，重新设置 SearchHotel 条件，然后重新请求服务器获取新的数据，更新 Adapter 和 RecyclerView。

排序条件使用的是下拉菜单，一共三个下拉菜单，首先分别给这三个下拉菜单设置点击监听事件，当用户点击不同的菜单的时候，触发不同的点击事件，然后更改 SearchHotel 条件重新请求服务器获取新的数据，更新 Adapter 和 RecyclerView。下面为其中一个下拉菜单监听的伪代码：

```
zhpxView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Overwrite
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        zhpxAdapter.setCheckItem(position);//获取点击的是那一项
        mDropDownMenu.setTabText(position == 0 ? headers[0] : zhpx[position]);//改变下拉菜单的选中项
        mDropDownMenu.closeMenu();//关闭下拉菜单
        key.setSortKey(szhpx[position]);//重新设置关键字
        key.setSort("DESC");
        mPresenter.getHotelData(key);//调用 P 层方法请求服务器
    }
});
```

图 5-14 酒店列表

图 5-15 搜素条件

5.4.3 查看酒店详细信息

用户点击酒店列表的任意一个酒店就可以进入到酒店详细界面，首先请求服务器，把该酒店的 HotelId 发送的服务器查询酒店的详细信息。然后进入到酒店详情界面，在酒店详情界面使用了自定义 View 来实现一个滑动特效，首先创建 TranslucentScrollView 类继承 ScrollView，然后监听其中的滑动事件，当用户滑动的时候改变 ActionBar 的透明度，这样就实现了，进去的时候是一个全屏界面，然后向上滑动的时候就会改变 ActionBar 的透明度，并且显示酒店的名称。实现特效以后就是显示酒店的详细信息了，只用把服务器传过来的数据进行显示就可以。滑动前和滑动后的对比如下图 5-16 和 5-17 所示。

图 5-16 滑动前效果图

图 5-17 滑动后效果图

5.4.4 获取酒店房型列表

用户进入详细界面之后，就调用 P 层方法请求服务器。然后查询该酒店的所有的房型信息，然后服务器返回是个 List<RoomType>，里面是房型列表，把该 List 放进 TypeListAdapter 里面，然后填填充到 RecyclerView 中，这样就可以显示房型列表了。

当用户点击房型列表的时候，出现一个弹出框 PopWindow 来显示房型的详细信息。首先让 PopWindow 显示出来，然后去服务获取数据，最后把房型列表的详细信息显示到 PopWindow 中。

5.5 订单模块

订单也是一个非常重要的模块，在该模块主要分为三部分，就是预定酒店，查看订单和修改订单状态也就取消订单和退房离店等操作。

5.5.1 在线预订

用户查找到指定酒店之后，选择对应的房型信息，点击预订按钮，就会弹出一个 PopWindow，然后向服务器发送请求获取该房间类型的可用房间号，已经预定或者正在入住的不会返回。该界面是使用 RecyclerView 的网格模式，就是每行能显示多个 Item，效果如下图所示。如果用户需要预定房间就可是点击对应的房间号，然后向服务器发送请求，参数包括入住时间，预订酒店 ID、预订房间类型和预订的房间号。服务器会判断该用户是否登录或者是否通过人证核验，只有通过人证核验的才能进行预订。通过认证就后会跳转到确认订单界面，让用户确认订单信息是否有误，如果无误就点击确认订单，服务器会保存订单，订单状态为未付款，然后进入付款界面，用户选择支付方式进行支付。由于支付接口的限制，该系统只修改订单状态，不做支付功能。

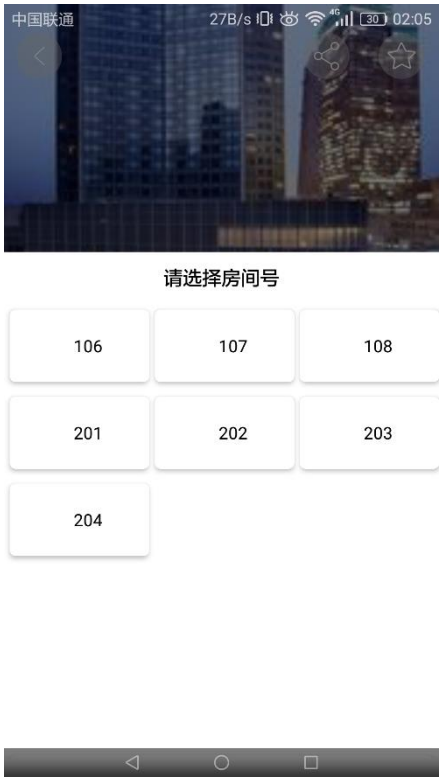

图 5-18 选择房间界面

图 5-19 确认订单界面

5.5.2 查询订单

用户点击订单，就会进入到订单界面，在订单界面是使用一个顶部导航，用

户可以查看全部订单、待入住、待付款和待退房订单。这四部分是使用了四个 Fragment，然后放进一个 ViewPage 里面，然后监听 ViewPage 其中的滑动事件，当用户滑动的时候就会切换 Fragment 来显示不同的界面。每个界面的操作会不一样，待付款的界面可以取消订单，点击取消订单按钮就请求服务器，服务器就会修改订单状态为已取消。在待退房界面用户可以申请退房离店，然后服务器会给订单状态修改为退房待确认，并且提醒保洁人员进行确认房间无误并且打扫，确认无误之后，就退房成功，订单完成。

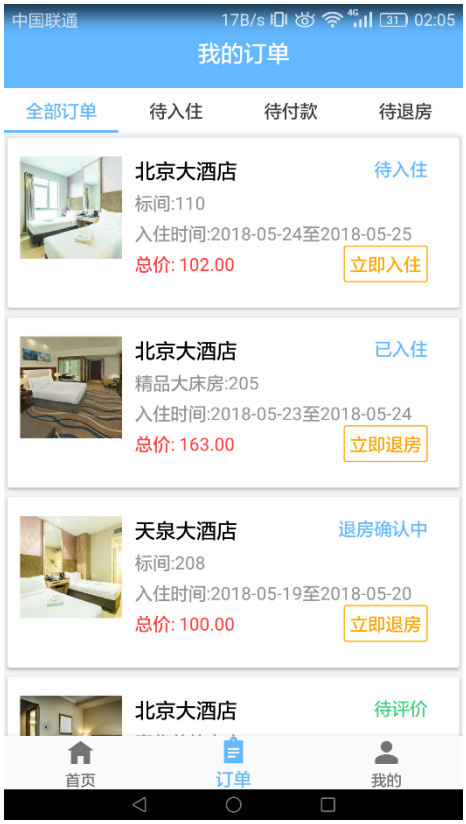

图 5-20 查看全部订单界面

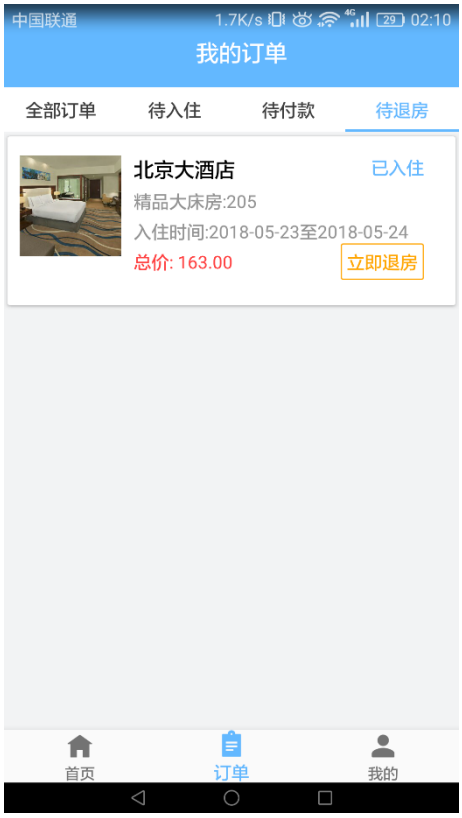

图 5-21 查看待退房订单界面

5.6 用户信息模块

当用户点击我的界面就会进入到用户信息模块，然后请求服务器去获取用户的个人信息，如果用户没有登录就跳转到登录界面。登录之后就会返回用户的个人信息。其中包括用户名、头像、手机号和是否进行了人证核验，如果没有进行人证核验，可以提示用户进行认证核验。点击实名认证就可以进入到人证核验界面。如果进行了人证核验，就提示用户已经通过实名认证了。

6 系统测试

6.1 测试目的

系统测试是软件开发流程中不可缺少的一部分，主要是在此阶段尽可能多的发现系统的问题，然后进行改正。测试的过程中一定要进行多方面的考虑，把程序的所有流程和会遇到的可能性都要走一遍，发现错误及时改正，这样才能达到测试的目的。

6.2 功能测试

在该系统中主要使用功能测试，就是测试系统中的所有功能是否符合用户的需求。下面是一些具体的测试用例。

6.2.1 用户登录测试

测试项目：用户登录

测试目的：检查用户是否成功登录，正确返回 token，错误提示错误信息

测试预置条件：已经注册过的用户信息

执行步骤：

（1）打开登录界面，输入用户名和密码；

（2）点击登录按钮。

预期结果：

（1）用户名密码正确返回 token，并保存到 SharePreference 中，然后跳转到主界面。

（2）用户名密码错误并返回错误信息，不进行跳转。

实际结果：与预期结果一致。下图为保存到手机中的 token 信息。

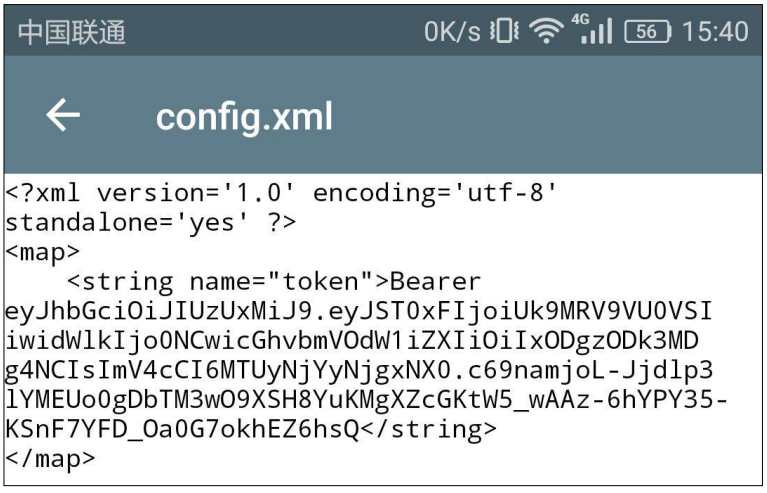

图 6-1 保存到手机中的 token 信息

6.2.2 用户注册测试

测试项目：用户注册

测试目的：查看用户是否可以成功注册，注册失败返回错误信息。

测试预置条件：注册需要填写的信息

执行步骤：

（1）打开注册界面，输入手机号；

（2）点击下一步按钮，输入用户名和密码；

（3）点击注册按钮。

预期结果：

（1）注册成功返回 token，并保存到 SharePreference 中，然后到主界面。

（2）注册失败返回错误信息，不进行跳转。

（3）用户信息输入错误进行提示。

实际结果：与预期结果一致。错误提示如下。

图 6-2 输入手机号错误提示

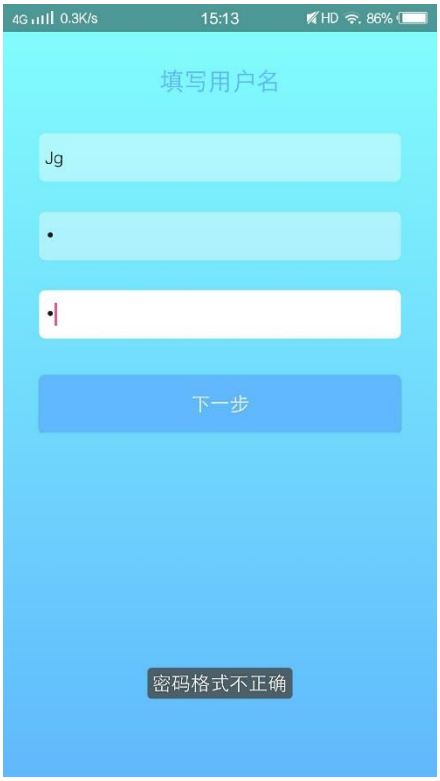

图 6-3 输入密码格式错误提示

6.2.3 人证核验测试

测试项目：人证核验

测试目的：查看用户是否可以进行人证核验

测试预置条件：拍摄身份证和人脸

执行步骤：

（1）打开人证核验界面，拍摄身份证正反面；

（2）点击下一步按钮，对比人脸信息；

（3）对比成功上传信息。

预期结果：

（1）上传成功，提示核验成功，然后跳转到主界面。

（2）上传失败返回错误信息，不进行跳转。

实际结果：与预期结果一致。

图 6-4 提示用户不要退出

图 6-5 人证核验成功提示

6.2.4 搜索酒店

测试项目：搜索酒店

测试目的：查看用户是否可以进行酒店搜索

测试预置条件：输入搜索条件

执行步骤：

（1）进入到主界面，输入要搜索的条件；

（2）点击搜索按钮，跳转到酒店列表界面。

预期结果：

（1）搜索到结果，显示酒店列表。

（2）未搜索到酒店，显示暂无符合要求的酒店。

（3）出现错误提示错误信息。

实际结果：与预期结果一致。

图 6-6 未查询到结果

6.2.1 预订酒店

测试项目：预订酒店

测试目的：查看用户是否可以进行酒店预订

测试预置条件：选择预订的酒店和房间号

执行步骤：

（1）进入到预订界面，选择要预订的房间号。

预期结果：

（1）预订成功，进入到确认订单界面；

（2）用户未登录，进入到登录界面；

（3）出现错误提示错误信息；

实际结果：与预期结果一致。

7 总结与展望

7.1 总结

在这个移动互联的时代，人们急切需求一款可以足不出户、随时随地通过自己的手机来浏览并预订快捷酒店的住宿信息管理系统。该系统基于共享经济和人工智能，利用共享经济的优势，让用户可以快速，方便入住酒店。然后结合人工智能的人脸识别算法，验证人脸和身份证信息来保证安全，可靠。最终实现方便用户的人证核验的共享酒店住宿信息管理系统。

在对系统进行设计前，我查阅了相关资料，阅读了大量相关文献，并对同类系统进行了学习与研究。开发利用 Android Studio 为工具，用 Java 语言设计和实现主要运用 C/S 模式，具有友好的界面，便于用户的操作和管理。系统使用流行的 MVP 模式，运用 MVPArms 框架，快速搭建一个 Dagger2+Rxjava+Retrofit 的 Android 应用。

基于人证核验的共享酒店用户端可以满足用户预订酒店的需求，该 APP 通过对市场的几家 APP 进行研究和对用户体验调研，设计出了简洁实用的 APP，通过对需求进行分析，把该 APP 一共分为四大模块：登录注册、酒店模块、订单模块和用户信息模块。首先对所有模块的业务进行总体设计，确定所有的业务需求，然后进行界面设计，紧接着对代码进行编写和业务实现，最后进行测试。测试结果完全符合预期结果和业务需求。用户可以通过该 APP 进行登录和注册，人证核验，通过指定条件进行搜索酒店和排序，查看酒店的详细信息和预订房间，查看订单信息，取消订单和申请离店退房等操作。用户完全可以在手机上进行预订、入住和退房的全部流程。对于用户非常方便快捷，并且采用人脸信息和身份证进行对比，保证了安全性。

通过对该系统的开发和实现，学习到了很多很多，本小组一共有三个人，从业务需求分析，到制定详细的业务逻辑，再到 APP 和服务器的接口和传输数据的确定。然后编写代码，最后到系统实现，通过测试。在这个过程中收获了很多，首先在软件开发上，了解了软件的开发流程，学会了分析实际业务和软件开发之间的联系，怎么实现具体的业务逻辑，学会了和小组中其它成员进行沟通和交流，共同解决遇到的问题。其次在技术上了解了 MVP 的设计模式，体会到了其中的精髓。还学会了 Android 相关的框架 Dagger2+Rxjava+Retrofit 的使用。除此之外，还使用了虹软的人脸识别 API、百度的文字识别 API 和高德地图 API。

总之通过对这个系统的设计和实现，从中学习到了很多实践性的东西，为未来找工作打下了良好的基础。

7.2 展望

通过对系统的测试和使用，系统满足了用户的基本需求。但由于时间和技术的问题，系统还存在一些不足。首先由于支付接口的限制，该系统未实现支付功能。其次由于时间的限制，该系统未实现评价功能和在地图上显示酒店并且进行导航功能。由于条件的限制，只在华为、小米和 VIVO 部分机型进行了测试，并未在其它机型进行测试。最后就是某些功能在用户体验方面不是很好。

通过以上的不足，如果时间和条件允许的情况下，将会在以后的开发中解决这些问题。并且逐步优化代码，提升系统的稳定性和流畅性；优化业务逻辑，减少一些不必要的操作；优化界面，让用户不仅在业务上有很好的体验还能喜欢上 APP 的界面；优化细节，让用户获得更好的使用体验。最终为用户打造一个简洁、美观、实用的 APP。

参考文献

[1] 李刚.疯狂 Android 讲义[M].北京:电子工业出版社,2015.6
[2] 满江月.生物特征识别技术的实现原理与前景分析[J].中国安防,2014(11):72-75
[3] 刘坤.基于人脸识别的身份认证系统的设计与开发[D].河北大学,2017
[4] 刘曦芝.基于 Android 平台人脸识别算法移植的研究[D]. 武汉理工大学,2015
[5] 明日科技.Java 从入门到精通[M].北京:清华大学出版社.2016
[6] 明日科技.Java Web 从入门到精通[M].北京：清华大学出版社.2016
[7] 欧阳燊. Android Studio 开发实战:从零基础到 APP 上线[M].北京：清华大学出版社.2017
[8] 孙卫琴. Tomcat 与 Java Web 开发技术详解[M].第 2 版.北京:电子工业出版社,2010
[9] 温谦. HTML+CSS 网页设计与布局从入门到精通[M].北京:人民邮电出版社,2014
[10] 李兴华.名师讲坛—Java Web 开发实战经典基础篇[M].北京:清华大学出版社,2014
[11] 张建兵,蔡长安.基于 JAVA 的 B2C 网上购物系统的设计与实现[J].重庆工商大学学报,2009,06:563-566
[12] 赵洋,张丽,王恩东,张素宁.基于 Structs+Spring+Hibernate 的 J2EE 的架构研究[J].现代电子技术,2009,2(289):107-110
[13] 贺松平.基于 MVC 模式的 B/S 架构的研究及应用[D].武汉:华中科技大学,2013
[14] 毕建信.基于 MVC 设计模式的 Web 应用研究与实现[D].武汉:武汉理工大学,2013
[15] 刘畅.基于 iOS 的国际酒店预订 APP 的设计与实现[D].北京:北京交通大学,2015
[16] 杨振娟.高职院校物资管理系统的设计与实现[D].河北:河北科技大学,2013
[17] 凌宇翔.华荣宾馆物资管理系统的设计与实现[D].成都:电子科技大学,2014
[18] Face Net:A Unified Embedding For Face Recognition And Clustering. Schroff F,Kalenichenko D,Philbin J. IEEE Conference on Computer Vision and Pattern . 2015
[19] Axsater S. Modelling emergency lateral transshipments in inventory systems[J]. Management Science, 2003, 15(2): 1369-1338
[20] Mobile Platform Architecture Review: Android, iPhone, Qt. M Lettner,M Tschernuth,R Mayrhofer. Lecture Notes in Computer Science . 2012
[21]Deep learning identity-preserving face space. Zhu Z,Luo P,Wang X,et al. 2013 IEEE International Conference on Computer Vision (ICCV) . 2013

致谢

从系统的实现到完成，再到论文的终稿，历时几个月，在这个过程中吴颖老师给予了我细心的指导，对系统的设计、实现和论文的修改方面提出了很多意见。导师渊博的知识、精益求精的工作态度、平易近人的人格魅力对我影响深远，不仅让我明确了学习目标、学习到了技术知识。还让我明白了许多为人处世的道理。在这里除了要感谢导师的悉心指导，也要感谢那些一起完成系统的同学们汪潭潭和张行，还要感谢在系统制作过程中给予帮助的同学高猛和李静文。他们的帮助让我更快更好的完成了系统和论文。虽然在系统的实现过程中遇到了很多问题，但通过吴颖导师的指导和帮助，最终完成了系统和论文。除此之外，也离不开大学四年各个老师教育和同学们的帮助，最后，对他们表示衷心地感谢！

附录

附录 A：主要源程序

1. 登录模块

```
/*验证手机号和密码的格式*/
public boolean verified(){
    if (!CommentUtils.isChinaPhoneLegal(etPwdPhone.getText().toString().trim())){
        ArmsUtils.makeText(this,"请输入正确的手机号码");
        return false;
    }
    if (!CommentUtils.rexCheckPassword(etPwdPwd.getText().toString().trim())){
        ArmsUtils.makeText(this,"请输入正确格式的密码");
    }
    return true;
}
/*请求服务器进行登录*/
public void login(String phone ,String  password){
    mModel.login(phone,password)
            .retryWhen(new RetryWithDelay(3,2))
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .doOnSubscribe(disposable -> {
                mRootView.showLoading();
            }).doFinally(()->{
                mRootView.hideLoading();
    }).subscribe(new ErrorHandleSubscriber<Result>(mErrorHandler) {
        @Override
        public void onNext(Result result) {
            if (result.getSuccess()){
                mRootView.startMain();
            }else{
                mRootView.showMessage(result.getMessage());
            }
        }
    });
}
```

2. 注册模块

```
/*验证用户名和密码格式*/
private boolean verifiedName() {
```

```
        String username = etUsername.getText().toString().trim();
        String password = etPassword.getText().toString().trim();
        String password1 = etPassword1.getText().toString().trim();
        if (username.isEmpty() || username == "") {
            ToastUtils.showToast(getActivity(), "用户名不能为空");
            return false;
        }
        if (password.isEmpty() || username == "") {
            ToastUtils.showToast(getActivity(), "密码不能为空");
            return false;
        }
        if (password1.isEmpty() || username == "") {
            ToastUtils.showToast(getActivity(), "确认密码不能为空");
            return false;
        }
        if (!CommentUtils.rexCheckPassword(password)){
            ToastUtils.showToast(getActivity(), "密码格式不正确");
            return  false;
        }
        if (!CommentUtils.rexCheckPassword(password1)){
            ToastUtils.showToast(getActivity(), "确认密码格式不正确");
            return false;
        }
        if(!password.endsWith(password1)){
            ToastUtils.showToast(getActivity(), "两次密码不一致");
            return false;
        }
        return true;
    }
/*请求服务器进行注册*/
    public void register(String phone, String username, String password) {
        mModel.register(phone, username, password)
                .retryWhen(new RetryWithDelay(3, 2))
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
                    mRootView.showLoading();
                }).doFinally(() -> {
            mRootView.hideLoading();
        }).observeOn(AndroidSchedulers.mainThread())
                .compose(RxLifecycleUtils.bindToLifecycle(mRootView))
                .subscribe(new ErrorHandleSubscriber<Result>(mErrorHandler) {
```

```
                    @Override
                    public void onNext(Result result) {
                        if (result.getSuccess()) {
                            mRootView.regData(true);
                        } else {
                            mRootView.showMessage(result.getMessage());
                        }
                    }
                });
    }
```

3. 人证核验模块

```
    /*处理拍摄完传回的身份证图片*/
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == SHOOT_ID_CARD_FRONT_RESULT) {
            switch (resultCode) {
                case RESULT_OK:
                    Bitmap bitmap = null;
                    Bundle b = data.getExtras(); //data 为 B 中回传的 Intent
                    frontUrl = b.getString("frontUrl");//str 即为回传的值
                    if (!TextUtils.isEmpty(frontUrl)) {
                        ivIdCardFront.setVisibility(View.GONE);
                        linearLayoutShowFront.setVisibility(View.VISIBLE);
                        showIvIdCardFront.setVisibility(View.VISIBLE);
                        bitmap = getCompressPhoto(frontUrl);
                        showIvIdCardFront.setImageBitmap(bitmap); //设置 Bitmap
                        mPresenter.baiduOCR(frontUrl);
                    } else {
                        ivIdCardFront.setVisibility(View.VISIBLE);
                        linearLayoutShowFront.setVisibility(View.GONE);
                        btnSubmit.setTextColor(getResources().getColor(R.color.submit_co
lor));
                        btnSubmit.setEnabled(false);
                    }
                    break;
                default:
            }
        } else if (requestCode == SHOOT_ID_CARD_SIDE_RESULT) {
            switch (resultCode) {
                case RESULT_OK:
```

```
                    Bitmap bitmap = null;
                    Bundle b = data.getExtras(); //data 为 B 中回传的 Intent
                    backUrl = b.getString("backUrl");//str 即为回传的值
                    if (!TextUtils.isEmpty(backUrl)) {
                        ivIdCardBack.setVisibility(View.GONE);
                        linearLayoutShowBack.setVisibility(View.VISIBLE);
                        bitmap = getCompressPhoto(backUrl);
                        showIvIdCardBack.setImageBitmap(bitmap); //设置 Bitmap
                        btnSubmit.setTextColor(getResources().getColor(R.color.white));
                        btnSubmit.setEnabled(flag);
                    } else {
                        ivIdCardBack.setVisibility(View.VISIBLE);
                        linearLayoutShowBack.setVisibility(View.GONE);
                        btnSubmit.setTextColor(getResources().getColor(R.color.submit_co
lor));
                        btnSubmit.setEnabled(false);
                    }
                    break;
                default:
            }
        }
    }
    /*对身份证进行文字识别*/
    public void baiduOCR(String path) {
        // 身份证识别参数设置
        IDCardParams param = new IDCardParams();
        param.setImageFile(new File(path));
        param.setIdCardSide(IDCardParams.ID_CARD_SIDE_FRONT);
        // 设置方向检测
        param.setDetectDirection(true);
        // 设置图像参数压缩质量 0-100, 越大图像质量越好但是请求时间越长。 不设置
则默认值为 20
        param.setImageQuality(100);
        // 调用身份证识别服务
        OCR.getInstance().recognizeIDCard(param, new OnResultListener<IDCardResult>()
{
            @Override
            public void onResult(IDCardResult result) {
                // 调用成功，返回 IDCardResult 对象
                mRootView.onSuccess(result);
            }
            @Override
```

```
            public void onError(OCRError error) {
                // 调用失败，返回 OCRError 对象
                mRootView.onError(error);
            }
        });
    }
    /*对身份证中的人脸进行识别*/
    @Override
    public void regFace(byte[] bytes) {
        AFD_FSDKError error1 = new AFD_FSDKError();
        List<AFD_FSDKFace> result2 = new ArrayList<>();
        AFD_FSDKEngine engine1 = new AFD_FSDKEngine();
        //人脸检测引擎
        error1 = engine1.AFD_FSDK_InitialFaceEngine(key,value, AFD_FSDKEngine.AFD
_OPF_0_HIGHER_EXT, 16, 1);
        error1 = engine1.AFD_FSDK_StillImageFaceDetection(bytes, 1920, 1080, AFD_FS
DKEngine.CP_PAF_NV21, result2);
        for (AFD_FSDKFace face : result2) {
            idFace = face.clone();
        }
        //销毁人脸检测引擎
        error1 = engine1.AFD_FSDK_UninitialFaceEngine();
        afr_fsdkError = engine2.AFR_FSDK_ExtractFRFeature(bytes, 1920, 1080, AFR_FS
DKEngine.CP_PAF_NV21, idFace.getRect(), idFace.getDegree(), face1);
}
/*识别摄像头中的人脸*/
public void process(byte[] data, int width, int height){
    //输入的 data 数据为 NV21 格式（如 Camera 里 NV21 格式的 preview 数据），其中 heig
ht 不能为奇数，人脸跟踪返回结果保存在 result。
    afd_fsdkError = engine.AFD_FSDK_StillImageFaceDetection(data, width, height, AFD_
FSDKEngine.CP_PAF_NV21, result);
    Log.d("com.arcsoft", "人脸检测 AFT_FSDK_FaceFeatureDetect =" + afd_fsdkError.getC
ode());

    if (result.size() > 1) {
        for (AFD_FSDKFace face : result) {
            Log.d("com.arcsoft", "Face:" + face.toString());
            resultFace = face.clone();
            afr_fsdkError = engine2.AFR_FSDK_ExtractFRFeature(data, width, height, AF
R_FSDKEngine.CP_PAF_NV21, resultFace.getRect(), resultFace.getDegree(), face2);
            Log.d(TAG, "347.process: " + resultFace.toString());
            if (compare(face1, face2)) {
```

```
                    mCamera.stopPreview();
                    Face userFace = (Face) face2;
                    Log.d(TAG, "327.process: "+Arrays.toString(userFace.getFeatureData()));
                    verifyUser.setFace(userFace.getFeatureData());
                    textView.setText("识别成功！");
                    btRegFace.setVisibility(View.VISIBLE);
                    break;
                }
            }
        } else {
            ArmsUtils.makeText(getActivity(), "没有检测到人脸，请调整方向重试！");
        }
    }
    /*对比身份证和人脸信息*/
    private boolean compare(AFR_FSDKFace face1, AFR_FSDKFace face2) {
        float grade;
        //score 用于存放人脸对比的相似度值
        AFR_FSDKMatching score = new AFR_FSDKMatching();
        //调用对比方法
        afr_fsdkError = engine2.AFR_FSDK_FacePairMatching(face1, face2, score);
        Log.d("com.arcsoft", "405AFR_FSDK_FacePairMatching=" + afr_fsdkError.getCode
    ());
        Log.d("com.arcsoft", "406Score:" + score.getScore());
        grade = score.getScore();
        ArmsUtils.makeText(getActivity(), "比对分数" + grade);
        if (grade >= 0.5) {
            return true;
        }
        return false;
    }
    /*发送实名人证信息到服务群殴*/
    public void verified(VerifyUser verifyUser) {
        mModel.verified(verifyUser)
                .retryWhen(new RetryWithDelay(3,2))
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
                    mRootView.showLoading();
                }).doFinally(() -> {
            mRootView.hideLoading();
        }).observeOn(Schedulers.io())
                .map(getResultData ->{
```

```
                return getResultData;
            })
            .subscribe(new ErrorHandleSubscriber<Result>(mErrorHandler) {
                @Override
                public void onNext(Result result) {
                    if(result.getSuccess()){
                        Timber.d("====="+result.toString());
                        mRootView.startPage();
                    }else{
                        mRootView.showMessage(result.getMessage());
                    }
                }
            });
    }
```

4. 酒店模块

```
    /*获取位置信息*/
     private void getLocation() {
        AMapLocationClientOption mLocationOption = new AMapLocationClientOption();

        //声明定位回调监听器
        AMapLocationListener mLocationListener = new AMapLocationListener() {
            @Override
            public void onLocationChanged(AMapLocation aMapLocation) {
                if (aMapLocation.getErrorCode() == 0) {
                    coordinate.setLatitude(aMapLocation.getLatitude());
                    coordinate.setLongitude(aMapLocation.getLongitude());
                    //aMaoLocation 是位置信息
                    currentLocation=aMapLocation.getCity() + aMapLocation.getDistrict()
+ aMapLocation.getStreet();
                    etDestination.setText(currentLocation);
                    Log.d(TAG, "onLocationChanged: " + aMapLocation.getCity() + aM
apLocation.getDistrict() + aMapLocation.getStreet());
                    Log.d(TAG, "onLocationChanged: "+aMapLocation.getLongitude()+"--
"+aMapLocation.getLatitude());
                } else {
                    //定位失败时，可通过 ErrCode（错误码）信息来确定失败的原因，
errInfo 是错误信息，详见错误码表。
                    Log.e("AmapError", "location Error, ErrCode:"
                            + aMapLocation.getErrorCode() + ", errInfo:"
                            + aMapLocation.getErrorInfo());
                }
```

```
            }
        };
        //初始化定位
        mLocationClient = new AMapLocationClient(getActivity().getApplicationContext());
        //设置定位回调监听
        mLocationClient.setLocationListener(mLocationListener);
        mLocationOption =
                //获取一次定位结果：
                //该方法默认为 false。
                mLocationOption.setOnceLocation(true);

        //获取最近 3s 内精度最高的一次定位结果：
        //设置 setOnceLocationLatest(boolean b)接口为 true，启动定位时 SDK 会返回最近
3s 内精度最高的一次定位结果。如果设置其为 true，setOnceLocation(boolean b)接口也会被
设置为 true，反之不会，默认为 false。
        mLocationOption.setOnceLocationLatest(true);
        //给定位客户端对象设置定位参数
        mLocationClient.setLocationOption(mLocationOption);
        mLocationClient.startLocation();
    }
    /*搜索酒店 获取酒店列表*/
    public void getHotelData(SearchHotel<List<Hotel>> searchHotel){
        mSearchHotel = searchHotel;
        if (searchHotel.getCurrentPage()==1){
            //如果是第一页 证明要重新加载  所以给 adapter 致为空
            hotelListAdapter=null;
        }
        mModel.getHotelData(searchHotel)
                .retryWhen(new RetryWithDelay(3,2))
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
                    mRootView.showLoading();
                }).doFinally(() -> {
            mRootView.hideLoading();
        }).observeOn(Schedulers.io())
                .map(getHotelDataResult -> {
                    if (getHotelDataResult.getData().getCurrentPage()==1&&(getHotelData
Result.getData()==null||getHotelDataResult.getData().getData().isEmpty())){
                        mRootView.showError();
                    }
                    return getHotelDataResult.getData().getData();
```

```
                }).observeOn(AndroidSchedulers.mainThread())
                .compose(RxLifecycleUtils.bindToLifecycle(mRootView))
                .subscribe(new ErrorHandleSubscriber<List<Hotel>>(mErrorHandler) {
                    @Override
                    public void onNext(List<Hotel> hotels) {
                        if (!hotels.isEmpty()) {
                            setAdapter(hotels);
                        }else {
                            mRootView.showMessage("已经是最后一页了！");
                            hotelListAdapter.setEnableLoadMore(false);
                        }
                    }
                });
    }
    /*对酒店进行排序*/
    zhpxView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            zhpxAdapter.setCheckItem(position);//获取点击的是那一项
            mDropDownMenu.setTabText(position == 0 ? headers[0] : zhpx[position]);//改变
下拉菜单的选中项
            mDropDownMenu.closeMenu();//关闭下拉菜单
            key.setSortKey(szhpx[position]);//重新设置关键字
            key.setSort("DESC");
            key.setCurrentPage(1);
            mPresenter.getHotelData(key);//调用 P 层方法请求服务器
        }
    });
    /*显示酒店的详细信息*/
    @Override
    public void setView(HotelDetail hotelDetail, TypeListAdapter adapter) {
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        ArmsUtils.obtainAppComponentFromContext(this).imageLoader().loadImage(this, Ima
geConfigImpl.builder().imageView(imageView).url(hotelDetail.getImageUrl()).build());
        tvScore.setText(hotelDetail.getScore().toString());
        tvAddress.setText(hotelDetail.getAddressDes());
        this.hotelDetail = hotelDetail;
        recyclerView.setAdapter(adapter);
        adapter.setOnItemClickListener(new BaseQuickAdapter.OnItemClickListener() {
            @Override
            public void onItemClick(BaseQuickAdapter adapter, View view, int position)
{
```

```
                showRoomTypeDes();
                TextView textView = view.findViewById(R.id.tv_room_type_id);
                Integer roomTypeId = new Integer(textView.getText().toString());
                mPresenter.getRoomTypeDetail(hotelDetail.getHotelId(), roomTypeId);
            }
        });
        adapter.setOnItemChildClickListener(new BaseQuickAdapter.OnItemChildClickListen
er() {
            @Override
            public void onItemChildClick(BaseQuickAdapter adapter, View view, int posit
ion) {
                switch (view.getId()) {
                    case R.id.bt_book:
                        showPopRoomNumber();
                        Log.d(TAG, "234.onItemChildClick: " + position);
                        TextView textView = (TextView) adapter.getViewByPosition(recy
clerView, position, R.id.tv_room_type_id);
                        Integer roomTypeId = new Integer(textView.getText().toString());
                        mPresenter.getRoomNumber(hotelDetail.getHotelId(), roomTypeId,
 startDay + " 12:00:00", endDay + " 12:00:00");
                        break;
                    default:
                        Log.d(TAG, "237.onItemChildClick: =========");
                }
            }
        });
        init();
    }
    /*弹出层显示房间的详细信息*/
        private void showRoomTypeDes() {
            View popView = View.inflate(this, R.layout.pop_view_room_type_detail, null);
            tvTypeNameDes = popView.findViewById(R.id.tv_type_name_des);
            typePhoto = popView.findViewById(R.id.iv_type_photo);
            tvWifi = popView.findViewById(R.id.wifi);
            tvWindow = popView.findViewById(R.id.tv_window);
            tvBathroom = popView.findViewById(R.id.tv_bathroom);
            tvCapacity = popView.findViewById(R.id.tv_capacity);
            tvBedDes = popView.findViewById(R.id.tv_bed_des);
            tvLiveDes = popView.findViewById(R.id.tv_live_des);
            int weight = getResources().getDisplayMetrics().widthPixels;
            int height = getResources().getDisplayMetrics().heightPixels * 2 / 3;
            popupWindow = new PopupWindow(popView, weight, height);
```

```
                popupWindow.setFocusable(true);
                //点击外部 popueWindow 消失
                popupWindow.setOutsideTouchable(true);
                //popupWindow 消失屏幕变为不透明
                popupWindow.setOnDismissListener(new PopupWindow.OnDismissListener() {
                    @Override
                    public void onDismiss() {
                        WindowManager.LayoutParams lp = getWindow().getAttributes();
                        lp.alpha = 1.0f;
                        getWindow().setAttributes(lp);
                    }
                });
                //popupWindow 出现屏幕变为半透明
                WindowManager.LayoutParams lp = getWindow().getAttributes();
                lp.alpha = 0.5f;
                getWindow().setAttributes(lp);
                popupWindow.showAtLocation(popView, Gravity.BOTTOM, 0, 0);
            }
        /*展示可用房间号 点击并进行预订*/
        @Override
        public void showRoomNumber(RoomType roomType) {
            Log.d(TAG, "265.showRoomNumber: " + roomType.toString());
            rvRoomNumber.setLayoutManager(new GridLayoutManager(this, 3));
            RoomListAdapter roomListAdapter = new RoomListAdapter(R.layout.item_room_nu
mber, roomType.getRooms());
            rvRoomNumber.setAdapter(roomListAdapter);
            roomListAdapter.setOnItemClickListener(new BaseQuickAdapter.OnItemClickListener
() {
                @Override
                public void onItemClick(BaseQuickAdapter adapter, View view, int position)
{
                    TextView textView = view.findViewById(R.id.tv_room_number);
                    String roomNumber = textView.getText().toString().trim();
                    GenerateOrder generateOrder = new GenerateOrder();
                    generateOrder.setRoomId(roomType.getRooms().get(position).getRoomId());
                    generateOrder.setHotelId(hotelDetail.getHotelId());
                    generateOrder.setTypeId(roomType.getTypeId());
                    generateOrder.setRoomNum(roomNumber);
                    generateOrder.setStartTime(startDay + " 12:00:00");
                    generateOrder.setEndTime(endDay + " 12:00:00");
                    Log.d(TAG, "288.onItemClick: " + generateOrder.toString());
                    popupWindow.dismiss();
```

```
                mPresenter.getUserOrderData(generateOrder);

            }
        });
    }
    /*订单信息发送到服务器生成订单*/
    public void getUserOrderData(GenerateOrder generateOrder){
        mModel.generateOrder(generateOrder)
                .retryWhen(new MyRetryWithDelay(3,2))
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
                    mRootView.showLoading();
                }).doFinally(()->{
            mRootView.hideLoading();
        }).observeOn(Schedulers.io())
                .map(getUserOrder->{
                    return getUserOrder.getData();
                }).observeOn(AndroidSchedulers.mainThread())
                .compose(RxLifecycleUtils.bindToLifecycle(mRootView))
                .subscribe(new ErrorHandleSubscriber<UserOrder>(mErrorHandler) {
                    @Override
                    public void onNext(UserOrder userOrder) {
                        mRootView.startGenerateActivity(userOrder);
                    }
                });
    }
```

5. 订单模块

```
    /*获取用户的全部订单*/
    public void findUserOrder(){
        mModel.findUserOrder()
                .retryWhen(new MyRetryWithDelay(3,2))
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
                    mRootView.showLoading();
                }).doFinally(() -> {
                    mRootView.hideLoading();
                }).observeOn(Schedulers.io())
                .map(findUserOrder -> {
                    return findUserOrder.getData();
```

```
                }).observeOn(AndroidSchedulers.mainThread())
                .compose(RxLifecycleUtils.bindToLifecycle(mRootView))
                .subscribe(new ErrorHandleSubscriber<List<UserOrder>>(mErrorHandler)
{

                    @Override
                    public void onNext(List<UserOrder> userOrders) {
                        mRootView.setAdapter(userOrders);
                    }
                });
    }
    /*获取指定状态订单*/
    public void findUserOrderByStatus(Integer status){
        mModel.findUserOrderByStatus(status)
                .retryWhen(new MyRetryWithDelay(3,2))
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
                    mRootView.showLoading();
                }).doFinally(() -> {
            mRootView.hideLoading();
        }).observeOn(Schedulers.io())
                .map(findUserOrder -> {
                    return findUserOrder.getData();
                }).observeOn(AndroidSchedulers.mainThread())
                .compose(RxLifecycleUtils.bindToLifecycle(mRootView))
                .subscribe(new ErrorHandleSubscriber<List<UserOrder>>(mErrorHandler)
{

                    @Override
                    public void onNext(List<UserOrder> userOrders) {
                        mRootView.setAdapter(userOrders);
                    }
                });
    }
    /*申请退房*/
    public void checkOutHotel(String orderId){
        mModel.checkOutHotel(orderId)
                .retryWhen(new MyRetryWithDelay(3,2))
                . subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
```

```
                mRootView.showLoading();
            }).doFinally(() -> {
                mRootView.hideLoading();
            }).compose(RxLifecycleUtils.bindToLifecycle(mRootView))
            .subscribe(new ErrorHandleSubscriber<Result>(mErrorHandler) {

                @Override
                public void onNext(Result result) {
                    if(result.getSuccess()) {
                        mRootView.showMessage("申请退房成功，请等待审核");
                        findUserOrderByStatus(2);
                    }else {
                        mRootView.showMessage("申请对方失败");
                    }
                }
            });
    }
```

6. 用户模块

```
    /*获取用户的个人信息*/
    public void getUserInfo(){
        mModel.getUserInfo()
                .retryWhen(new MyRetryWithDelay(2,3))
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .doOnSubscribe(disposable -> {
                    mRootView.showLoading();
                }).doFinally(() -> {
            mRootView.hideLoading();
        }).observeOn(Schedulers.io())
                .map(getMainInfo->{
                    return getMainInfo.getData();
                }).observeOn(AndroidSchedulers.mainThread())
                .subscribe(new ErrorHandleSubscriber<MineInfo>(mErrorHandler) {
            @Override
            public void onNext(MineInfo MineInfo) {
                mRootView.showData(MineInfo);
            }
        });
    }

    /*退出登录*/
```

```
    @OnClick(R.id.bt_sign_out)
    public void signOut(View view){
        final AlertDialog.Builder dialog =
                new AlertDialog.Builder(getActivity());
        dialog.setTitle("提示");
        dialog.setMessage("您确定要退出登录？");
        dialog.setPositiveButton("退出",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        SharedPreferences sharedPreferences = getActivity().getSharedPre
ferences("config", Context.MODE_PRIVATE);
                        SharedPreferences.Editor editor = sharedPreferences.edit();
                        editor.putString("token","");
                        editor.commit();
                        ArmsUtils.makeText(getActivity(),"您已成功退出");
                        ArmsUtils.killAll();
                        ArmsUtils.startActivity(MainActivity.class);
                    }
                });
        dialog.setNegativeButton("取消",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
        // 显示
        dialog.show();
    }
```

附录 B：软件使用说明书

人证核验的共享酒店住宿信息管理系统使用说明

1 软件概述

本软件是基于 Android 的人证核验共享酒店住宿信息管理系统用户端 APP，用户可在 APP 进行预订酒店等操作，效率高且方便实用。

1.1 功能

其中包括登录模块、注册模块、人证核验模块、酒店模块、订单模块和用户信息模块。登录模块和注册模块为用户提供注册和登录功能。人证核验就是对比身份证照片和实时人脸。酒店模块主要实现了用户搜索酒店，对酒店进行排序和查看酒店、房间详情等。订单模块提供了预订功能、查看订单功能和在线退房等。用户模块主要提供查看用户信息、获取帮助和退出登录等操作。

1.2 原理

用户 APP 系统使用 MVPArms 架构，使用 Dagger2+Rxjava+Retrofit 搭建，使用 Java 语言编写。后台服务器使用 SSM 技术+Spring Boot 开发，数据库使用 MySQL。

2 软件安装

2.1 系统要求

要求 Android 系统在 5.0 以上

2.2 安装

在源程序压缩包下 app/release 里面有程序安装包，只需安装对应版本即可。

附录 C：光盘

光盘说明

1. 目录结构

光盘内包含：实习报告、毕业论文、开题报告、毕业论文相关文件、答辩 PPT 和一个源程序压缩包。

2. 文件说明

源程序压缩包在一个文件中，各种文档在相应的文件夹中。