**火车票售票系统设计报告**

目录

[一、系统开发平台](#_Toc8774_WPSOffice_Level1) [4](#_Toc8774_WPSOffice_Level1)

[二、数据库规划](#_Toc21050_WPSOffice_Level1) [5](#_Toc21050_WPSOffice_Level1)

[2.1 任务陈述](#_Toc21050_WPSOffice_Level2) [5](#_Toc21050_WPSOffice_Level2)

[2.2 任务目标](#_Toc29254_WPSOffice_Level2) [5](#_Toc29254_WPSOffice_Level2)

[三、 系统定义](#_Toc29254_WPSOffice_Level1) [6](#_Toc29254_WPSOffice_Level1)

[3.1 系统边界](#_Toc32516_WPSOffice_Level2) [6](#_Toc32516_WPSOffice_Level2)

[3.2 用户视图](#_Toc30101_WPSOffice_Level2) [7](#_Toc30101_WPSOffice_Level2)

[3.2.1 普通用户(User)视图](#_Toc21050_WPSOffice_Level3) [7](#_Toc21050_WPSOffice_Level3)

[3.2.2 管理员(Manager)视图](#_Toc29254_WPSOffice_Level3) [7](#_Toc29254_WPSOffice_Level3)

[四、需求分析](#_Toc32516_WPSOffice_Level1) [7](#_Toc32516_WPSOffice_Level1)

[4.1 用户需求说明](#_Toc20788_WPSOffice_Level2) [7](#_Toc20788_WPSOffice_Level2)

[4.1.1 数据需求](#_Toc32516_WPSOffice_Level3) [7](#_Toc32516_WPSOffice_Level3)

[4.1.2 事务需求](#_Toc30101_WPSOffice_Level3) [8](#_Toc30101_WPSOffice_Level3)

[1.数据录入](#_Toc20788_WPSOffice_Level3) [8](#_Toc20788_WPSOffice_Level3)

[2.数据更新](#_Toc32591_WPSOffice_Level3) [8](#_Toc32591_WPSOffice_Level3)

[3.数据查询](#_Toc30026_WPSOffice_Level3) [8](#_Toc30026_WPSOffice_Level3)

[4.2 系统需求说明](#_Toc32591_WPSOffice_Level2) [9](#_Toc32591_WPSOffice_Level2)

[4.2.1 初始数据库大小](#_Toc2279_WPSOffice_Level3) [9](#_Toc2279_WPSOffice_Level3)

[4.2.2 网络和共享需求](#_Toc18961_WPSOffice_Level3) [9](#_Toc18961_WPSOffice_Level3)

[4.2.3 性能](#_Toc15486_WPSOffice_Level3) [9](#_Toc15486_WPSOffice_Level3)

[4.2.4 安全性](#_Toc1447_WPSOffice_Level3) [9](#_Toc1447_WPSOffice_Level3)

[4.2.5 法律问题](#_Toc16972_WPSOffice_Level3) [9](#_Toc16972_WPSOffice_Level3)

[五、数据库逻辑设计](#_Toc30101_WPSOffice_Level1) [10](#_Toc30101_WPSOffice_Level1)

[5.1 E-R图](#_Toc30026_WPSOffice_Level2) [10](#_Toc30026_WPSOffice_Level2)

[5.1.1 用户实体](#_Toc6058_WPSOffice_Level3) [10](#_Toc6058_WPSOffice_Level3)

[5.1.2 管理员实体](#_Toc15004_WPSOffice_Level3) [10](#_Toc15004_WPSOffice_Level3)

[5.1.3 火车实体](#_Toc22997_WPSOffice_Level3) [11](#_Toc22997_WPSOffice_Level3)

[5.1.4 火车经停站实体](#_Toc20587_WPSOffice_Level3) [11](#_Toc20587_WPSOffice_Level3)

[5.1.6 火车票价格实体](#_Toc27292_WPSOffice_Level3) [12](#_Toc27292_WPSOffice_Level3)

[5.1.7 订单实体](#_Toc29215_WPSOffice_Level3) [13](#_Toc29215_WPSOffice_Level3)

[5.1.8 总E-R图](#_Toc10627_WPSOffice_Level3) [13](#_Toc10627_WPSOffice_Level3)

[5.2 数据字典](#_Toc2279_WPSOffice_Level2) [14](#_Toc2279_WPSOffice_Level2)

[5.2.1 用户实体](#_Toc26879_WPSOffice_Level3) [14](#_Toc26879_WPSOffice_Level3)

[5.2.2 管理员实体](#_Toc19264_WPSOffice_Level3) [14](#_Toc19264_WPSOffice_Level3)

[5.2.3 火车实体](#_Toc3404_WPSOffice_Level3) [14](#_Toc3404_WPSOffice_Level3)

[5.2.4 火车经停站实体](#_Toc27631_WPSOffice_Level3) [15](#_Toc27631_WPSOffice_Level3)

[5.2.5 火车座位实体](#_Toc1059_WPSOffice_Level3) [15](#_Toc1059_WPSOffice_Level3)

[5.2.6 火车票价格实体](#_Toc30686_WPSOffice_Level3) [15](#_Toc30686_WPSOffice_Level3)

[5.2.7 订单实体](#_Toc7582_WPSOffice_Level3) [16](#_Toc7582_WPSOffice_Level3)

[六、数据库物理设计](#_Toc20788_WPSOffice_Level1) [17](#_Toc20788_WPSOffice_Level1)

[6.1 索引](#_Toc18961_WPSOffice_Level2) [17](#_Toc18961_WPSOffice_Level2)

[（1）用户表以username为索引](#_Toc18359_WPSOffice_Level3) [17](#_Toc18359_WPSOffice_Level3)

[（2）管理员表以managerid为索引](#_Toc23756_WPSOffice_Level3) [17](#_Toc23756_WPSOffice_Level3)

[（3）火车表以trainid为索引](#_Toc29469_WPSOffice_Level3) [17](#_Toc29469_WPSOffice_Level3)

[（4）火车经停站表以startstation为索引](#_Toc16491_WPSOffice_Level3) [17](#_Toc16491_WPSOffice_Level3)

[（5）火车座位表以trainid为索引](#_Toc13549_WPSOffice_Level3) [17](#_Toc13549_WPSOffice_Level3)

[（6）火车票价格表以startstation为索引](#_Toc23191_WPSOffice_Level3) [17](#_Toc23191_WPSOffice_Level3)

[（7）订单表以orderid为索引](#_Toc22206_WPSOffice_Level3) [17](#_Toc22206_WPSOffice_Level3)

[6.2 视图](#_Toc15486_WPSOffice_Level2) [17](#_Toc15486_WPSOffice_Level2)

[6.3 安全机制](#_Toc1447_WPSOffice_Level2) [18](#_Toc1447_WPSOffice_Level2)

[6.3.1 系统安全](#_Toc30405_WPSOffice_Level3) [18](#_Toc30405_WPSOffice_Level3)

[6.3.2 数据安全](#_Toc1385_WPSOffice_Level3) [18](#_Toc1385_WPSOffice_Level3)

[七、应用程序设计](#_Toc32591_WPSOffice_Level1) [18](#_Toc32591_WPSOffice_Level1)

[7.1 功能模块](#_Toc16972_WPSOffice_Level2) [18](#_Toc16972_WPSOffice_Level2)

[7.2 界面设计](#_Toc6058_WPSOffice_Level2) [19](#_Toc6058_WPSOffice_Level2)

[7.2.1 登录界面](#_Toc18629_WPSOffice_Level3) [19](#_Toc18629_WPSOffice_Level3)

[7.2.2 用户界面](#_Toc16010_WPSOffice_Level3) [19](#_Toc16010_WPSOffice_Level3)

[7.2.3管理员界面](#_Toc13471_WPSOffice_Level3) [22](#_Toc13471_WPSOffice_Level3)

[7.3 事务设计](#_Toc15004_WPSOffice_Level2) [24](#_Toc15004_WPSOffice_Level2)

[7.3.1 注册新用户](#_Toc18275_WPSOffice_Level3) [25](#_Toc18275_WPSOffice_Level3)

[7.3.2 登录](#_Toc32387_WPSOffice_Level3) [26](#_Toc32387_WPSOffice_Level3)

[7.3.3 查询列车](#_Toc24457_WPSOffice_Level3) [28](#_Toc24457_WPSOffice_Level3)

[7.3.4 选座购票](#_Toc27440_WPSOffice_Level3) [30](#_Toc27440_WPSOffice_Level3)

[7.3.5 用户查看订单](#_Toc13889_WPSOffice_Level3) [34](#_Toc13889_WPSOffice_Level3)

[7.3.6 改签](#_Toc23663_WPSOffice_Level3) [36](#_Toc23663_WPSOffice_Level3)

[7.3.7 退票](#_Toc8053_WPSOffice_Level3) [37](#_Toc8053_WPSOffice_Level3)

[7.3.8 查看余额](#_Toc4765_WPSOffice_Level3) [38](#_Toc4765_WPSOffice_Level3)

[7.3.9 增加、删除、查看列车](#_Toc11693_WPSOffice_Level3) [39](#_Toc11693_WPSOffice_Level3)

[7.3.10 订单查询](#_Toc17241_WPSOffice_Level3) [40](#_Toc17241_WPSOffice_Level3)

[7.3.11 封号解封](#_Toc16494_WPSOffice_Level3) [44](#_Toc16494_WPSOffice_Level3)

[八、测试和运行](#_Toc30026_WPSOffice_Level1) [45](#_Toc30026_WPSOffice_Level1)

[8.1 登录](#_Toc22997_WPSOffice_Level2) [45](#_Toc22997_WPSOffice_Level2)

[（1） 用户名不存在](#_Toc30418_WPSOffice_Level3) [45](#_Toc30418_WPSOffice_Level3)

[（2） 账号被封号](#_Toc21227_WPSOffice_Level3) [45](#_Toc21227_WPSOffice_Level3)

[（3） 密码错误](#_Toc26694_WPSOffice_Level3) [45](#_Toc26694_WPSOffice_Level3)

[（4） 密码正确](#_Toc29935_WPSOffice_Level3) [46](#_Toc29935_WPSOffice_Level3)

[8.2 用户系统](#_Toc20587_WPSOffice_Level2) [46](#_Toc20587_WPSOffice_Level2)

[（1） 选择城市](#_Toc19376_WPSOffice_Level3) [47](#_Toc19376_WPSOffice_Level3)

[（2） 选择发车日期](#_Toc26729_WPSOffice_Level3) [47](#_Toc26729_WPSOffice_Level3)

[（3） 查询](#_Toc10683_WPSOffice_Level3) [47](#_Toc10683_WPSOffice_Level3)

[（4）选座购票](#_Toc28673_WPSOffice_Level3) [48](#_Toc28673_WPSOffice_Level3)

[（5）查看订单](#_Toc19566_WPSOffice_Level3) [48](#_Toc19566_WPSOffice_Level3)

[（6）查看订单详情](#_Toc32664_WPSOffice_Level3) [49](#_Toc32664_WPSOffice_Level3)

[（7）显示二维码](#_Toc19000_WPSOffice_Level3) [49](#_Toc19000_WPSOffice_Level3)

[（8）改签提示](#_Toc17839_WPSOffice_Level3) [50](#_Toc17839_WPSOffice_Level3)

[（9）确定改签](#_Toc27193_WPSOffice_Level3) [51](#_Toc27193_WPSOffice_Level3)

[（10）重复购票操作](#_Toc31995_WPSOffice_Level3) [51](#_Toc31995_WPSOffice_Level3)

[（11）退票提示](#_Toc13314_WPSOffice_Level3) [51](#_Toc13314_WPSOffice_Level3)

[（12）退票成功](#_Toc12217_WPSOffice_Level3) [51](#_Toc12217_WPSOffice_Level3)

[（13）显示余额](#_Toc19998_WPSOffice_Level3) [52](#_Toc19998_WPSOffice_Level3)

[8.3 管理员系统](#_Toc27292_WPSOffice_Level2) [52](#_Toc27292_WPSOffice_Level2)

[（1）列车管理界面：](#_Toc8830_WPSOffice_Level3) [52](#_Toc8830_WPSOffice_Level3)

[（2）添加经停站：](#_Toc10789_WPSOffice_Level3) [52](#_Toc10789_WPSOffice_Level3)

[（3）删除列车：](#_Toc2701_WPSOffice_Level3) [53](#_Toc2701_WPSOffice_Level3)

[（4）查看列车](#_Toc12842_WPSOffice_Level3) [53](#_Toc12842_WPSOffice_Level3)

[（5）订单查询页面：](#_Toc5128_WPSOffice_Level3) [54](#_Toc5128_WPSOffice_Level3)

[（6）按用户名查询：](#_Toc2746_WPSOffice_Level3) [54](#_Toc2746_WPSOffice_Level3)

[（7）按发车日期查询：](#_Toc7794_WPSOffice_Level3) [55](#_Toc7794_WPSOffice_Level3)

[（8）按下单日期查询：](#_Toc21968_WPSOffice_Level3) [55](#_Toc21968_WPSOffice_Level3)

[（9）用户管理页面：](#_Toc2643_WPSOffice_Level3) [56](#_Toc2643_WPSOffice_Level3)

[（10）封号或解封](#_Toc1638_WPSOffice_Level3) [56](#_Toc1638_WPSOffice_Level3)

[九、总结](#_Toc2279_WPSOffice_Level1) [56](#_Toc2279_WPSOffice_Level1)

[9.1 系统优点](#_Toc29215_WPSOffice_Level2) [57](#_Toc29215_WPSOffice_Level2)

[9.2 系统改进](#_Toc10627_WPSOffice_Level2) [57](#_Toc10627_WPSOffice_Level2)

[9.3 经验与收获](#_Toc26879_WPSOffice_Level2) [57](#_Toc26879_WPSOffice_Level2)

**一、系统开发平台**

题目、开发工具、数据库、操作系统等。

题目：火车票售票系统

开发工具：Android Studio

数据库：SQLite

操作系统：Windows 10

**二、数据库规划**

**2.1 任务陈述**

为了火车客运系统的正常运行，需要一个强大的火车票售票系统，包括用户端的购票系统和管理员端的管理系统，以此来提高工作效率。

用户端的购票系统具有车次查询，余票查询，选座购票，订单查询，改签，退票，余额管理等功能；管理员端的管理系统具有车次管理（增加列车，删除列车，查询列车信息），订单查询（按用户名，发车时间，下单时间查询订单），用户管理（封号，解封）等功能。此外，系统还应具有异常处理功能。

系统开发的总体任务是实现火车票销售中各环节的系统化、规范化和自动

**2.2 任务目标**

|  |  |
| --- | --- |
| **目标类型** | **子目标** |
| 用户系统  目标 | 车次查询，选择出发城市，到达城市，出发时间查询满足条件的车次 |
| 余票查询，查询列车各类型座位的余票数，选座购票，生成订单 |
| 订单查询，查询当前用户所有订单；余额查询，查询当前用户余额 |
| 改签；退票 |
| 管理员系  统目标 | 车次管理，包括增加列车，删除列车，查询列车信息 |
| 订单查询，包括按用户名查询订单，按发车时间查询订单，按下单时间查询订单 |
| 用户管理，包括封号，解封 |

1. **系统定义**

**3.1 系统边界**

系统边界描述数据库系统和火车票售票系统的其他部分的接口，是火车票售票内部构成元素与外部有联系实体之间的信息关系的描述与分割。它并不需要在它们之间划一条物 理边界，而只需要弄清它们之间信息输入与输出的分割。

数据库系统的内部构成元素与外部其他部分之间的信息关系如下图所示：

用户

火车经停站

火车

订单

火车票价格

火车座位

管理员

**3.2 用户视图**

**3.2.1 普通用户(User)视图**

（1）车次查询，选择出发城市，到达城市，出发时间来查询满足条件的车次。

（2）余票查询，查询列车各类型座位的余票数。

（3）选座（A|B|C|D|F）购票，购票成功后生成订单。

（4）订单查询，查询当前用户所有订单的详细信息。

（5）余额查询，查询当前用户余额。

（6）改签，用户可以在发车之前更改列车，但出发城市，到达城市和发车时间必须和原订单一致，费用多退少补，没有手续费。

（7）退票，用户可以在发车之前退票，如果距离发车时间超过两天全额退款，如果距离发车时间不足两天，则扣除10%的手续费，返回90%的金额。

**3.2.2 管理员(Manager)视图**

（1）车次管理，包括增加列车，删除列车，查询列车信息。

（2）订单查询，包括按用户名查询订单，按发车时间查询订单，按下单时间查询订单。

（3）用户管理，包括封号，解封。

**四、需求分析**

**4.1 用户需求说明**

**4.1.1 数据需求**

每天有多列火车，在固定的一天内火车的编号唯一，经过多个经停站。火车实体的信息包括火车日期，火车编号，火车类型，火车名，出发城市，出发站，出发时间，到达城市，到达站，到达时间；经停站实体的信息包括日期，火车编号，城市名，火车站名，到达时间，离开时间，经停站编号；火车票价格实体的信息包括火车编号，出发站，到达站，座位等级，价格；用户可以设置出发城市，到达城市和发车时间来查找满足条件的列车，管理员可以查看所有列车的信息。用户查询到某次列车的之后，可以查看该列车的余票以及价格，然后选座购票，火车座位实体的信息包括日期，火车编号，车厢编号，行编号，位置，座位等级，起始占用编号，终止占用编号；用户购票成功之后系统生成订单，订单实体包含的信息有订单编号，用户名，火车编号，出发站，到达站，车厢编号，行编号，位置，座位等级，价格，下单时间。

管理员可以添加或删除列车，添加列车时设置列车的编号，火车类型，火车名，出发城市，出发站，出发时间，到达城市，到达站，到达时间，以及各经停站的城市名，火车站名，到达时间，离开时间，价格。管理员还可以进行订单查询包括按用户名查询

订单，按发车时间查询订单，按下单时间查询订单。

用户实体的信息包括用户名，密码，用户状态（是否被封号），用户余额（初始为10000）；管理员实体的信息包括管理员编号，管理员名，管理员密码

**4.1.2 事务需求**

1.数据录入

（1）新用户注册时录入用户的信息。

（2）管理员添加列车时录入列车的基本信息。

（3）管理员添加列车时录入列车的经停站以及价格信息。

（4）用户购票成功之后录入订单信息。

2.数据更新

（1）更新用户的个人信息。

（2）更新列车的基本信息。

（3）更新列车的经停站。

（4）更新列车的价格信息。

（5）删除列车。

（6）更新列车座位的信息。

（7）用户改签之后更新订单信息。

（8）用户退票之后删除订单信息。

（9）管理员对某个用户封号或解封之后更新用户信息。

3.数据查询

（1）查询用户的个人信息。

（2）查询满足条件的车次。

（3）查询某列火车的余票。

（4）查询火车票的价格。

（5）查询自己的订单详情。

（6）管理员查询所有列车的信息。

（7）管理员按用户名或发车时间或下单时间查询用户订单。

**4.2 系统需求说明**

**4.2.1 初始数据库大小**

（1）初始共有20辆列车，每辆列车经过6-8个经停站。

（2）每辆列车有2个车厢，每个车厢有30个座位，6排，每排有5个座位（ABCDF）。

（3）每辆列车的每2个相邻的经停站之间都有1个价格，每辆列车有5-7个站到站的价格。

（4）初始时有1个管理员。

**4.2.2 网络和共享需求**

（1）必须能够连接互联网。

（2）必须能够支持至少 1000 名用户同时访问，需要考虑这么大数量并发访问 的许可需求。

**4.2.3 性能**

高峰期：每年的 1月、2月、7月、9月

（1）单个记录查询时间少于1秒，高峰期少于3秒

（2）多个记录查询时间少于3秒，高峰期少于5秒

（3）更新/保存记录时间少于1秒，高峰期少于5秒

**4.2.4 安全性**

（1）数据库必须有口令保护

（2）每个用户分配特定的用户视图所应有的访问权限

（3）用户只能在适合他们完成工作需要的窗口中看到需要的数据

**4.2.5 法律问题**

对用户信息管理，遵守法律

**五、数据库逻辑设计**

**5.1 E-R图**

火车票售票系统共有7个实体集，分别是用户实体，管理员实体，火车实体，火车经停站实体，火车座位实体，火车票价格实体和订单实体。

**5.1.1 用户实体**

用户名

密码

账号状态

年龄

用户

电话号码

性别

**5.1.2 管理员实体**

管理员

管理员名

密码

姓名

管理员编号

电话号码

性别

年龄

**5.1.3 火车实体**

火车车次

车厢座位数

日期

火车编号

车厢数

始发站

火车类型

火车名字

终点站

**5.1.4 火车经停站实体**

火车经停站

日期

火车编号

城市名

离开时间

停留时间

到达时间

火车站名

**5.1.5 火车座位实体**

火车座次

日期

火车编号

车厢编号

终止占用编号

座位等级

座位类型

排编号

位置编号

起始占用编号

**5.1.6 火车票价格实体**

火车票价格

价格

火车编号

出发站

座位等级

座位类型

到达站

**5.1.7 订单实体**

订单编号

火车编号

排编号

车厢号

出发站

下单时间

用户名

到达站

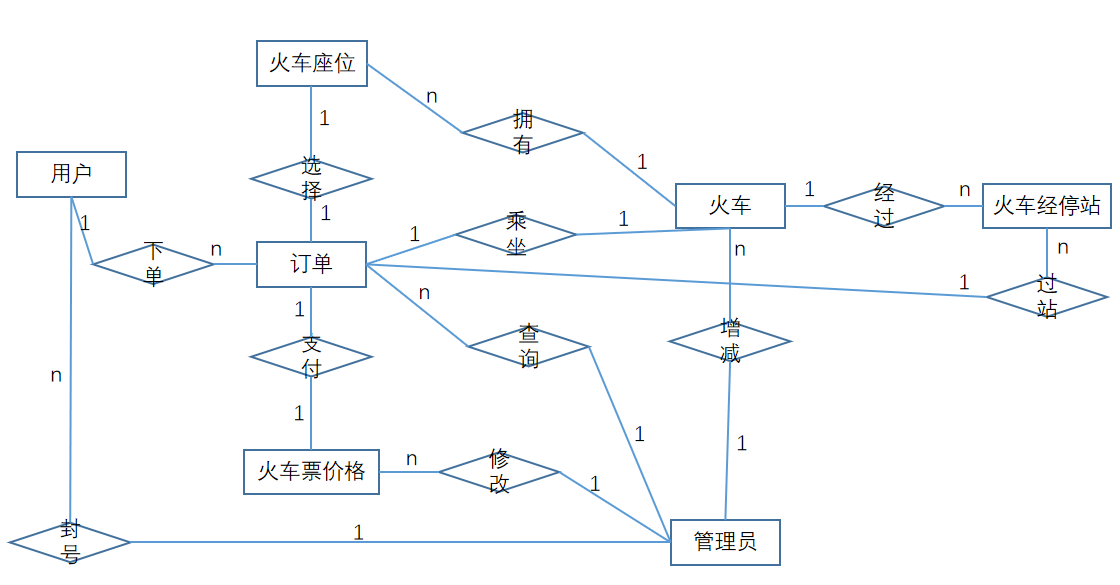
订单

座位等级

座位类型

位置编号

**5.1.8 总E-R图**



**5.2 数据字典**

根据上面描述的 E-R 图进行数据库的逻辑结构设计。在这里以表格的形式给出数据库的设计。

**5.2.1 用户实体**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| username | varchar(20) | 用户名，主键 |
| password | varchar(20) | 密码 |
| name | varchar(10) | 姓名 |
| age | int | 年龄 |
| sex | varchar(1) | 性别 |
| phonenumber | varchar(20) | 电话号码 |
| status | int | 账号状态（是否封号） |

**5.2.2 管理员实体**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| managername | varchar(20) | 管理员名 |
| password | varchar(20) | 密码 |
| name | varchar(10) | 姓名 |
| age | int | 年龄 |
| sex | varchar(1) | 性别 |
| phonenumber | varchar(20) | 电话号码 |
| managerid | varchar(20) | 管理员编号，主键 |

**5.2.3 火车实体**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| date | varchar(20) | 日期 |
| trainid | varchar(20) | 列车编号，主键 |
| trainname | varchar(10) | 列车名 |
| traintype | varchar(5) | 列车类型 |
| startstation | varchar(10) | 始发站 |
| arrivestation | varchar(10) | 终点站 |
| carriageamount | int | 车厢数 |
| seatnumber | int | 每个车厢的座位数 |

**5.2.4 火车经停站实体**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| date | varchar(20) | 日期 |
| trainid | varchar(10) | 列车编号，主键 |
| cityname | varchar(10) | 城市名 |
| stationname | varchar(10) | 火车站名 |
| arrivetime | varchar(20) | 到达时间 |
| remaintime | varchar(10) | 停留时间 |
| leavetime | varchar(20) | 离开时间 |

**5.2.5 火车座位实体**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| date | varchar(20) | 日期 |
| trainid | varchar(10) | 列车编号 |
| carriageid | int | 车厢号 |
| rowid | int | 行编号 |
| position | varchar(1) | 位置 |
| seattype | varchar(5) | 座位类型（高铁，动车，普快） |
| seatrank | varchar(10) | 座位等级（一二等座，硬座等） |
| from\_station | int | 从哪个站开始有人坐 |
| to\_station | int | 到哪个站结束有人坐 |

**5.2.6 火车票价格实体**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| trainid | varchar(20) | 列车编号 |
| startstation | varchar(10) | 出发站 |
| arrivestation | varchar(10) | 到达站 |
| seattype | varchar(5) | 座位类型（高铁，动车，普快） |
| seakrank | varchar(5) | 座位等级（一二等座，硬座等） |
| price | int | 价格 |

**5.2.7 订单实体**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **备注** |
| orderid | varchar(20) | 订单编号，主键 |
| username | varchar(20) | 用户名 |
| date | varchar(20) | 发车日期 |
| trainid | varchar(20) | 列车编号 |
| startstation | varchar(10) | 出发站 |
| arrivestation | varchar(10) | 到达站 |
| carriageid | int | 车厢号 |
| rowid | int | 行编号 |
| position | varchar(1) | 位置 |
| seattype | varchar(5) | 座位类型 |
| seatrank | varchar(5) | 座位等级 |
| price | int | 价格 |
| ordertime | varchar(20) | 下单时间 |

**六、数据库物理设计**

**6.1 索引**

（1）用户表以username为索引

create index username\_index on usermessege(username);

（2）管理员表以managerid为索引

create index managerid\_index on managermessege(managerid);

（3）火车表以trainid为索引

create index trainid\_index on trainmessege(trainid);

（4）火车经停站表以startstation为索引

create index startstation\_index on trainpassstation(startstation);

（5）火车座位表以trainid为索引

create index trainid\_index on trainseat(trainid);

（6）火车票价格表以startstation为索引

create index startstation\_index on trainticketprice(startstation);

（7）订单表以orderid为索引

create index orderid\_index on orderstable(orderid);

**6.2 视图**

此环节设计在数据库应用生命周期的需求分析和收集阶段标识的用户视图。通常， 视图使用 SQL 或类似 QBE 的工具创建。例如，对火车票售票系统的用户：

create view user\_view as

select trainid,trainname,traintype,startstation,arrivestation

from trainmessege

where date= ‘2020-11-23’；

**6.3 安全机制**

**6.3.1 系统安全**

1. 提供了充足的异常处理机制，能够捕获由各种错误引发的异常(如：越权操作、输入 数据类型与数据库要求类型不一致、查询过程中出现的错误等等)。

2、系统登录时要凭用户名和口令进入，输入正确时才可以进入系统。用户需要选择自己的身份（用户、管理员），若用户名和密码输入正确，系统可以打开符合该用户身份的操作界面。其他界面该用户没有权限访问。

**6.3.2 数据安全**

1、根据用户身份，用户进入不同的用户界面，即享有不同的权限，只有管理员才可以修改一些重要信息，以确保数据库不被随意更改，保证数据安全。

2、火车票售票系统共有两种种身份：用户、管理员。他们相应的权限如下：

（1）用户：查看修改个人信息，选择出发城市到达城市和发车时间查询满足条件的列车，查看列车经停站，查看列车余票，查看车票价格，选座购票，查看订单，查看余额，改签，退票。

（2）管理员：查看修改个人信息，车次管理，包括增加列车，删除列车，查询列车信息。订单查询，包括按用户名查询订单，按发车时间查询订单，按下单时间查询订单。用户管理，包括封号，解封。

**七、应用程序设计**

**7.1 功能模块**

火车票售票系统

管理员系统

用户系统

改签退票

用户管理

订单查询

车次管理

查看余额

查看订单

选座购票

余票查询

车次查询

**7.2 界面设计**

**7.2.1 登录界面**

用户打开软件后进入登录界面，选择身份（用户或管理员），然后输入账号密码，如果账号密码正确就跳转到对应的系统。登录界面如下：

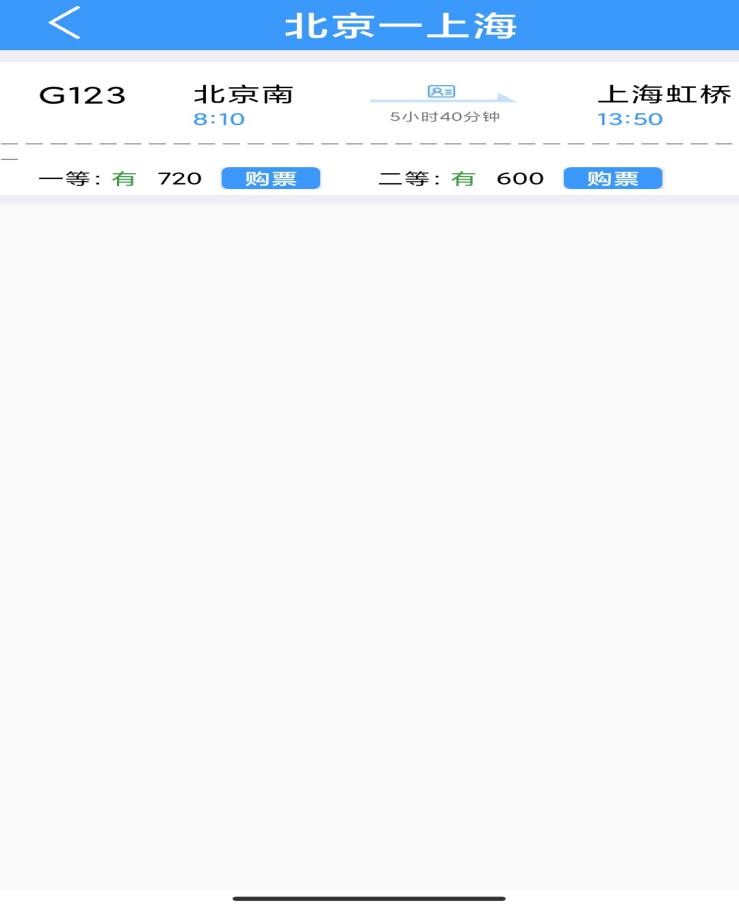
****

**7.2.2 用户界面**

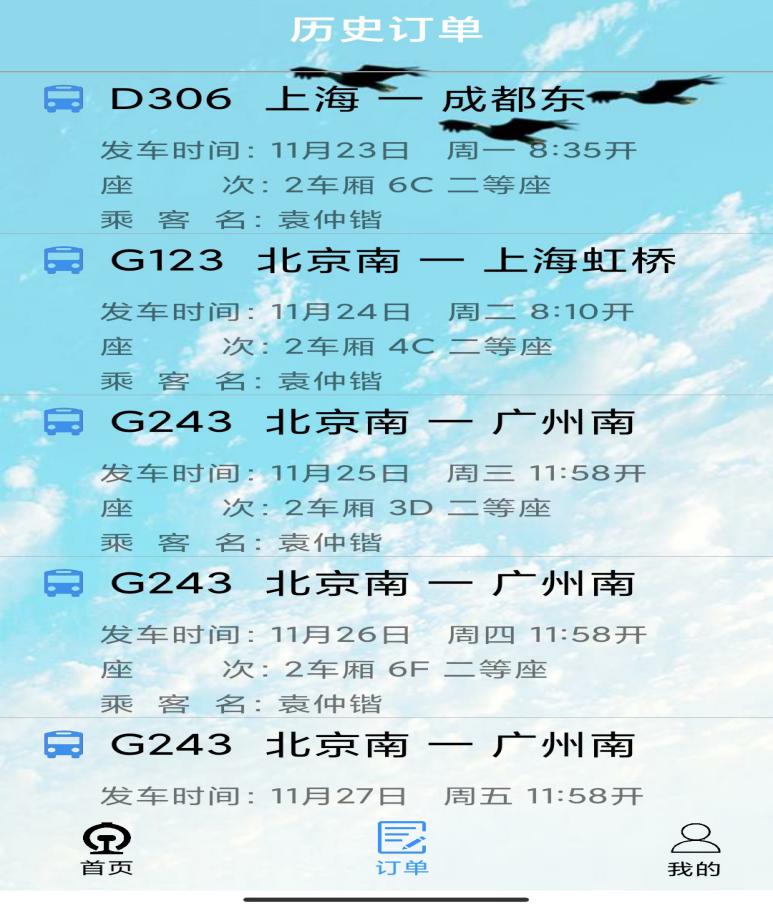
1. 用户首页。用户登录成功之后，进入用户系统，跳转到用户首页，在这个页面可以选择出发城市，到达城市和发车日期，然后按照选择的条件进行查询。首页如下：



（2）查询页面。用户设置好条件之后点击查询按钮，进入查询页面，在查询页面可以看到满足条件的所有列车的信息以及余票数和价格数，点击购票即可买票，查询界面如下：



（3）订单页面。购票成功后系统会生成订单，用户可以点击订单按钮进入订单界面，订单界面会显示当前用户的所有订单。订单界面如下：

****

（4）订单详情页面。用户查询到自己所有的订单之后，点击某一个订单，就会进入到该订单的详情页面，订单详情页面会显示该订单的所有信息，可以展示二维码，并具有改签退票按钮，点击按钮后可以执行相应的操作，订单详情页面如下：



1. “我的”界面。用户点击“我的”按钮之后会进入“我的”界面，界面上会显示用户名和余额，同时还有一个退出登录按钮，点击按钮之后会退出登录。“我的”界面如下：



**7.2.3管理员界面**

1. 车次管理界面。管理员登陆成功之后进入管理员系统，跳转到车次管理界面，车次管理界面有三个功能，增加列车，删除列车，查看列车信息，点击对应的按钮可以实现对应的操作，车次管理界面如下：



（2）订单查询界面。管理员点击订单查询按钮可以进入订单查询界面，订单查询有三种模式，按用户名查询，按发车日期查询和按下单日期查询，点击对应的按钮可以执行相应的查询。订单查询界面如下：

（3）用户管理界面。管理员点击用户管理按钮可进入用户管理界面，用户管理界面有两个功能，封号和解封，点击对应的按钮可以执行相应的查询。用户管理界面如下：



**7.3 事务设计**

**7.3.1 注册新用户**

register\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 //获取用户输入  
 username=username\_field.getText().toString();  
 password=password\_field.getText().toString();  
 confirmpassword=confirmpassword\_field.getText().toString();  
  
  
 //判断用户两次输入的密码是否一致  
 if(password.equals(confirmpassword)==true){  
  
 //两次密码输入一致，将注册信息写入数据库中  
  
 //打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
  
 //向数据库中插入数据  
 String sql="insert into usermessege values('"+username+"','"+password+"',1,10000)";  
 db.execSQL(sql);  
  
 AlertDialog.Builder adb=new AlertDialog.Builder(RegisterActivity.this);  
 adb.setTitle("提示");  
 adb.setMessage("注册成功!");  
 adb.show();  
  
 //跳转到登录界面并关闭原来的界面  
 Intent intent=new Intent(RegisterActivity.this,MainActivity.class);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK );  
 startActivity(intent);  
  
 }else{  
  
 //两次密码不一致，提醒用户并清空密码框  
 AlertDialog.Builder adb=new AlertDialog.Builder(RegisterActivity.this);  
 adb.setTitle("提示");  
 adb.setMessage("您两次输入的密码不一致，请重新输入!");  
 adb.show();  
  
 password\_field.setText("");  
 confirmpassword\_field.setText("");  
 }  
  
 }  
});

**7.3.2 登录**

login\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 //获取用户输入  
 username=username\_field.getText().toString();  
 password=password\_field.getText().toString();  
 //判断是用户登录还是管理员登录  
 if(login\_mode.equals("USER")){  
 //用户登录  
 //验证账号密码是否正确  
 String sql="select \* from usermessege where username='"+username+"'";  
 result=db.rawQuery(sql,null);  
 if(result.getCount()==0){  
 //用户不存在  
 Tips("用户不存在!");  
 username\_field.setText("");  
 password\_field.setText("");  
 }  
 else{  
 //用户存在，检查是否被封号  
 result.moveToFirst();  
 int status=result.getInt(2);  
 if(status==0){  
 //已被封号  
 Tips("您的账号已被封号！");  
 }  
 else{  
 //未被封号，检查密码是否正确  
 result.moveToFirst();  
 real\_password=result.getString(1);  
  
 if(password.equals(real\_password)){  
 //密码正确  
 //判断是否勾选记住密码  
 CheckOrNot();  
 //密码正确,跳转到用户首页并传递用户名  
 Intent intent=new Intent(MainActivity.this,HomeActivity.class);  
 Bundle bundle=new Bundle();  
 bundle.putString("username",username);  
 intent.putExtras(bundle);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK );  
 startActivity(intent);  
 }  
 else{  
 //密码错误  
 Tips("密码错误!");  
 password\_field.setText("");  
 }  
 }  
 }  
 }  
 else{  
 //管理员登录  
 //验证账号密码是否正确  
 String sql="select \* from managermessege where username='"+username+"'";  
 result=db.rawQuery(sql,null);  
  
 if(result.getCount()==0){  
 //用户不存在  
 Tips("用户不存在!");  
  
 username\_field.setText("");  
 password\_field.setText("");  
 }else{  
  
 //用户存在，检查密码是否正确  
 result.moveToFirst();  
 real\_password=result.getString(1);  
  
 if(password.equals(real\_password)){  
 //密码正确  
 //判断是否勾选记住密码  
 CheckOrNot();  
 //密码正确,跳转到管理员界面并传递管理员名  
 Intent intent=new Intent(MainActivity.this,TrainManageActivity.class);  
  
 Bundle bundle=new Bundle();  
 bundle.putString("username",username);  
 intent.putExtras(bundle);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK );  
 startActivity(intent);  
 }else{  
 //密码错误  
 Tips("密码错误!");  
 password\_field.setText("");  
 }  
 }  
 }  
 }  
});

**7.3.3 查询列车**

//打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
  
 //查询符合要求的车次  
 String sql="select distinct trainid from trainpassstation as A where cityname='"+start\_city\_name+"' and exists(select trainid from trainpassstation as B where B.trainid=A.trainid and B.cityname='"+arrive\_city\_name+"' and B.stationid>A.stationid)";  
 Cursor c=db.rawQuery(sql,null);  
  
 //判断查询结果是否为空  
 if(c.getCount()==0){  
 //没有满足条件的车次  
 Toast.makeText(QueryActivity.this,"未查询到满足条件的车次!",Toast.LENGTH\_SHORT).show();  
 }  
 else{  
 //有满足条件的车次  
 c.moveToFirst();  
 while(c.isAfterLast()==false){  
 String trainid=c.getString(0);  
 //查询发车信息与到达信息  
 //查询发车信息  
 String StartSQL="select \* from trainpassstation where trainid='"+trainid+"' and cityname='"+start\_city\_name+"'";  
 Cursor c2=db.rawQuery(StartSQL,null);  
 c2.moveToFirst();  
 String start\_station\_name=c2.getString(3);  
 int start\_station\_id=c2.getInt(4);  
 String leavetime=c2.getString(7);  
 //查询到达信息  
 String ArriveSQL="select \* from trainpassstation where trainid='"+trainid+"' and cityname='"+arrive\_city\_name+"'";  
 Cursor c3=db.rawQuery(ArriveSQL,null);  
 c3.moveToFirst();  
 String arrive\_station\_name=c3.getString(3);  
 int arrive\_station\_id=c3.getInt(4);  
 String arrivetime=c3.getString(5);  
  
 //查询余票数  
 //查询一等票余票数  
 String FirstSeatSql="select \* from trainseat where trainid='"+trainid+"' and seattype='一等座' and (tostation<="+start\_station\_id+" or fromstation>="+arrive\_station\_id+")";  
 Cursor c4=db.rawQuery(FirstSeatSql,null);  
 int FirstSeatNumber=c4.getCount();  
  
 //查询二等票余票数  
 String SecondSeatSql="select \* from trainseat where trainid='"+trainid+"' and seattype='二等座' and (tostation<="+start\_station\_id+" or fromstation>="+arrive\_station\_id+")";  
 Cursor c5=db.rawQuery(SecondSeatSql,null);  
 int SecondSeatNumber=c5.getCount();  
  
 //查询票价  
 //查询一等座的票价  
 String FirstPriceSql="select \* from trainticketprice where trainid='"+trainid+"' and seattype='一等座' and fromstationid>="+start\_station\_id+" and tostationid<="+arrive\_station\_id;  
 Cursor c6=db.rawQuery(FirstPriceSql,null);  
 c6.moveToFirst();  
 int FirstSeatPrice=0;  
 while(c6.isAfterLast()==false){  
 FirstSeatPrice+=c6.getInt(6);  
 c6.moveToNext();  
 }  
 c6.close();  
  
 //查询二等座的票价  
 String SecondPriceSql="select \* from trainticketprice where trainid='"+trainid+"' and seattype='二等座' and fromstationid>="+start\_station\_id+" and tostationid<="+arrive\_station\_id;  
 Cursor c7=db.rawQuery(SecondPriceSql,null);  
 c7.moveToFirst();  
 int SecondSeatPrice=0;  
 while(c7.isAfterLast()==false){  
 SecondSeatPrice+=c7.getInt(6);  
 c7.moveToNext();  
 }  
 c7.close();  
  
 String messege="满足条件的车次为:"+trainid+"发车时间为:"+leavetime+"到达时间为:"+arrivetime+"一等座余票为:"+FirstSeatNumber+"一等座票价为:"+FirstSeatPrice;  
 Toast.makeText(QueryActivity.this,messege,Toast.LENGTH\_SHORT).show();  
  
 //将信息显示在查询界面  
 TrainMessegeView tmv=new TrainMessegeView(this,null,time,trainid,start\_station\_name,start\_station\_id,arrive\_station\_name,arrive\_station\_id,leavetime,arrivetime,FirstSeatNumber,SecondSeatNumber,FirstSeatPrice,SecondSeatPrice,username,db);  
 query\_page.addView(tmv);  
  
 //添加分隔线  
 TextView divide\_line=new TextView(QueryActivity.this);  
 divide\_line.setHeight(20); divide\_line.setBackgroundColor(getResources().getColor(R.color.graywhite));  
 query\_page.addView(divide\_line);  
 c.moveToNext();  
 }  
 c.close();  
 }

**7.3.4 选座购票**

first\_seat\_buy\_button.setOnClickListener(new OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 //首先判断是否有余票  
 if(FirstSeatNumber<=0){  
 //没有余票  
 }else{  
 //有余票  
  
 //弹出选择框让用户选择座位  
  
 //建立选择器  
 final RadioGroup seat\_position=new RadioGroup(getContext());  
 seat\_position.setOrientation(HORIZONTAL);  
  
 RadioButton AButton=new RadioButton(getContext());  
 AButton.setText("A");  
 AButton.setId(R.id.AButton);  
 RadioButton BButton=new RadioButton(getContext());  
 BButton.setId(R.id.BButton);  
 BButton.setText("B");  
 RadioButton CButton=new RadioButton(getContext());  
 CButton.setId(R.id.CButton);  
 CButton.setText("C");  
 RadioButton DButton=new RadioButton(getContext());  
 DButton.setId(R.id.DButton);  
 DButton.setText("D");  
 RadioButton FButton=new RadioButton(getContext());  
 FButton.setId(R.id.FButton);  
 FButton.setText("F");  
  
  
 seat\_position.addView(AButton);  
 seat\_position.addView(BButton);  
 seat\_position.addView(CButton);  
 seat\_position.addView(DButton);  
 seat\_position.addView(FButton);  
  
  
 AlertDialog.Builder adb=new AlertDialog.Builder(getContext());  
 adb.setTitle("请选择座位的位置");  
 adb.setView(seat\_position);  
 adb.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 //点击确定后开始购票  
  
  
 //获取用户选择的位置  
 String position\_detail=getResources().getResourceName(seat\_position.getCheckedRadioButtonId());  
 String position=position\_detail.substring(position\_detail.length()-7,position\_detail.length()-6);  
  
 //查询用户选择的位置是否还有余票  
 String FirstSeatSql0="select \* from trainseat where trainid='"+trainid+"' and seattype='一等座' and position='"+position+"' and (tostation<="+start\_station\_id+" or fromstation>="+arrive\_station\_id+")";  
 Cursor result0=db.rawQuery(FirstSeatSql0,null);  
  
 if(result0.getCount()>0){  
  
 //所选类型的座位还有余票,在该类型的座位中随机选一个  
  
  
 //首先在可以购票的座次之间随机选择一个座次,产生一个1-FirstSeatNumber的随机数,代表选择第几个座位  
 Random random=new Random();  
 int rn=random.nextInt(result0.getCount())+1;  
  
 String FirstSeatSql="select \* from trainseat where trainid='"+trainid+"' and seattype='一等座' and position='"+position+"' and (tostation<="+start\_station\_id+" or fromstation>="+arrive\_station\_id+")";  
 Cursor fc=db.rawQuery(FirstSeatSql,null);  
 fc.moveToFirst();  
  
 int current=1;  
 while(current<rn){  
 fc.moveToNext();  
 current++;  
 }  
  
 //获取选中座位的信息  
 int rowid=fc.getInt(3);  
 int from=fc.getInt(6);  
 int to=fc.getInt(7);  
  
 //修改该座位的信息  
 int new\_from=Math.min(from,start\_station\_id);  
 int new\_to=Math.max(to,arrive\_station\_id);  
 String updateSql1="update trainseat set fromstation="+new\_from+" where trainid='"+trainid+"' and seattype='一等座' and rowid="+rowid+" and position='"+position+"'";  
 String updateSql2="update trainseat set tostation="+new\_to+" where trainid='"+trainid+"' and seattype='一等座' and rowid="+rowid+" and position='"+position+"'";  
 db.execSQL(updateSql1);  
 db.execSQL(updateSql2);  
  
 //添加订单  
 SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");//设置日期格式  
 String ordertime=df.format(new Date());// 获取当前系统时间(下单时间)  
  
 String orderSql="insert into orderstable values('"+time+"','"+username+"','"+trainid+"','"+start\_station\_name+"','"+leavetime+"','"+arrive\_station\_name+"','"+arrivetime+"',1,"+rowid+",'"+position+"','一等座',"+FirstSeatPrice+",'"+ordertime+"')";  
 db.execSQL(orderSql);  
  
 //扣除用户余额  
  
 //获取用户当前余额  
 String money\_sql="select \* from usermessege where username='"+username+"'";  
 Cursor money\_result=db.rawQuery(money\_sql,null);  
 money\_result.moveToFirst();  
 int old\_money=money\_result.getInt(3);  
  
 //更改余额  
 int new\_money=old\_money-FirstSeatPrice;  
 String new\_money\_sql="update usermessege set money="+new\_money+" where username='"+username+"'";  
 db.execSQL(new\_money\_sql);  
  
 //提示用户购买成功  
 AlertDialog.Builder adb2=new AlertDialog.Builder(getContext());  
 adb2.setTitle("购买成功提醒");  
 adb2.setMessage("恭喜你成功购买了"+time+"从"+start\_station\_name+"开往"+arrive\_station\_name+"的"+trainid+"次列车一等座,您的座位为1车厢"+rowid+position);  
 adb2.show();  
  
 }  
 else{  
  
 //所选座位无余票,在所有座位中随机选一个  
  
 //首先在可以购票的座次之间随机选择一个座次,产生一个1-FirstSeatNumber的随机数,代表选择第几个座位  
 Random random=new Random();  
 int rn=random.nextInt(FirstSeatNumber)+1;  
  
 String FirstSeatSql="select \* from trainseat where trainid='"+trainid+"' and seattype='一等座' and (tostation<="+start\_station\_id+" or fromstation>="+arrive\_station\_id+")";  
 Cursor fc=db.rawQuery(FirstSeatSql,null);  
 fc.moveToFirst();  
  
 int current=1;  
 while(current<rn){  
 fc.moveToNext();  
 current++;  
 }  
  
 //获取选中座位的信息  
 int rowid=fc.getInt(3);  
 String position2=fc.getString(4);  
 int from=fc.getInt(6);  
 int to=fc.getInt(7);  
  
 //修改该座位的信息  
 int new\_from=Math.min(from,start\_station\_id);  
 int new\_to=Math.max(to,arrive\_station\_id);  
 String updateSql1="update trainseat set fromstation="+new\_from+" where trainid='"+trainid+"' and seattype='一等座' and rowid="+rowid+" and position='"+position2+"'";  
 String updateSql2="update trainseat set tostation="+new\_to+" where trainid='"+trainid+"' and seattype='一等座' and rowid="+rowid+" and position='"+position2+"'";  
 db.execSQL(updateSql1);  
 db.execSQL(updateSql2);  
  
 //添加订单  
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");//设置日期格式  
String ordertime=df.format(new Date());// 获取当前系统时间(下单时间)  
  
String orderSql="insert into orderstable values('"+time+"','"+username+"','"+trainid+"','"+start\_station\_name+"','"+leavetime+"','"+arrive\_station\_name+"','"+arrivetime+"',1,"+rowid+",'"+position2+"','一等座',"+FirstSeatPrice+",'"+ordertime+"')";  
 db.execSQL(orderSql);  
  
 //扣除用户余额  
 //获取用户当前余额  
 String money\_sql="select \* from usermessege where username='"+username+"'";  
 Cursor money\_result=db.rawQuery(money\_sql,null);  
 money\_result.moveToFirst();  
 int old\_money=money\_result.getInt(3);  
 //更改余额  
 int new\_money=old\_money-FirstSeatPrice;  
 String new\_money\_sql="update usermessege set money="+new\_money+" where username='"+username+"'";  
 db.execSQL(new\_money\_sql);  
  
 //提示用户购买成功  
 AlertDialog.Builder adb2=new AlertDialog.Builder(getContext());  
 adb2.setTitle("购买成功提醒");  
 adb2.setMessage("恭喜你成功购买了"+time+"从"+start\_station\_name+"开往"+arrive\_station\_name+"的"+trainid+"次列车一等座,您的座位为1车厢"+rowid+position2);  
 adb2.show();  
 }  
 }  
 });  
 adb.setNegativeButton("取消",null);  
 adb.show();  
 }  
 }  
});

**7.3.5 用户查看订单**

//打开数据库  
db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
  
//查询当前用户的所有订单  
String order\_sql="select \* from orderstable where username='"+username+"'";  
result=db.rawQuery(order\_sql,null);  
  
if(result.getCount()==0){  
 //无订单  
 Toast.makeText(OrderActivity.this,"未查询到订单！",Toast.LENGTH\_SHORT).show();  
}else{  
 //有订单  
 result.moveToFirst();  
 while(result.isAfterLast()==false){  
  
 //读取当前订单项的内容  
 time=result.getString(0);  
 trainid=result.getString(2);  
 startstationname=result.getString(3);  
 leavetime=result.getString(4);  
 arrivestationname=result.getString(5);  
 arrivetime=result.getString(6);  
 carriageid=result.getInt(7);  
 rowid=result.getInt(8);  
 position=result.getString(9);  
 seattype=result.getString(10);  
 price=result.getInt(11);  
 ordertime=result.getString(12);  
 //新建订单项并添加到界面  
 OrderOptionActivity ooa=new OrderOptionActivity(this,null,username,time,trainid,startstationname,leavetime,arrivestationname,arrivetime,carriageid,rowid,position,seattype,price,ordertime,OrderActivity.this);  
 ooa.setTop(15);  
  
 //设置点击事件监听器  
 try {  
 ooa.setOnClickListener(new View.OnClickListener() {  
  
 String time2=time;  
 String trainid2=trainid;  
 String startstationname2=startstationname;  
 String leavetime2=leavetime;  
 String arrivestationname2=arrivestationname;  
 String arrivetime2=arrivetime;  
 int carriageid2=carriageid;  
 int rowid2=rowid;  
 String position2=position;  
 String seattype2=seattype;  
 int price2=price;  
 String ordertime2=ordertime;  
 @Override  
 public void onClick(View v) {  
 //点击后进入详情界面并传递数据  
 Intent intent=new Intent(OrderActivity.this,OrderDetailActivity.class);  
  
 Bundle bundle=new Bundle();  
 bundle.putString("order\_time",ordertime2);  
 bundle.putString("start\_station\_name",startstationname2);  
 bundle.putString("leave\_time",leavetime2);  
 bundle.putString("trainid",trainid2); bundle.putString("total\_time",calculateTotalTime(leavetime,arrivetime2)) bundle.putString("arrive\_station\_name",arrivestationname2);  
 bundle.putString("arrive\_time",arrivetime2);  
 bundle.putString("leave\_date",time2);  
 bundle.putString("passenger\_name",username);  
 bundle.putInt("price",price2);  
 bundle.putString("seat\_type",seattype2);  
 bundle.putInt("carriage\_id",carriageid2);  
 bundle.putInt("row\_id",rowid2);  
 bundle.putString("position",position2);  
 intent.putExtras(bundle);  
 startActivity(intent);  
 }  
 });  
 }catch (Exception e){  
 AlertDialog.Builder adb=new AlertDialog.Builder(OrderActivity.this);  
 adb.setMessage(e.getMessage());  
 adb.show();  
 }  
 jiemian.addView(ooa);  
  
 //添加分隔线  
 TextView divide\_line=new TextView(OrderActivity.this);  
 divide\_line.setHeight(1);  
 divide\_line.setTop(15); divide\_line.setBackgroundColor(getResources().getColor(R.color.gray));  
 jiemian.addView(divide\_line);  
 result.moveToNext();  
 }  
}

**7.3.6 改签**

change\_ticket\_view.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 //弹出提示框  
 AlertDialog.Builder adb=new AlertDialog.Builder(OrderDetailActivity.this);  
 adb.setTitle("提示");  
 adb.setMessage("您确定要改签吗?");  
 adb.setNegativeButton("取消",null);  
 adb.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 //首先将该订单从订单表中删除  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);//打开数据库  
 String deleteSql="delete from orderstable where username='"+passenger\_name+"' and ordertime='"+order\_time+"'";  
 db.execSQL(deleteSql);  
  
 //恢复用户余额  
  
 //获取用户当前余额  
 String money\_sql="select \* from usermessege where username='"+passenger\_name+"'";  
 Cursor money\_result=db.rawQuery(money\_sql,null);  
 money\_result.moveToFirst();  
 int old\_money=money\_result.getInt(3);  
  
 //更改余额  
 int new\_money=old\_money+price;  
 String new\_money\_sql="update usermessege set money="+new\_money+" where username='"+passenger\_name+"'";  
 db.execSQL(new\_money\_sql);  
  
 //然后跳转到查询界面并传递相应的数据  
 Intent intent=new Intent(OrderDetailActivity.this,QueryActivity.class);  
 Bundle bundle=new Bundle();  
 bundle.putString("username",passenger\_name);  
 bundle.putString("start\_city\_name",start\_station\_name.substring(0,2)); bundle.putString("arrive\_city\_name",arrive\_station\_name.substring(0,2));  
 bundle.putString("time",leave\_date);  
 bundle.putBoolean("only",true);  
 intent.putExtras(bundle);  
 startActivity(intent);  
 }  
 });  
  
 adb.show();  
 }  
});

**7.3.7 退票**

refund\_ticket\_view.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 //弹出提示框  
 AlertDialog.Builder adb=new AlertDialog.Builder(OrderDetailActivity.this);  
 adb.setTitle("提示");  
 adb.setMessage("您确定要退票吗?");  
 adb.setNegativeButton("取消",null);  
 adb.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 //将该订单从订单表中删除  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);//打开数据库  
 String deleteSql="delete from orderstable where username='"+passenger\_name+"' and ordertime='"+order\_time+"'";  
 db.execSQL(deleteSql);  
 //恢复用户余额  
 //获取用户当前余额  
 String money\_sql="select \* from usermessege where username='"+passenger\_name+"'";  
 Cursor money\_result=db.rawQuery(money\_sql,null);  
 money\_result.moveToFirst();  
 int old\_money=money\_result.getInt(3);  
 //更改余额(距离发车多于两天全额退款，不足两天扣除10%)  
  
 try {  
 //获取当前时间  
 SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");//设置日期格式  
 String current\_time = df.format(new Date());// new Date()为获取当前系统时间  
 String s[] = current\_time.split("-");  
 int current\_month = Integer.parseInt(s[1]);  
 int current\_day = Integer.parseInt(s[2]);  
 //获取发车时间  
 String l\_date = leave\_date.substring(0, leave\_date.length() - 6);  
 String s2[] = l\_date.split("月");  
 int leave\_month = Integer.parseInt(s2[0]);  
 int leave\_day = Integer.parseInt(s2[1]);  
 int new\_money;  
 if ((leave\_month \* 30 + leave\_day) - (current\_month \* 30 + current\_day) >= 2) {  
 new\_money = old\_money + price;  
 } else {  
 new\_money = old\_money + price \* 9 / 10;  
 }  
 String new\_money\_sql = "update usermessege set money=" + new\_money + " where username='" + passenger\_name + "'";  
 db.execSQL(new\_money\_sql);  
  
 }catch (Exception e){ Toast.makeText(OrderDetailActivity.this,e.getMessage(),Toast.LENGTH\_SHORT).show();  
 }  
 //提示用户订单已删除  
 Toast.makeText(OrderDetailActivity.this,"订单已删除!",Toast.LENGTH\_SHORT);  
 //跳转到订单界面并传递用户名  
 Intent intent=new Intent(OrderDetailActivity.this,OrderActivity.class);  
 Bundle bundle=new Bundle();  
 bundle.putString("username",passenger\_name);  
 intent.putExtras(bundle);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK );  
 startActivity(intent);  
 }  
 });  
 adb.show();  
 }  
});

**7.3.8 查看余额**

//获取用户当前余额  
db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
String money\_sql="select \* from usermessege where username='"+username+"'";  
Cursor money\_result=db.rawQuery(money\_sql,null);  
money\_result.moveToFirst();  
int current\_money=money\_result.getInt(3);  
money\_view.setText("¥"+current\_money);

**7.3.9 增加、删除、查看列车**

exec\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 //点击执行按钮后执行增加或删除列车的操作  
 String train\_id=train\_id\_view.getText().toString();  
  
 if(operate\_type.getCheckedRadioButtonId()==R.id.add\_train){  
 //增加列车  
 //清空输入  
 operate\_type.clearCheck();  
 train\_id\_view.setText("");  
 train\_name\_view.setText("");  
 train\_type.clearCheck();  
 leave\_date\_view.setText("");  
 pass\_station\_view.removeAllViews();  
 //提示  
 AlertDialog.Builder adb=new AlertDialog.Builder(TrainManageActivity.this);  
 adb.setTitle("提示");  
 adb.setMessage("您已成功添加"+train\_id+"次列车");  
 adb.show();  
  
 }else if(operate\_type.getCheckedRadioButtonId()==R.id.see\_train){  
  
 //查看列车信息  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
  
 String see\_sql="select \* from trainmessege";  
 Cursor result=db.rawQuery(see\_sql,null);  
 result.moveToFirst();  
 int count=666;  
 while(result.isAfterLast()==false){  
  
 //读取信息  
 String t\_id=result.getString(1);  
 String t\_name=result.getString(2);  
 String t\_type=result.getString(3);  
 String startstation=result.getString(4);  
 String arrivestation=result.getString(5);  
  
 TextView tv=new TextView(TrainManageActivity.this);  
 tv.setText(t\_id+" "+t\_name+" "+t\_type+" "+startstation+" "+arrivestation);  
 tv.setTextSize(20);  
 pass\_station\_view.addView(tv);  
 TextView tv2=new TextView(TrainManageActivity.this);  
 tv2.setText("G"+count+" "+t\_name+" "+t\_type+" "+arrivestation+" "+startstation);  
 tv2.setTextSize(20);  
 pass\_station\_view.addView(tv2);  
 count++;  
 result.moveToNext();  
 }  
 }else{  
 //删除列车  
 //打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
 //删除该列车的所有信息(包括车次表，经停站表，座位表，价格表)  
 String delete\_sql\_1="delete from trainmessege where trainid='"+train\_id+"'";  
 String delete\_sql\_2="delete from trainpassstation where trainid='"+train\_id+"'";  
 String delete\_sql\_3="delete from trainseat where trainid='"+train\_id+"'";  
 String delete\_sql\_4="delete from trainticketprice where trainid='"+train\_id+"'";  
 db.execSQL(delete\_sql\_1);  
 db.execSQL(delete\_sql\_2);  
 db.execSQL(delete\_sql\_3);  
 db.execSQL(delete\_sql\_4);  
 //清空输入  
 operate\_type.clearCheck();  
 train\_id\_view.setText("");  
 //提示管理员删除已完成  
 AlertDialog.Builder adb=new AlertDialog.Builder(TrainManageActivity.this);  
 adb.setTitle("提示");  
 adb.setMessage("您已成功删除"+train\_id+"次列车");  
 adb.show();  
 }  
 }  
});

**7.3.10 订单查询**

query\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 //点击查询按钮之后开始查询  
 //首先清空订单列表  
 order\_view.removeAllViews();  
  
 //判断查询方式  
 if(query\_method.getCheckedRadioButtonId()==R.id.query\_user){  
 //按用户名查询  
  
 //获取用户名  
 String un=query\_input\_view.getText().toString();  
  
 //打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
//查询该用户的所有订单  
 String order\_sql="select \* from orderstable where username='"+un+"'";  
 Cursor result=db.rawQuery(order\_sql,null);  
 if(result.getCount()==0){  
 //无订单  
 Toast.makeText(OrderQueryActivity.this,"未查询到订单！",Toast.LENGTH\_SHORT).show();  
 }  
 else{  
 //有订单  
 result.moveToFirst();  
 while(result.isAfterLast()==false){  
  
 //读取当前订单项的内容  
 String time=result.getString(0);  
 String trainid=result.getString(2);  
 String startstationname=result.getString(3);  
 String leavetime=result.getString(4);  
 String arrivestationname=result.getString(5);  
 String arrivetime=result.getString(6);  
 int carriageid=result.getInt(7);  
 int rowid=result.getInt(8);  
 String position=result.getString(9);  
 String seattype=result.getString(10);  
 int price=result.getInt(11);  
 String ordertime=result.getString(12);  
  
 //新建订单项并添加到界面  
 OrderOptionActivity ooa=new OrderOptionActivity(OrderQueryActivity.this,null,un,time,trainid,startstationname,leavetime,arrivestationname,arrivetime,carriageid,rowid,position,seattype,price,ordertime,OrderQueryActivity.this);  
 ooa.setTop(15);  
 order\_view.addView(ooa);  
 //添加分隔线  
 TextView divide\_line=new TextView(OrderQueryActivity.this);  
 divide\_line.setHeight(1);  
 divide\_line.setTop(15); divide\_line.setBackgroundColor(getResources().getColor(R.color.gray));  
 order\_view.addView(divide\_line);  
 result.moveToNext();  
  
 }  
 }  
 }  
 else if(query\_method.getCheckedRadioButtonId()==R.id.order\_date){  
 //按下单日期进行查询  
 String od=query\_input\_view.getText().toString();  
  
 //打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
  
 String order\_sql;  
 //查询该日期的所有订单  
 if(od.equals("11月23日 周一")){  
 order\_sql="select \* from orderstable";  
 }else{  
 order\_sql="select \* from orderstable where username='145616156'";  
 }  
  
 Cursor result=db.rawQuery(order\_sql,null);  
  
 if(result.getCount()==0){  
 //无订单  
 Toast.makeText(OrderQueryActivity.this,"未查询到订单！",Toast.LENGTH\_SHORT).show();  
 }  
 else{  
 //有订单  
 result.moveToFirst();  
 while(result.isAfterLast()==false){  
 //读取当前订单项的内容  
 String time=result.getString(0);  
 String un=result.getString(1);  
 String trainid=result.getString(2);  
 String startstationname=result.getString(3);  
 String leavetime=result.getString(4);  
 String arrivestationname=result.getString(5);  
 String arrivetime=result.getString(6);  
 int carriageid=result.getInt(7);  
 int rowid=result.getInt(8);  
 String position=result.getString(9);  
 String seattype=result.getString(10);  
 int price=result.getInt(11);  
 String ordertime=result.getString(12);  
  
 //新建订单项并添加到界面  
 OrderOptionActivity ooa=new OrderOptionActivity(OrderQueryActivity.this,null,un,time,trainid,startstationname,leavetime,arrivestationname,arrivetime,carriageid,rowid,position,seattype,price,od,OrderQueryActivity.this);  
 ooa.setTop(15);  
 order\_view.addView(ooa);  
  
 //添加分隔线  
 TextView divide\_line=new TextView(OrderQueryActivity.this);  
 divide\_line.setHeight(1);  
 divide\_line.setTop(15); divide\_line.setBackgroundColor(getResources().getColor(R.color.gray));  
 order\_view.addView(divide\_line);  
 result.moveToNext();  
 }  
 }  
  
 }else{  
 //按发车日期查询  
 String ld=query\_input\_view.getText().toString();  
 //打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
 //查询该日期的所有订单  
String order\_sql="select \* from orderstable where time='"+ld+"'";  
 Cursor result=db.rawQuery(order\_sql,null);  
  
 if(result.getCount()==0){  
 //无订单  
 Toast.makeText(OrderQueryActivity.this,"未查询到订单！",Toast.LENGTH\_SHORT).show();  
 }  
 else{  
 //有订单  
 result.moveToFirst();  
 while(result.isAfterLast()==false){  
  
 //读取当前订单项的内容  
 String un=result.getString(1);  
 String trainid=result.getString(2);  
 String startstationname=result.getString(3);  
 String leavetime=result.getString(4);  
 String arrivestationname=result.getString(5);  
 String arrivetime=result.getString(6);  
 int carriageid=result.getInt(7);  
 int rowid=result.getInt(8);  
 String position=result.getString(9);  
 String seattype=result.getString(10);  
 int price=result.getInt(11);  
 String ordertime=result.getString(12);  
  
 //新建订单项并添加到界面  
 OrderOptionActivity ooa=new OrderOptionActivity(OrderQueryActivity.this,null,un,ld,trainid,startstationname,leavetime,arrivestationname,arrivetime,carriageid,rowid,position,seattype,price,ordertime,OrderQueryActivity.this);  
 ooa.setTop(15);  
 order\_view.addView(ooa);  
  
 //添加分隔线  
 TextView divide\_line=new TextView(OrderQueryActivity.this);  
 divide\_line.setHeight(1);  
 divide\_line.setTop(15);  
 divide\_line.setBackgroundColor(getResources().getColor(R.color.gray));  
 order\_view.addView(divide\_line);  
 result.moveToNext();  
 }  
 }  
 }  
 }  
});

**7.3.11 封号解封**

exec\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 //点击执行按钮后完成对应操作  
 //获取用户名  
 String un=operate\_username\_view.getText().toString();  
 //判断操作类型  
 if(operate\_user\_type.getCheckedRadioButtonId()==R.id.seal){  
 //封号  
 //打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
 //执行封号操作  
String seal\_sql="update usermessege set status=0 where username='"+un+"'";  
 db.execSQL(seal\_sql);  
 Toast.makeText(UserManageActivity.this,"封号已完成",Toast.LENGTH\_SHORT).show();  
 }  
 else{  
 //解封  
 //打开数据库  
 db=openOrCreateDatabase("selltrainticket",MODE\_PRIVATE,null);  
 //执行解封操作  
 String unseal\_sql="update usermessege set status=1 where username='"+un+"'";  
 db.execSQL(unseal\_sql);  
 Toast.makeText(UserManageActivity.this,"解封已完成",Toast.LENGTH\_SHORT).show();  
 }  
 }  
});

**八、测试和运行**

**8.1 登录**

1. 用户名不存在会出现提示如下：



1. 账号被封号会出现提示如下：



1. 密码错误会出现如下提示：



1. 用户名存在，未被封号且密码正确会进入到用户系统，跳转到用户首页，如下：



**8.2 用户系统**

1. 选择城市，点击城市名称弹出城市选择框，可以选择出发城市和到达城市：



1. 选择发车日期，点击日期后会弹出日期选择框，可以选择发车日期：



1. 设置好出发城市到达城市和发车日期之后点击查询，进入查询界面：



（4）点击购票按钮之后弹出座位选择框：



（5）购票成功之后点击订单按钮进入订单界面：



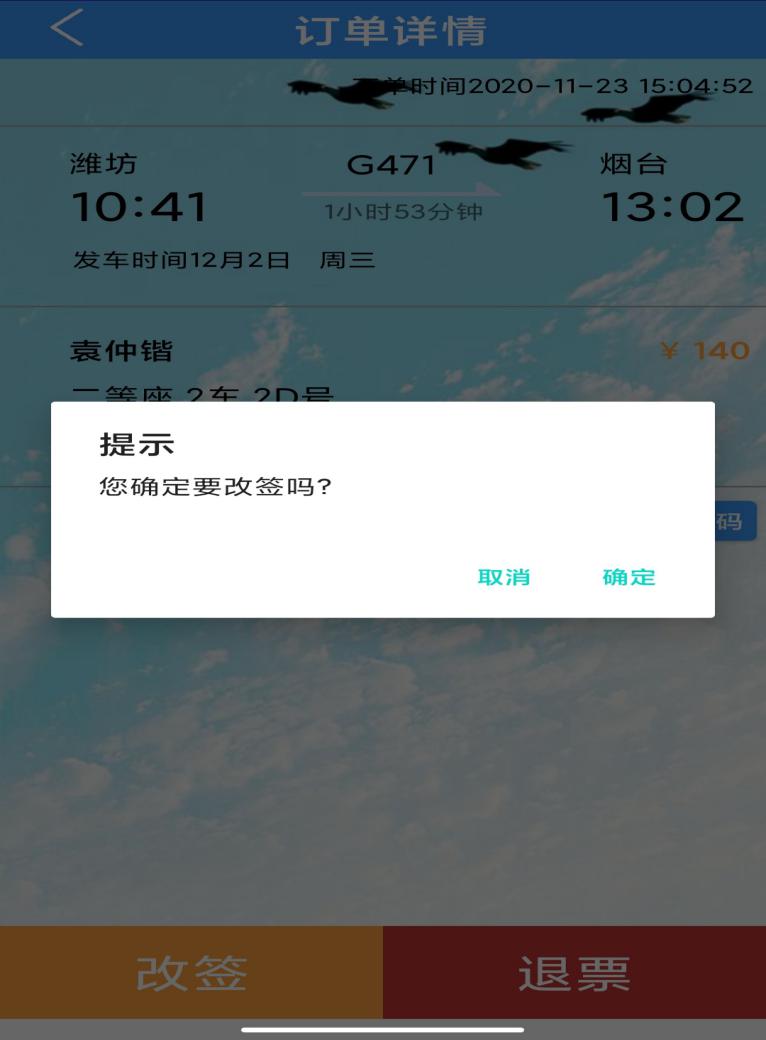
（6）点击该订单进入该订单的详情页面：



（7）点击显示二维码按钮显示二维码：



（8）点击改签弹出提示框：



（9）点击确定改签进入改签界面：



（10）重复购票操作，返回订单页面该订单已改变。

（11）点击退票弹出提示框：



（12）点击确定之后退票成功，返回订单页面发现订单已消失。

（13）点击我的按钮进入我的界面，显示余额：

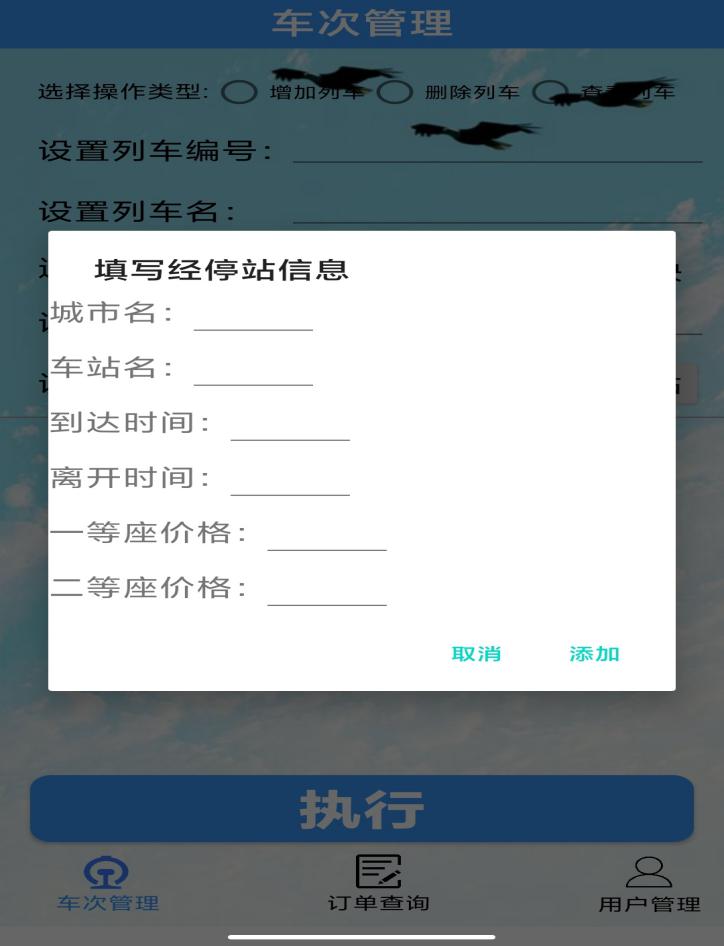


**8.3 管理员系统**

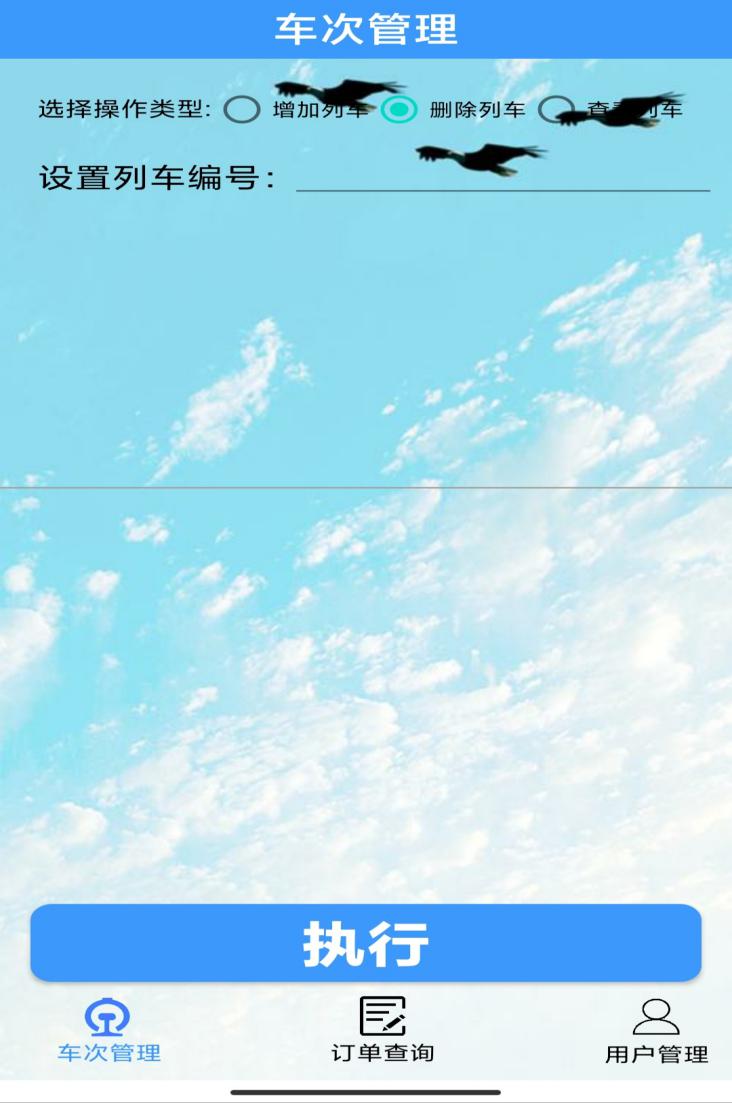
（1）进入管理员系统之后跳转到列车管理界面：



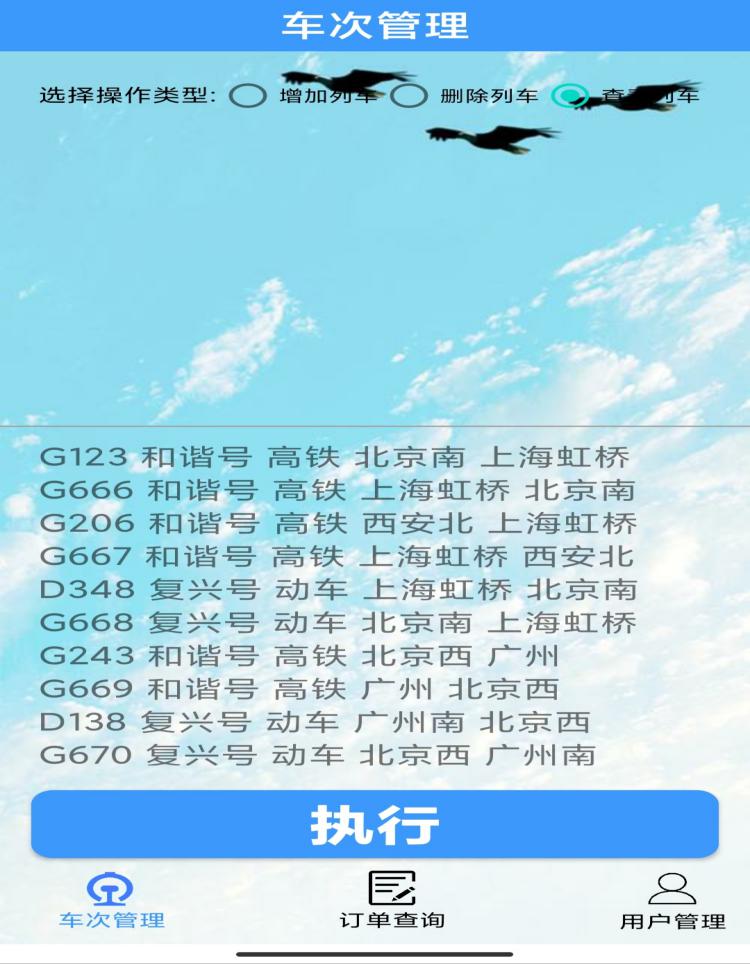
（2）点击添加经停站：



（3）点击删除列车：



（4）点击查看列车并执行：



（5）点击订单查询按钮进入订单查询页面：



（6）按用户名查询：



（7）按发车日期查询：



（8）按下单日期查询：



（9）点击用户管理按钮进入用户管理页面：



（10）选择封号或者是解封并输入用户名，点击执行即可完成相应操作

**九、总结**

**9.1 系统优点**

1. 软件拥有用户端和管理员端两个系统，权限分明，方便用户和管理员两种角色进行操作，防止越权访问的发生。
2. 软件功能强大，拥有12306的主要功能，用户可进行查询，购票，改签，退票，管理员可对列车和订单进行操作，功能齐全。

3、界面简洁美观，操作简单。

4、异常处理机制完善，有清晰明了的错误提示，让用户容易地明确操作错误的原因。

**9.2 系统改进**

在用户信息中加入余额，初始余额为10000，购票时从余额中扣除本次的价格，改签时多退少补，退票时返回费用。