目录

[第一章 云租车管理系统需求分析 4](#_Toc91841328)

[1.导言 4](#_Toc91841329)

[2.系统定义 5](#_Toc91841334)

[3.应用环境 7](#_Toc91841338)

[4.功能规格 8](#_Toc91841342)

[5.性能需求 26](#_Toc91841354)

[6.产品提交 27](#_Toc91841359)

[7.实现约束 27](#_Toc91841360)

[第二章 云租车管理系统概要设计 28](#_Toc91841361)

[1.系统分析 28](#_Toc91841362)

[2.体系结构 28](#_Toc91841363)

[3.界面设计 32](#_Toc91841366)

[4.系统运行环境 37](#_Toc91841367)

[第三章 云租车管理系统详细设计 38](#_Toc91841372)

[1.系统功能模块设计 38](#_Toc91841373)

[2.系统数据库设计 44](#_Toc91841379)

[第四章 设计规范 51](#_Toc91841382)

[1.引言 51](#_Toc91841383)

[2.编码规范 52](#_Toc91841387)

[3.命名规范 53](#_Toc91841392)

[4.声明规范 54](#_Toc91841400)

[5.语句规范 54](#_Toc91841403)

[6.注释规范 55](#_Toc91841406)

[7.目录规范 55](#_Toc91841407)

[第五章 项目实现 56](#_Toc91841408)

[1.技术方案 56](#_Toc91841409)

[2.关键模块的实现技术 57](#_Toc91841410)

[3.特殊问题及解决办法 61](#_Toc91841411)

[第六章 测试计划 63](#_Toc91841412)

[1.引言 63](#_Toc91841413)

[2.任务概述 64](#_Toc91841417)

[3.测试计划 70](#_Toc91841422)

[4.测试项目 72](#_Toc91841428)

[5.评价 85](#_Toc91841430)

# 第一章 云租车管理系统需求分析

## 1.导言

### 1.1 目的

该文档是关于用户对于云租车管理系统的功能和性能的要求，重点描述了云租车管理系统的功能需求，是概要设计阶段的重要输入。

本文档的预期读者是：

· 设计人员；

· 开发人员；

· 项目管理人员；

· 测试人员；

· 用户。

### 1.2 范围

该文档是借助于当前系统的逻辑模型导出目标系统的逻辑模型的，解决整个项目系统的“做什么”的问题。在这里，没有涉及开发技术，而主要是通过建立模型的方式来描述用户的需求，为客户、用户、开发方等不同参与方提供一个交流的平台。

### 1.3 编写说明

无

### 1.4 术语定义

无

## 2.系统定义

我们分别阐述一下项目的来源、背景，项目的用户特点和项目的目标。

### 2.1 项目来源及背景

如今在疫情的背景下，倡导尽量减少聚集，然而线下租车都是聚集到平台管理方进行车辆交易，因此，为了解决这个问题，我们决定设计一个线上云租车平台，为车辆空闲的人和需要临时用车的人提供便利，两者将直接进行交易，平台在其中起到维护和保障双方利益的作用，同时也收取一定的服务费。根据对租车市场现状的分析，一个合理可行的租车平台需要车主发布车辆相关信息，如型号、价格、期限、新旧等，租车人可以在平台上看到车主们发布的车辆相关信息，进行租车服务，同时平台管理员起到连接车主和租车人的作用，对车主发布车辆的相关信息进行核实验证，同时对租车人的信用进行考量，设定租车期限，对于租车过程和还车过程中发生的纠纷进行处理。平台的设计理念是操作简单，界面友好，运行稳定，容易上手。

### 2.2 用户的特点

本系统的用户都是网上用户，包括三类，第一类“管理员”，他们是云租车系统的管理人员。“管理员”负责核实验证“车主”发布的车辆信息、考察“租车人”的信用、解决交易纠纷以及管理“车主”和“租车人”的用户信息。第二类“车主”，他们是车辆的提供方，负责提供自己空闲的车辆给“租车人”使用，并收取一定的费用。第三类是“租车人”，可选择可供租的车辆进行租用，并付一定的费用。

### 2.3 项目目标

本项目设定的目标如下：

· 系统能够提供友好的用户界面，使操作人员的工作量最大限度的减少；

· 系统运行的效率一般，但是运算出结果所需的时间应该是用户在可以接受的范围内 （<=1秒）；

· 平台的设计具有一定的灵活性，能够适应云租车管理的变化；

· 通过这个项目可以锻炼自己，提高开发能力和项目管理能力。

## 3.应用环境

根据用户的需求陈述，可以确定本项目分为客户端（车主、租车人）和管理端（管理员）。

客户端主要功能是“账户信息管理”、“在平台上发布车辆出租信息”、“与租车人沟通”、“与平台管理员沟通”、“与车主沟通”、“租用车辆”。

管理端的功能提供“账户管理”、“客户管理”、“车辆管理”、“与客户沟通”。

### 3.1 系统运行的网络环境

无论是客户端的车主、租车人还是管理端的管理员都可以通过网络登录到本系统中。车主通过网络发布待租车辆，租车人通过网络选择需要租用的车辆。管理端的管理员负责维护平台秩序，审核车辆信息，评估用户信誉，解决交易纠纷。

### 3.2 系统运行的硬件环境

本系统的硬件环境如下：

客户机：普通PC

· CPU：2.8GHz以上

· 内存：4GB以上

· 能够运行IE11.0以上版本的机器

· 分辨率：推荐使用1024×768以上像素

Web服务器

· CPU：2.8GHz以上

· 内存：4GB以上

· 硬盘：64GB以上

· 网卡：1Mb/s

数据库服务器

·CPU：2.8GHz以上

· 内存：4GB以上

· 硬盘：64GB以上

### 系统运行软件环境

本系统的软件环境如下：

· 操作系统：Win8或以上版本

· 数据库：mysql5.0

· 开发工具包：JDK/IDEA

·Web服务器：Tomcat 10

·浏览器：IE11.0以上

## 4.功能规格

我们采用面向对象分析作为主要的系统建模方法，使用UML（Unified Modeling Language）作为建模语言。UML为建模活动提供了从不同角度观察和展示系统的各种特征的方法。在UML中，从任何一个角度对系统所作的抽象都可能需要几种模型来描述，而这些来自不同角度的模型图最终组成了系统的映像。

用例描述角色（用户、外部系统以及系统处理）是如何与系统交互来完成工作的。用例模型提供了一个非常重要的方式来界定系统边界以及定义系统功能，同时，该模型将来可以派生出动态对象模型。

设计用例时，我们遵循下列步骤：

1）识别出系统的角色。角色可以是用户、外部系统，甚至是外部处理，通过某种途径与系统交互。重要的是着重从系统外部执行者的角度来描述系统需要提供哪些功能，并指明这些功能的执行者（角色）是谁。尽可能地确保所有角色都被完全识别出来。

2）描述主要的用例。可以采取不断地问自已“这个角色究竟想过系统做什么？”来准确地描述用例。

3）重新审视每个用例，为它们下个详尽的定义。

### 4.1 角色定义

角色或者执行者指与系统产生交互的外部用户或者外部系统。

#### 4.1.1 车主

车主是车辆的拥有者，车主可以在车辆闲置时上传到租车平台，租借给有需要使用的车主，同时获得一定的租车收益。租车人成功租车后支付的金额，一部分用于平台的佣金，一部分算作车主的收益。

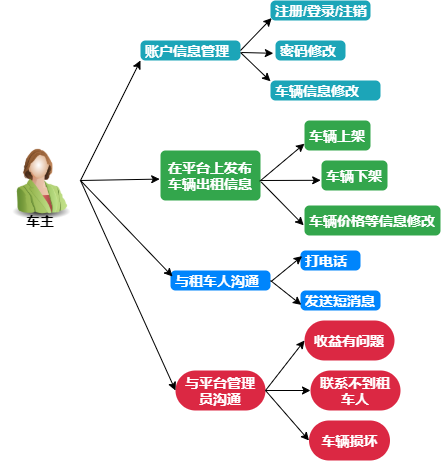


图1-车主需求图

#### 4.1.2 租车人

租车人是向车主申请租车的人。租车人在平台上寻找自己所需要的车型，并联系对应的车主进行租车服务。租车完成后，按照事先约定的规则支付一定的费用。对于这些管理，没有具体细分，有一个人来完成。

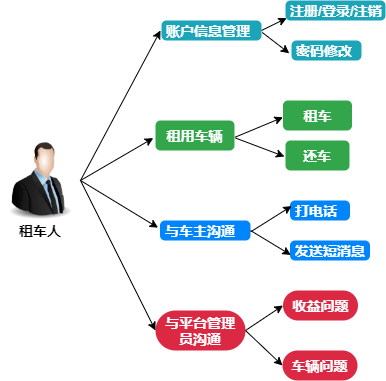


图2-租车人需求图

#### 4.1.3 平台管理员

在车主和租车人中间起到连接作用，维护系统的正常运行，查看平台用户数据，对违规行为进行纠正，对交易纠纷就行处理，保证交易的顺利进行，同时收取一定佣金用于平台运行成本。

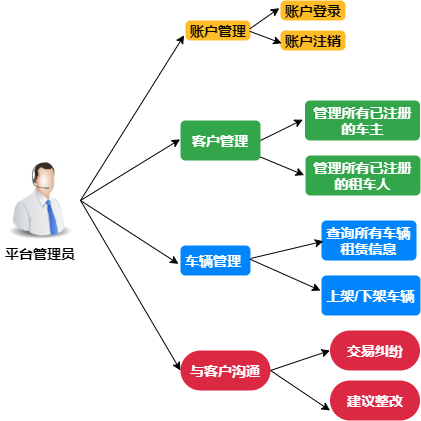


图3-平台管理员需求图

综上，系统主要包含车主、租车人以及平台管理员三方角色，他们相互联系，各有需求，共同参与这个系统的正常运行。

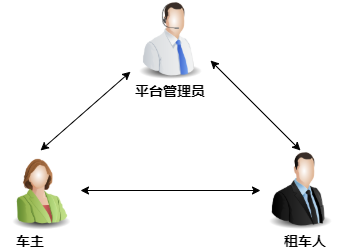


图4-系统角色图

系统各个角色对面向终端的云租车交易平台的整体需求如表所示。

|  |  |  |
| --- | --- | --- |
| **用户群** | **角色** | **需求** |
| **有空闲车辆的车主、汽车租赁商** | **车主** | **在平台上发布所要出租的车辆信息** |
| **通过平台与交易方达成交易** |
| **与租车人沟通** |
| **与平台管理员沟通** |
| **需要短期租赁汽车的客户** | **租车人** | **在平台上寻找需要租赁的车辆信息** |
| **通过平台申请租车** |
| **与车主联系** |
| **与平台管理员联系** |
| **通过平台还车** |
| **系统内部人员** | **平台管理员** | **客户管理** |
| **车辆管理** |
| **与客户沟通** |

### 4.2 系统功能需求分析

根据对汽车租赁市场的观察与分析，面向线上的的租车平台需要有充足的可租赁汽车，一定量的租车人，可靠的平台管理系统。对车主而言，经常会出现因为工作等原因，车辆长期空闲的状况，而对租车人来说，总会出现因为不经常用车没买车，有时却的确需要一辆车的情况。对于这两类群体的问题，我们设计了一个在线租车系统，车主可以将空闲的车放到平台上出租，而有需要短暂租车的人在平台上选择合适的车辆进行租赁并支付一定费用，一举两得。平台采用JavaWeb网站应用，需要操作简便，界面友好，运行稳定，实现基与互联网的租车系统管理，为租车与被租车方提供实时、便捷的操作和信息化管理手段。

用户需要系统提供多个管理系统，主要包括账户注册、登录、注销、账户信息维护、消息配置等通用功能，以及客户信息管理、租车收益分析、租车信息修改等特异功能，总体的系统功能用例图如图所示:

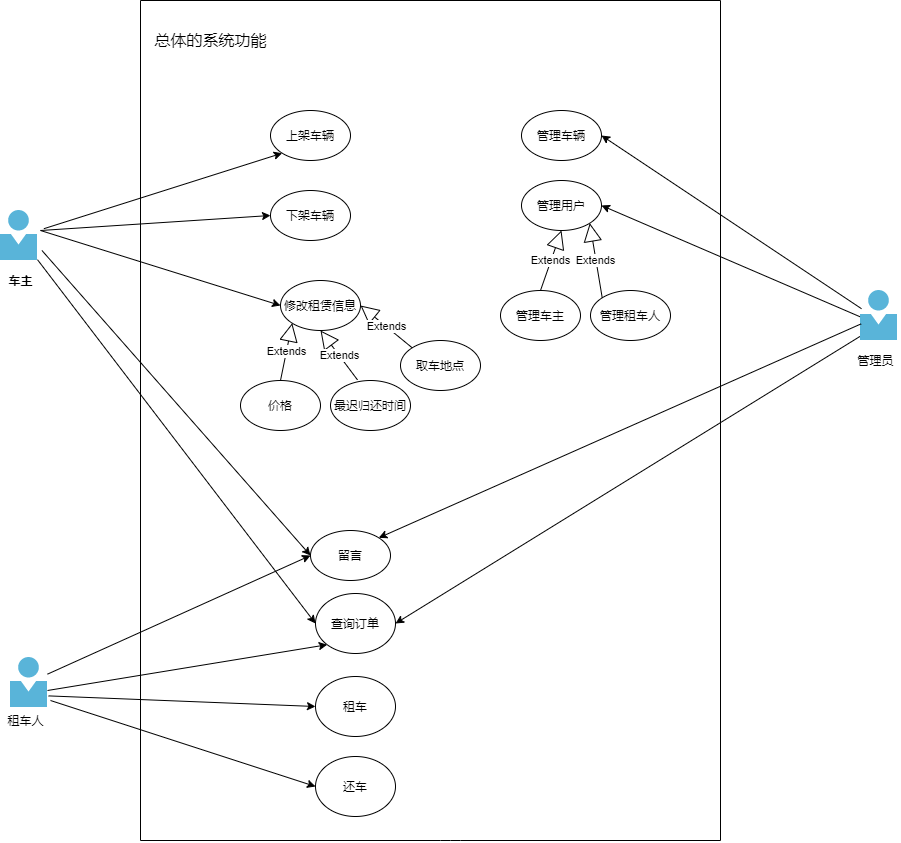


图5-系统功能用例图

### 4.2.1账户管理系统用力分析

用户需要系统提供账户管理系统，主要包括账户注册、登录、注销、账户信息维护、消息配置等通用功能，以及账户系统维护等特异功能。

**账户注册：**无论是车主用户还是租车人用户，在使用系统前首先要为自己注册一个账户，设置用户名、密码，填写手机号、身份证号等基本注册信息，提交注册申请。

**账户登录：**用户使用系统时，需要输入用户名和密码进行登录操作。

**账户注销：**用户不使用系统时，退出登录，避免他人对自己的账户进行误操作。**账户信息维护**：包括忘记密码等功能。用户可以修改密码等账户基础信息；用户如果忘记自己的密码或者用户名，可以通过邮箱等注册信息进行验证，重置密码。

**账户系统维护：**在实际运行中，会产生一些长期不再用的个人信息，或者一些用户在系统使用上肆意违反平台规则，影响系统规范，就需要平台管理员进行操作，例如加入黑名单等。

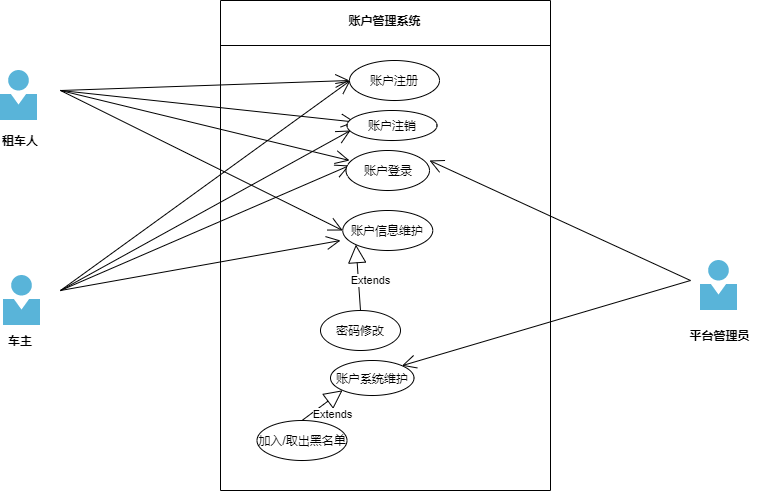


图6-账户管理系统用例

### 4.2.2租车管理系统用例分析

用户租车与被租车都需要在租车管理系统中进行，车主需要发布车辆出租信息，并拥有出租信息修改、收益查询、问题申诉等功能，而租车人有选择租赁汽车、问题申诉等主要功能。

**发布租车信息：**车主在平台上提交车辆出租信息，包括收费标准，车辆描述，以及相关备注信息等，以便于租车人选择。

**修改车辆信息：**车辆出租信息发布后，如果还未被租赁，车主可以对车辆信息进行更改。

**选择租赁汽车：**租车人选择合适的租车信息，进行租车，车主确认后，开始进行租赁计费。

**消息配置：**用户可以通过平台获取用户联系方式，通过站内留言系统与其他系统角色进行沟通交流。

**问题申诉：**当车主与租车人交易产生纠纷时，如账户问题、收益问题等，可对问题进行描述，反馈给平台管理员以求解决。主要途径为消息配置功能，留言方式进行沟通解决。

**解决问题：**平台管理员看到问题申诉后，对问题进行查看，根据平台规则进行交涉。

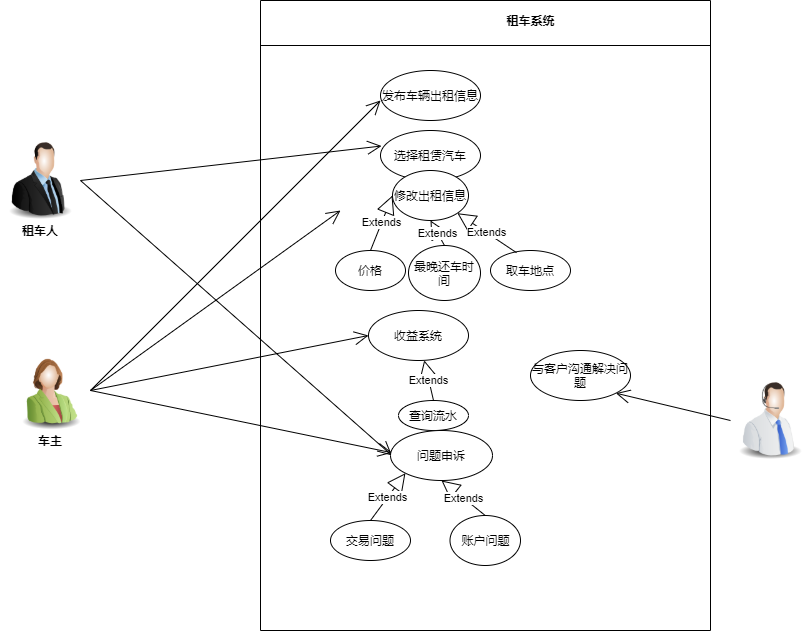


图7-租车管理系统用例分析

### 4.2.3系统功能性需求概览

在系统的每一项功能中，系统的三个角色都有不同的定位设置，总体来看，主要包含账户管理业务、租/还车业务、收益业务、留言业务，相互联结共同构成云租车系统。其系统概览图如下所示：

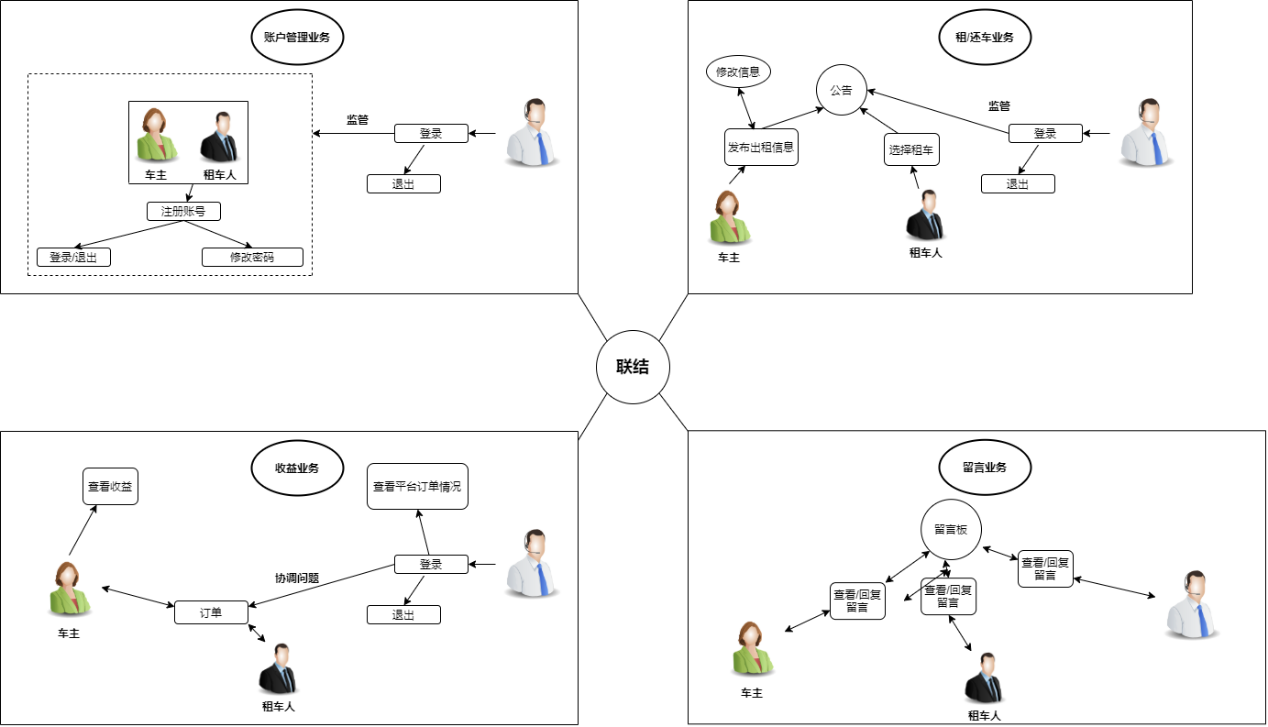


图8-系统功能性需求概览

### 4.2.4系统功能性需求概览

#### 4.2.4.1申请租车

租车人通过此业务，浏览可租赁的所有车辆信息，对满意的信息，可发起租车申请，待车主同意后，可进行协商取车。

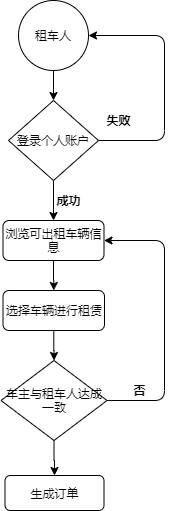


图9-申请租车

|  |  |
| --- | --- |
| **用例名称** | 申请租车 |
| **用例描述** | 租车人通过此用例向系统查询车辆信息并提交租车申请 |
| **主要参与者** | 租车人 |
| **前置条件** | 用户登录界面，通过验证。 |
| **后置条件** | 1. 创建租车记录 2. 更新租车人申请记录 |
| **主事件流** | 1.用户根据以前注册的账号登录系统，计算机显示可出租的所有车辆及车主信息  2.用户选择所需的车辆，计算机显示相应可出租车信息服务界面  3.用户选择申请租车，计算机显示申请结果。  4.计算机执行后置条件。用例结束 |
| **备选事件流** | 1. 用户信息不正确  1.1 拒绝登录，用例结束  2. 用户选择查看原有订单  2.1 计算机显示订单记录界面;  3. 用户选择放弃租车  4. 用户给车主留言询问出租信息 |
| **业务规则** | 根据用户选定的车辆显示对应不同车主与租车人的距离和租车的价格，租车人选择适合自己的，经过车主和租车人协商后可租出 |
| **涉及的业务实体** | 消息留言、车辆出租、订单、个人信息验证 |
| **非功能性需求** | 支持不同地区的用户协商达成交易 |

#### 4.2.4.2归还车辆

租车人使用完车辆后，将车辆开到租车地，将车辆归还车主，并在平台上按照之前的约定支付相应的费用，结束此次租车订单。

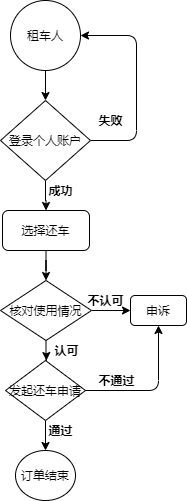


图10-归还车辆

|  |  |
| --- | --- |
| **用例名称** | 归还租车 |
| **用例描述** | 租车人通过此用例向车主归还车辆 |
| **主要参与者** | 租车人、车主 |
| **前置条件** | 1. 用户登录界面，通过验证。 2. 用户成功租车 |
| **后置条件** | 1. 更新租车人使用车辆情况的记录 2. 更新车主收益的记录 3. 更新车辆状态 |
| **主事件流** | 1. 用户根据以前注册的账号登录系统，计算机显示还车界面  2. 用户选择归还车辆，计算机显示使用记录界面  3. 用户确认车辆使用记录，并申请还车  4. 车主同意还车，用户可支付相应金额  5. 用户支付后，计算机显示支付成功结果界面  6. 计算机执行后置条件。用例结束 |
| **备选事件流** | 1. 用户信息不正确  1.1 拒绝登录，用例结束  2. 车主未收到归还车辆，或车辆受损  2.1 拒绝确认还车，请求解决问题 |
| **业务规则** | 用户归还正在租用中的单一车辆 |
| **涉及的业务实体** | 租车业务、还车业务、消息留言业务、个人信息验证、租车记录 |
| **非功能性需求** | 可通过协商，跨地域还车 |

#### 4.2.4.3违约举报

当车主或租车人受到另一方的权益侵害时，可通过平台及时沟通解决，给车主/租车人/平台管理员留言沟通，如果无法解决，可通过违约举报提交相关信息，进行维权。

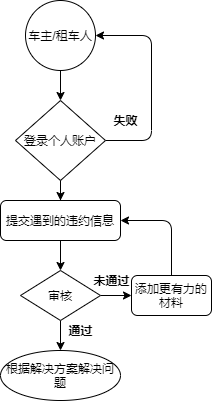


图11-违约举报

|  |  |
| --- | --- |
| **用例名称** | 违约举报 |
| **用例描述** | 用户通过此用例举报违约的另一方 |
| **参与者** | 车主、租车人 |
| **前置条件** | 1.用户登录界面，通过验证。  2. 受到另一方的权益侵犯 |
| **后置条件** | 1. 更新问题订单信息 2. 更新处理违约人记录 |
| **主事件流** | 1. 用户根据以前注册的账号登录系统，提交举报信息 2. 用户输入订单号，及相关描述信息并提交 3. 管理员监督该订单的问题，查看信息，并给出解决方案 4. 用户查看解决方案 5. 计算机执行后置条件。用例结束 |
| **备选事件流** | 1. 用户信息不正确  1.1 拒绝登录，用例结束  2. 用户接受解决方案  2.1 更新后台举报信息的状态  3. 用户不接受解决方案  3.1 提供更有效证据继续提交  3.2 向其他部门进行申诉 |
| **业务规则** | 用户提供被侵权信息提出举报 |
| **涉及的业务实体** | 订单信息、个人信息验证、消息管理 |
| **非功能性需求** | 无 |

### 4.2.5业务实体分析

基本结构：使用者使用目标系统能完成的业务--业务说明--业务场景分析--特定业务实现过程中涉及到的实体。

#### 4.2.5.1租车业务

租车人通过登录个人的信息进行验证，进入租车页面后选择所需的车辆，进行申请租车服务，根据租车人选定的车辆显示对应不同车主与租车人的距离和租车的价格，租车人选择适合自己的或是没有适合自己的选择退出，并对其最终租车的订单有一个记录。

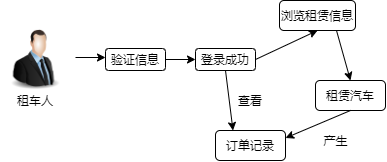


图12-租车业务

#### 4.2.5.2上传车辆信息业务

车主通过登录个人的信息进行验证，进入租车页面上传将要租出车辆的车辆信息，进行申请上架待租车服务，平台管理员审核通过后，等待租车人的租车订单，等车辆被出租后，出租车辆的订单有一个记录。

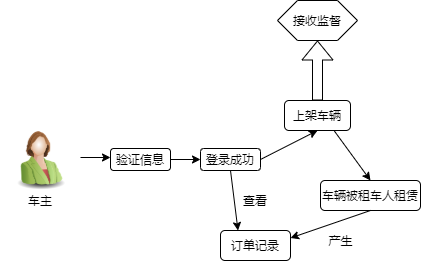


图13-上传车辆信息业务

#### 4.2.5.3车辆使用状况记录业务

在租车系统中，会对车辆的使用情况进行记录，例如此次使用时间、租赁状态、此次租出费用、以及租赁收费标准等。



图14-车辆使用状况记录业务

#### 4.2.5.4归还违规业务

到达规定时间后，租车人未按时归还车辆或者车辆出现了损坏，对其违规行为进行记录，并根据需要给予罚金等惩罚。

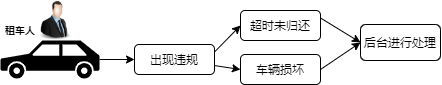


图15-归还违规业务

#### 4.2.5.5顺利还车业务

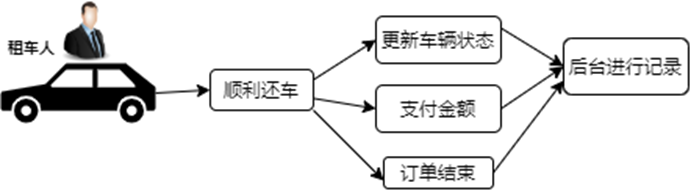
归还车辆顺利结束后，对租车人使用车辆情况进行一次记录，对车主的收益进行一次记录，同时根据使用情况结束订单并更新车辆状态等信息。

图16-顺利还车业务

#### 4.2.5.6留言业务

在车主/租车人个人界面下方或订单下方进行留言，等通过平台管理员的审核之后，对被留言者进行提示，被留言者可以回复留言。

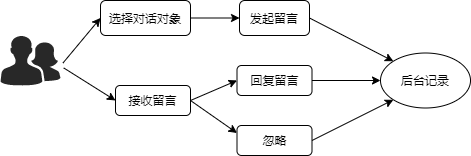


图17-留言业务.

### 4.2.6非功能性需求

#### 4.2.5.1性能方面

1. 后台进行数据库连接时，如果使用一次就创建一个连接，使用完再释放，这过程中会耗费大量时间，必然会影响用户体验。而使用数据库连接池可以很好地解决这一问题，在数据库中按需取出一定数量的连接，使用后再放回连接池，节省了频繁进行数据库连接的时间。
2. 采用数据库访问时，有时会出现多次访问同一数据的情况，采用Mybatis框架二级缓存的方式，可以将一些数据存储到本地，再次访问同一数据时，直接从本地取出，加快速度，减轻数据库负担。
3. 数据库并不很大的时候，采用一次请求，前端分页，提高用户分页体验，降低后台数据库的频繁请求。

#### 4.2.6.2安全方面

1. 登录密码直接使用表单提交，必然存在被恶意盗号的风险，而使用一些加密手段可以很好地保护用户账户信息。例如BASE64加密解密。
2. 使用数据库访问数据时，采用更安全的方式，防止数据库注入问题。

## 5.性能需求

无

### 5.1 界面需求

系统的界面要求如下。

1）页面内容：主题突出，站点定义、术语和行文格式统一、规范、明确、栏目、菜单设置和布局合理，传递的信息准确、及时。内容丰富，文字准确，语句通顺，专用术语规范，行文格式统一规范。

2）导航结构：页面具有明确的导航指示，且便于理解，方便用户使用。

3）技术环境：页面大小适当，能用各种常用浏览器以不同分辨率浏览，无错误链接和空链接；采用CSS处理，控制字体大小和版面布局。

4）艺术风格：界面、版面形象清晰悦目、布局合理，字号大小适宜、字体选择合理，前后一致，美观大方，动与静搭配恰当，动静效果好；色彩和谐自然，与主题内容相协调。

### 5.2 响应时间需求

无论是客户端还是管理端，当用户登录，进行任何操作的时候，系统应该及时地进行反应，反应的时间在1秒以内。系统应能监测出各种非正常情况，如与设备的通信中断，无法连接数据库服务器等，以避免出现长时间等待甚至无响应。

### 5.3 可靠性需求

系统应保证7×24小时内不宕机，保证1500人可以同时在客户端登录，此时系统能正常运行，正确提示相关内容。

### 5.4 开放性需求

系统应具有较强的灵活性，以适应将来功能扩展的需求。

## 6.产品提交

提交产品为：

a）应用系统软件包；

b）数据库中有一些测试是的数据

c）系统开发过程文档；

d）系统使用、维护说明文档。

## 7.实现约束

系统的实现约束如下：

a）操作系统为Windows8

b）开发平台为：IDEA2019

c）数据库为MySQL

# 第二章 云租车管理系统概要设计

## 1.系统分析

本系统可以为车主、租车人、管理员三个角色提供服务，主要功能为车主将闲置车辆添加到网站并发布出租信息，租车人可浏览所有的待租车辆，并选中车辆发起租车请求，再经过车主的同意即可开启交易，最终以相似的方式进行还车；而管理员主要维护系统稳定，解决交易问题，通过留言解决问题。系统整体结构图如下：

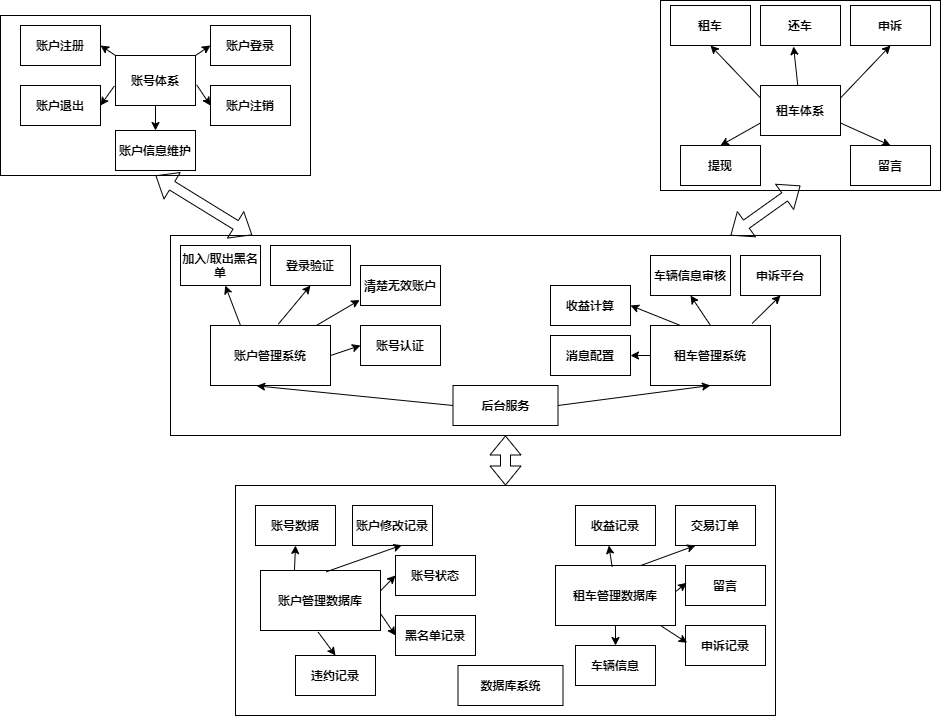


图18-系统的应用结构

## 2.体系结构

### 2.1 系统软件架构设计

线上云租车交易平台的系统设计工作是整个系统阶段实现的重要内容，其中系统架构的设计尤为重要，涉及到系统的总体架构，围绕用户操作界面、业务逻辑处理和数据信息存储等重要内容进行设计。本平台的软件设计采用MVC的设计模式，结合软件架构设计，项目的软件架构各层功能表示如下：

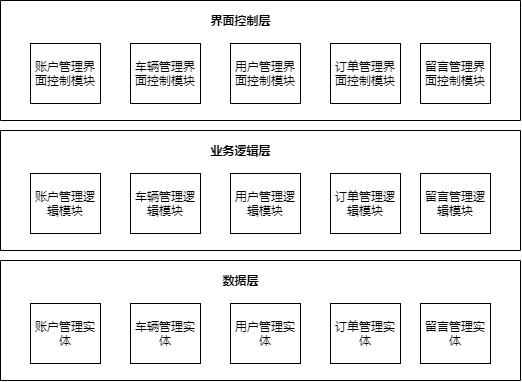


图19-三层结构

从上图可以看出MVC软件架构设计方面的内容，按照业务类型、功能类别、关联关系等方式，对系统每个逻辑层涉及的相关功能和服务进行模块设计，每一层操作的具体内容如下：

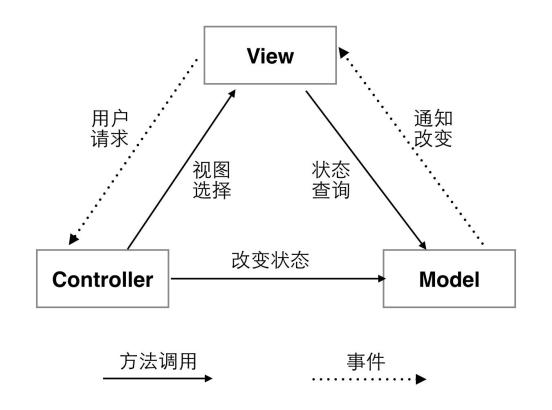


图20-关系图

**（一）视图 View**

主要负责系统数据的展现，同时接受用户的输入数据并对其进行校验。用于接收用户提交请求的代码在这里编写。MVC好处是它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个本地列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。从而使同一个程序可以使用不同的表现形式。

**（二）模型 Model**

表示业务数据和业务规则，是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据。在MVC的三个部件中，模型拥有最多的处理任务。例如它可能用JDBC这样的构件对象来处理数据库，被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。在Model层，又将程序具体分成业务逻辑层(Service)和持久化层(dao)，持久化层负责数据操作，业务逻辑子层负责调用相应的组件（如持久化层组件、其他组件、辅助类等）来组合成一定的逻辑，得到用户请求的数据信息。

**（三）控制器 Controller**

接受用户的输入并调用模型和视图去完成用户的需求，Controller层的主要功能在于控制、组合与调用。所以当单击Web页面中的超链接和发送HTML表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件(即相应的业务逻辑组件)去处理请求，当然执行某些业务逻辑组件的过程中有可能会涉及到数据库操作，但是无论是否涉及到数据库操作，处理用户请求以获得请求结果的过程都是在Model层完成的，Model层获取结果数据之后，再确定用哪个视图来显示返回的数据。

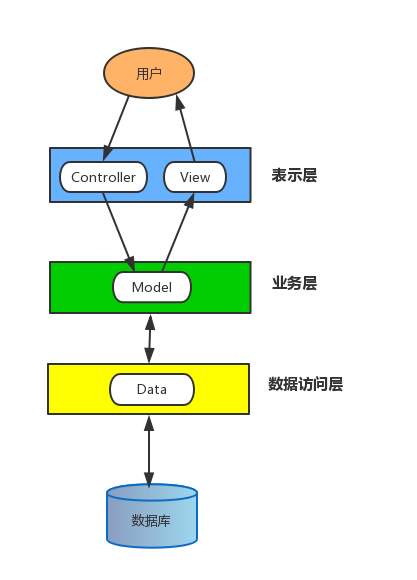


图21-MVC设计模式

而此次课程设计我们采用了较为流行的SSM（Spring+SpringMVC+MyBatis）框架进行开发，Spring主要负责整合，SpringMVC负责实现MVC设计模式，Mybatis负责持久层，主要联系如下图：

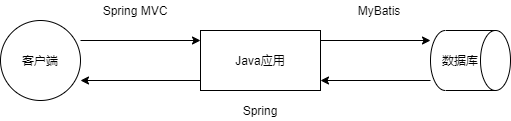


图22-框架联系

### 2.2 系统硬件架构设计

系统硬件架构是从物理部署的角度对系统的总体框架进行分析，云租车交易平台为了能够为地理上离散的多个参与者提供相关服务，因此系统的部署架构在设计上采用分布式架构模式，具体设计如下图所示。

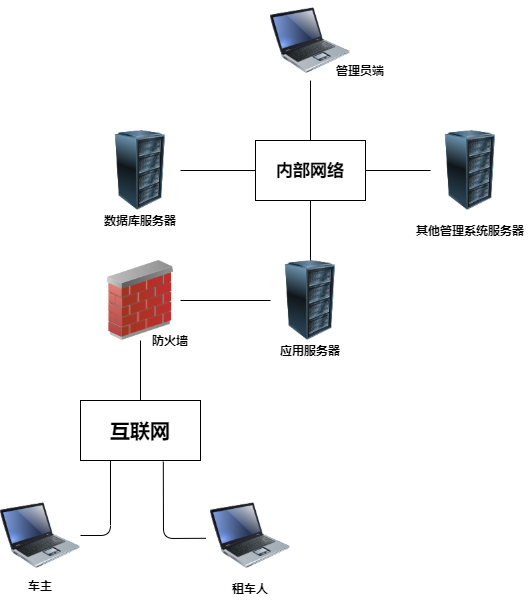


图23-网络结构

从上图可以看出，云租车交易平台系统逻辑架构设计中业务逻辑层的各类功能软件构件大多被部署于应用服务器中，以具体实现系统逻辑架构中的用户接口层。数据库服务器中存储着系统所需要的数据，通过部署于应用服务器中的数据库统一访问构件实现对数据的管理。其他管理系统服务器也连接于系统内部网络中，用以为系统提供数据支撑，或者从本系统中获取所需要的数据。内部网络中的管理员实现对系统各功能的配置工作，为确保系统内部资源的安全性，通过防火墙与外部网隔离。系统的各类用户均可以通过互联网络对系统的功能进行访问和操作，

需要提到的是，为起到学习本门软件课程设计课程的需要，简化了对其他服务器的依赖，主要以本地mysql数据库作为载体进行设计。

## 3.界面设计

本系统主要分为三个角色使用：租车人、车主、平台管理员，针对不同的角色，其对系统界面的需求不同，则其界面设计也有所不同，下面对一些主要的界面设计描述如下：

#### 3.1 主要界面设计

##### 3.1.1 登录界面

用户可以输入账户和密码，并选择自己的身份进行登录，这部分为可复用模块，可根据用户身份自动判定是否登录成功。



图24-登录界面

##### 3.1.2 注册界面

租车人和车主均为平台用户，点击右方“注册“按钮，可自行注册账号，在注册页面输入相关个人信息即可注册成功，界面设计如下：



图25-注册界面

##### 3.1.3 发送留言界面

三个角色均可进行相互留言交流，可选择用户进行交流，也可输入用户唯一id进行留言，其发送留言界面如下：

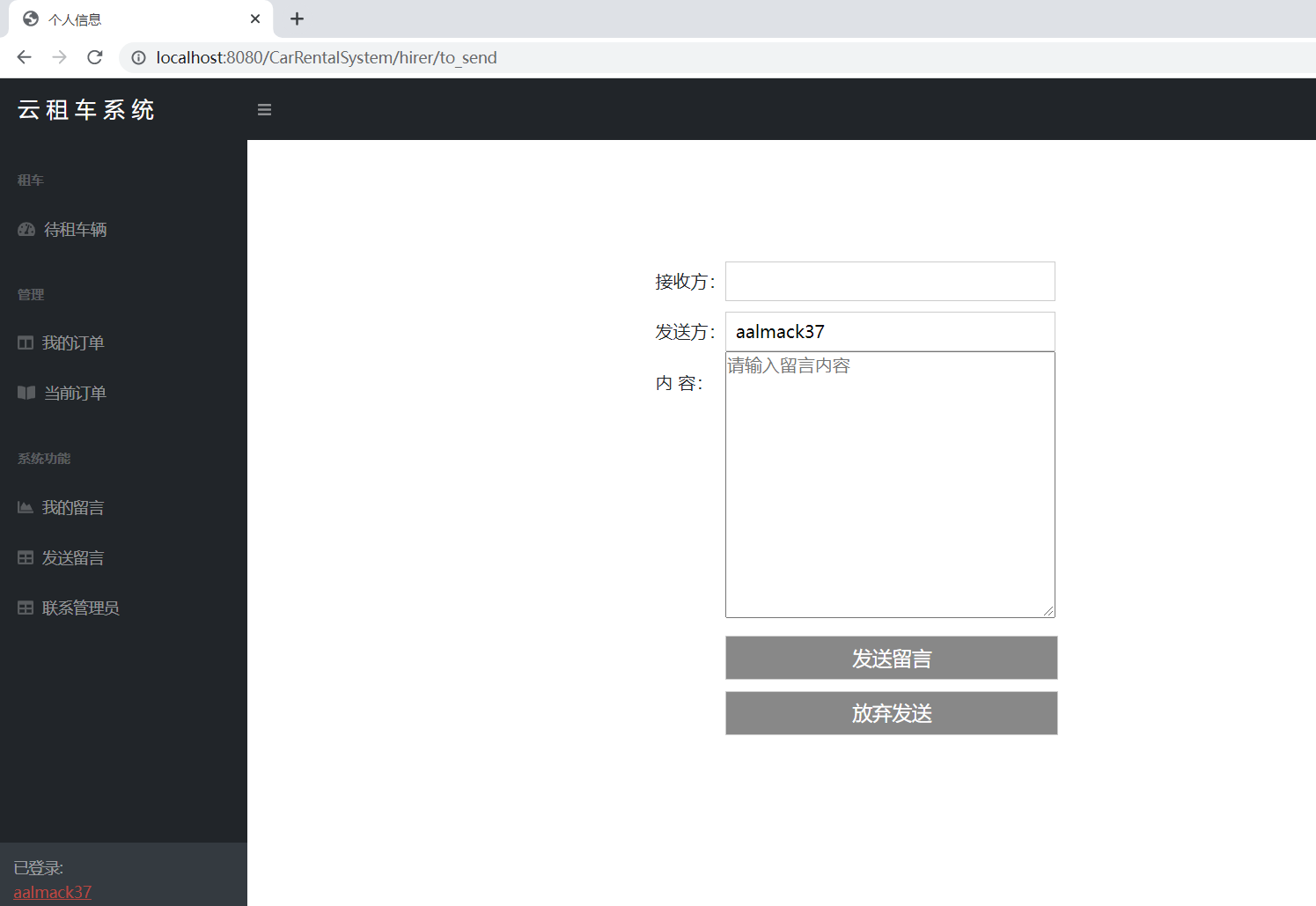


图26-发送留言界面

#### 3.2 角色界面详情

##### 3.2.1 车主端

车主的主要功能有“发布车辆出租信息“、”接受订单预约“、”拒绝订单预约“、”与其他用户留言交流“等，其需要的界面主要有：

* **注册界面：**通过输入用户名、姓名、身份证号、邮箱、手机号、密码、确认密码并选择性别和用户类别（车主/租车人）；
* **登录界面：**通过输入用户名、密码并选择身份（车主/租车人/管理员），点击登录即可进行验证身份；
* **历史订单界面：**能够查看自己所有的历史订单，包括成功与失败的，记录包括订单号、车辆品牌、车型、车牌号、租车人用户名、起止时间、费用及订单状态，方便订单跟踪；
* **车辆上下架界面：**显示当前车主已经上架/下架的车辆，并提供“下架“、”上架“按钮，车主可下架/上架自己的车辆；
* **租车管理界面：**当租车人向车主发起车辆租赁请求时，车主可对该请求进行处理，包括接受或拒绝；
* **还车管理界面：**当租车人需要还车时，向车主发起还车请求，车主可对其处理，当检查车辆没问题后，则接受还车，否则可向管理员等求助；
* **新增车辆界面：**当用户需要新增车辆时，可在此页面填写表单提交；
* **我的留言界面：**用户可查看自己发出/接收到的所有留言，并可对另一方进行回复；
* **联系管理员界面：**用户可查看所有的管理员信息并对其进行留言求助。

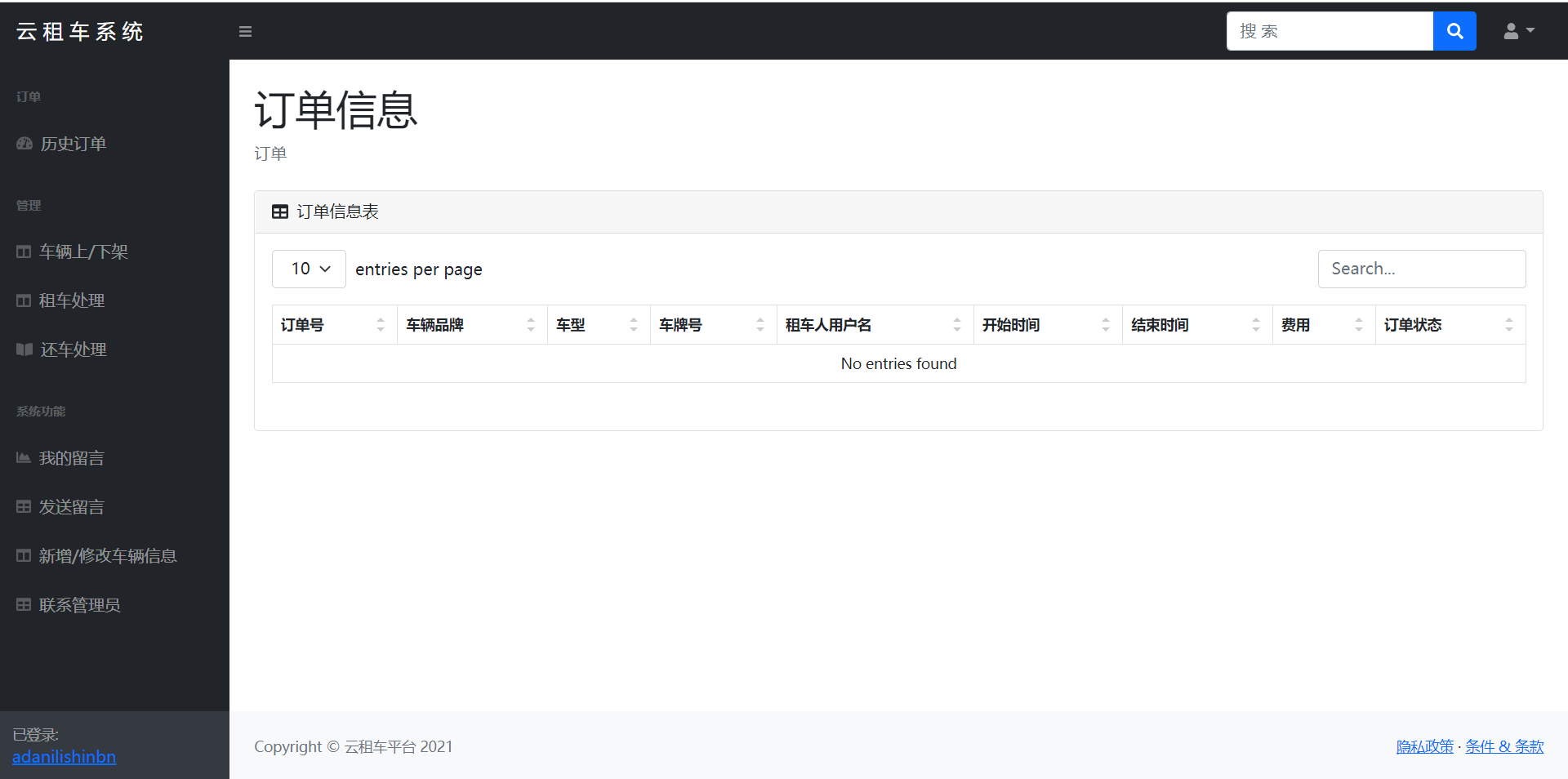


图27-车主端主界面

##### 3.2.2 租车人端

租车人的主要功能有“浏览车辆租赁信息“、”申请租车“、”申请还车“、”与其他用户留言交流“等，其需要的界面主要有：

* **注册界面：**通过输入用户名、姓名、身份证号、邮箱、手机号、密码、确认密码并选择性别和用户类别（车主/租车人）；
* **登录界面：**通过输入用户名、密码并选择身份（车主/租车人/管理员），点击登录即可进行验证身份；
* **待租车辆界面：**显示所有待租的车辆信息，包括品牌、型号、车牌号、车主用户名、已租时长、租赁状态、位置、最迟还车时间、价格以及“租车“按钮；
* **我的订单界面：**显示用户的所有订单，方便跟踪记录；
* **当前订单界面：**显示当前进行中的订单，提供按钮，以便发起还车请求；
* **我的留言界面：**用户可查看自己发出/接收到的所有留言，并可对另一方进行回复；
* **联系管理员界面：**用户可查看所有的管理员信息并对其进行留言求助。

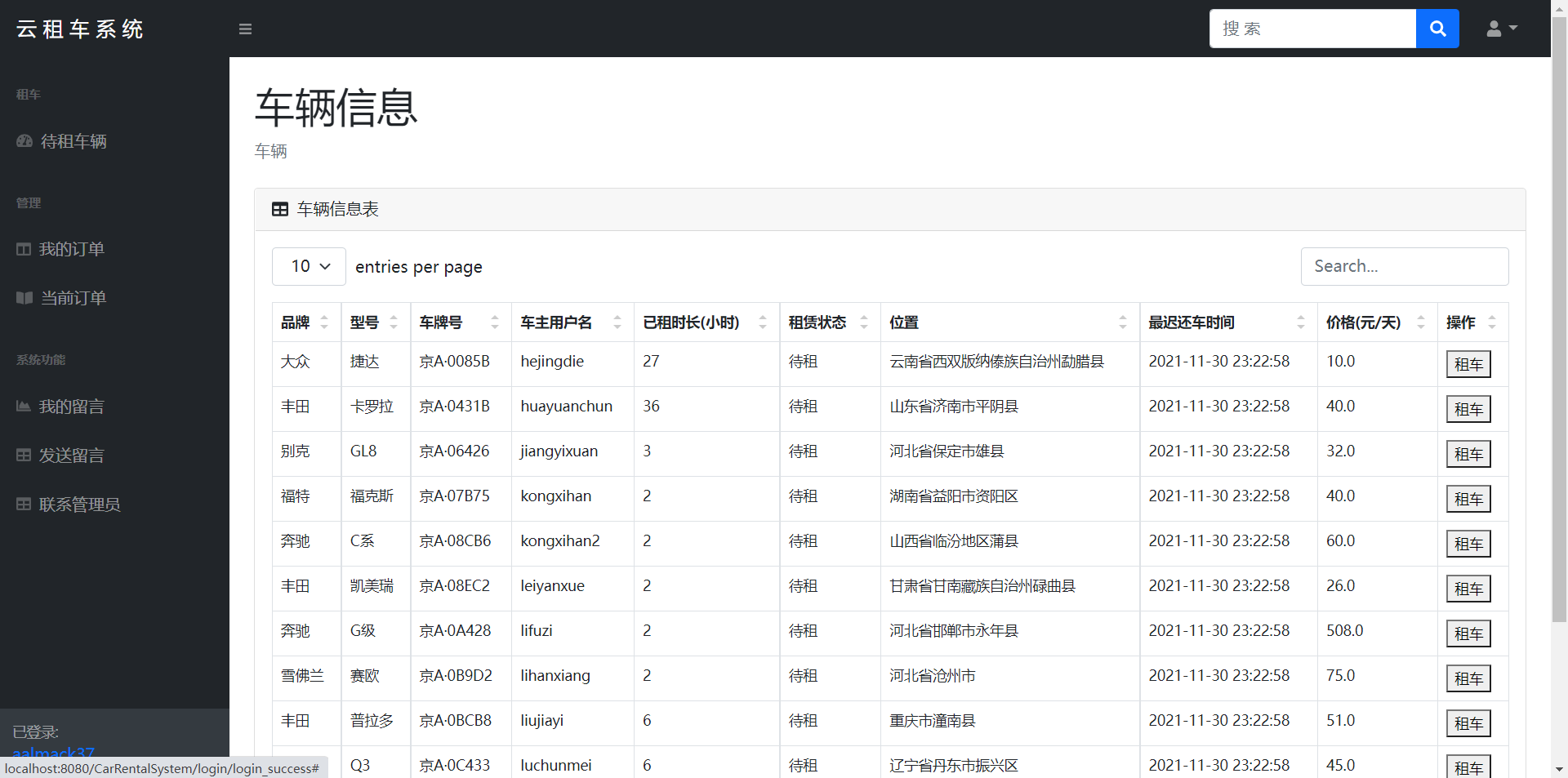


图28-租车人端主界面

##### 3.2.3 管理员端

管理员的主要功能有“客户管理“、”车辆管理“、”留言沟通“等，其需要的界面主要有：

* **登录界面：**通过输入用户名、密码并选择身份（车主/租车人/管理员），点击登录即可进行验证身份；
* **数据分析界面：**显示管理员的主要功能模块，以及订单、用户数据，包括用户信息等；
* **用户管理界面：**可将特定的车主或租车人加入黑名单或取出黑名单；
* **车辆管理界面：**可查看所有的车辆信息并对车辆进行上下架处理；
* **订单管理界面：**管理员可查看所有的订单情况，以便对问题订单进行沟通；
* **我的留言界面：**用户可查看自己发出/接收到的所有留言，并可对另一方进行回复；

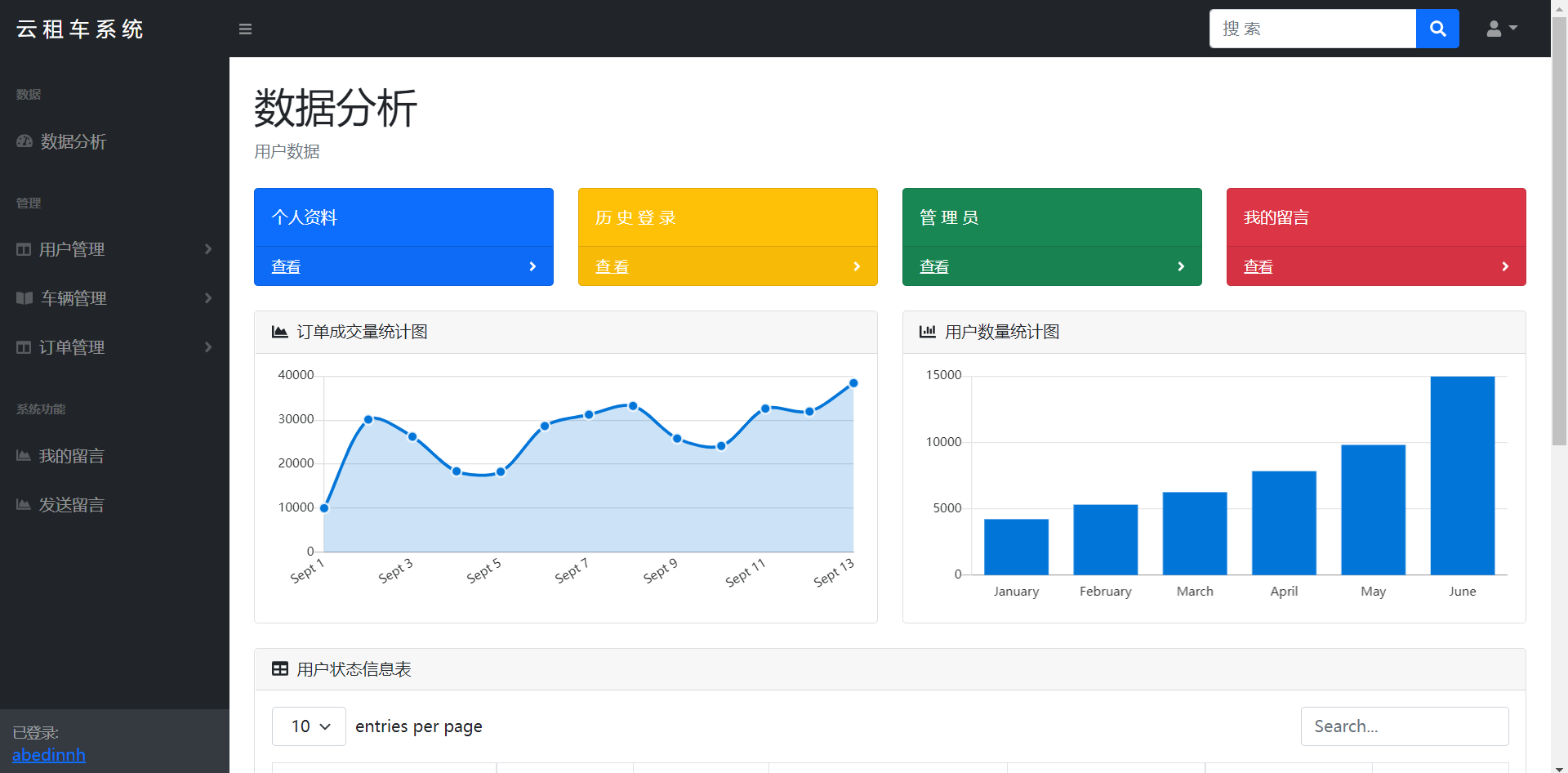


图29-管理员端主界面

## 4.系统运行环境

本项目为C/S结构的web项目，需要服务器部署、客户端使用浏览器进行访问，下面将说明相关的网络结构、硬件环境和软件环境：

### 4.1 网络环境

主要在互联网下部署，其他联网的客户机可随时访问。

### 4.2 硬件环境

**服务器端：**

CPU：intel i5及以上；

内存：8G及以上；

硬盘：512G及以上；

**客户端：**

CPU：intel i5及以上；

内存：8G及以上；

硬盘：512G及以上；

### 4.3 软件环境

* 操作系统：Windows 7及以上；
* 数据库：使用免费开源的MySQL（推荐版本5.0+）；
* Web服务器：免费开源的Tomcat（推荐版本8.0+）；
* 浏览器：推荐使用谷歌浏览器、火狐浏览器等主流浏览器；

### 4.4 开发环境

项目后端框架：SSM(Spring5+SpringMVC+Mybatis)

构建工具：Maven-3.3.9

服务器：tomcat-8.5.51

开发工具：idea2019

Java版本：jdk-1.8

前端框架：bootstrap、jQuery

数据库：MySQL-5.0.17

# 第三章 云租车管理系统详细设计

## 1.系统功能模块设计

基于前文对本云租车系统的需求分析，将系统主要的业务功能模块设计为账户管理、租车管理为主，车辆管理、订单管理、留言管理、用户管理为等几个模块，为车主、租车人等用户提供良好的服务。车主通过在平台上发布车辆租赁信息，租车人在平台上寻求合适的信息，供需双方各取所需，通过平台达成订单交易，平台详细的功能模块如图所示。

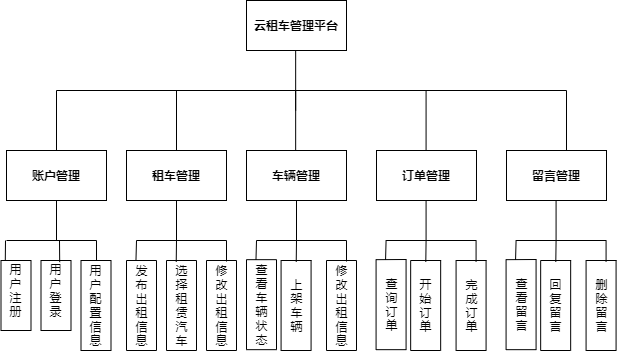


图30-平台功能模块示意图

### 1.1 账户管理模块设计

在云租车系统中，账户管理模块主要负责对系统账户的注册、登录以及信息配置进行管理。系统账户管理类图如下所示，从图中可以看出此业务功能模块的详细参与人员，具体业务类包括车主、租车人、平台管理员。

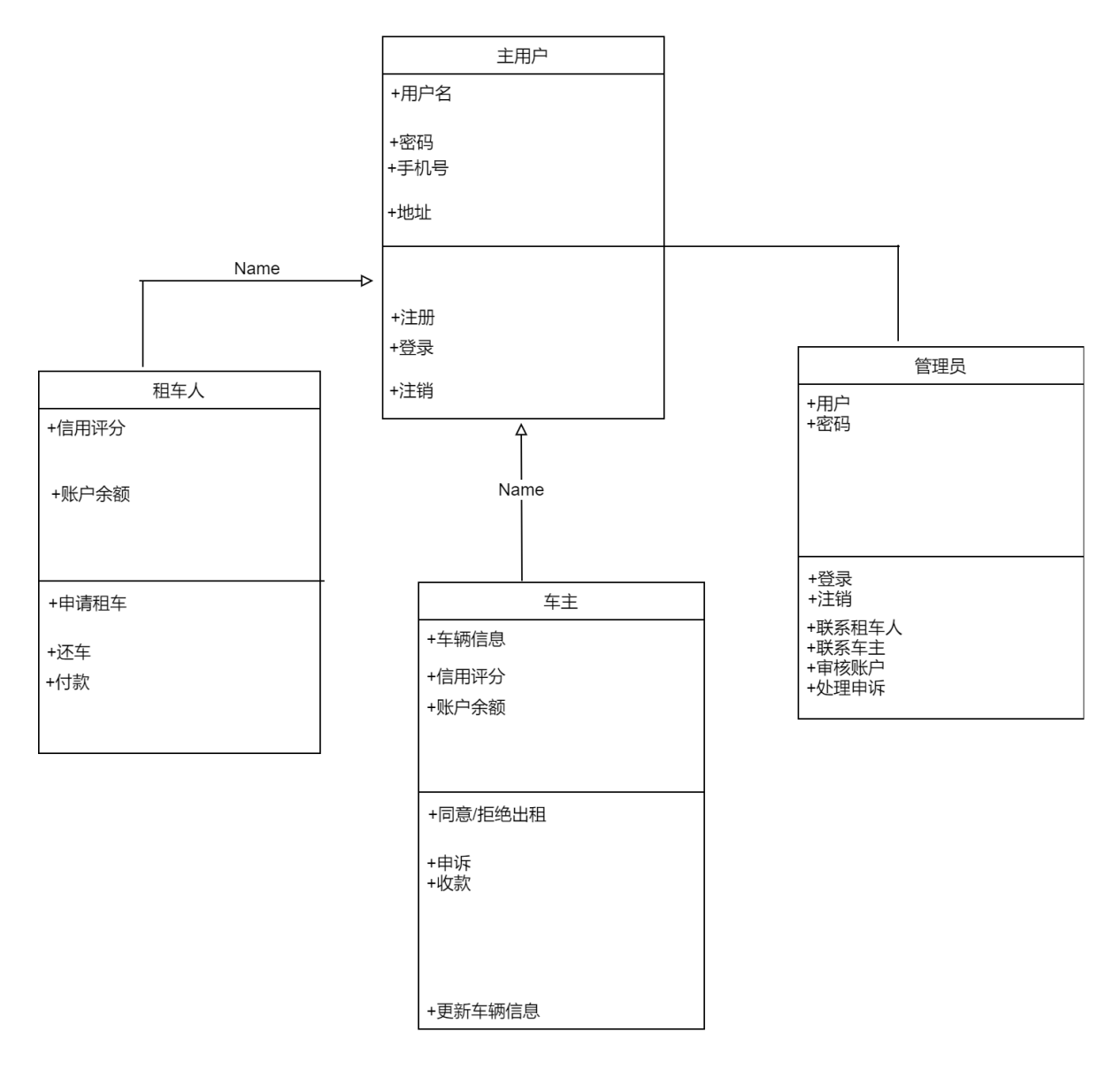


图31-账户管理模块类图

主用户类为租车人类和车主类的父类，有注册、登录、注销等基本功能，含有用户名、密码和手机号等基本信息。

租车人类拥有信用评分和账户余额等基本信息。信用评分来表明租车人是否可信。租车人能够申请租车、还车和付款基本功能。

车主也有信用评分以及车主属下的车辆信息。主要基本功能有同意/拒绝出租、申诉和更新车辆状况等基本功能

管理员类只有用户名和密码属性已验证账户，具有登录、注销等基本功能，并且管理员能联系租车人和车主，以维护用户利益和平台规则，最终的是管理员拥有处理申诉的权利。

**（一）账户注册功能表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 输入项 | 具体要求 | 是否必填 |
| 1 | 用户角色 | 提供“租车人”和“车主”两种选项供用户选择 | 是 |
| 2 | 用户名 | 只能由4-16位字母、数字下划线组成,具有唯一性 | 是 |
| 3 | 密码 | 最少6位 | 是 |
| 4 | 确认密码 | 必须与“密码”输入项内容一致 | 是 |
| 5 | 手机号 | 必须为11位数字 | 是 |
| 6 | 验证码 | 验证码在10分钟内有效 | 是 |
| 7 | 身份证号 | 符合规范 | 是 |

系统依照表中的具体要求，对用户输入的注册信息进行合法校验，若信息不合法，系统将给出用户提示，告知其具体非法项，例如用户名设置重复，提示用户“用户名已存在”；两次密码不同则提示“密码输入不正确”；验证码错误则提示“验证码错误”；信息校验无误后，系统返回注册成功信息，提示用户“注册成功”。

### 1.2 租车管理模块设计

用户租车与被租车都需要在租车管理系统中进行，车主需要发布车辆出租信息，并拥有出租信息修改、收益查询、问题申诉等功能，而租车人有选择租赁汽车、问题申诉等主要功能。

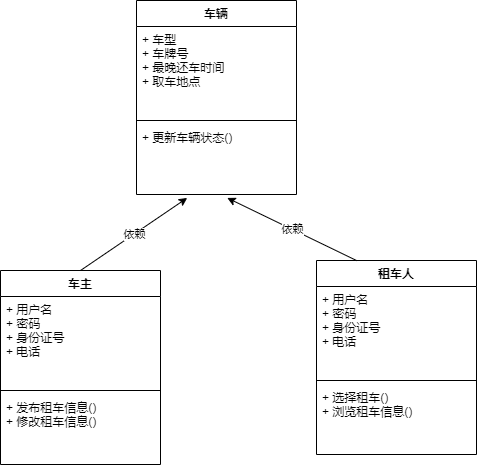


图32-租车管理模块类图

总体来说, 该模块主要功能：车主发布租车信息，在平台上提交车辆出租信息，包括收费标准，车辆描述，以及相关备注信息等，以便于租车人选择。租车人选择合适的租车信息，进行租车，并告知车主，同时进行租赁计费。

### 1.3 车辆管理模块设计

车辆拥有多项基本属性，包括车主用户名、车牌号、已租时长、状态、出租价格、取车地点等，车辆是本系统的重要组成部分，车主是车辆的拥有者，可以发布车辆信息；租车人是车辆的租赁者，可以选择租赁某辆在租的汽车。而对于车辆管理，主要由车主和管理员负责。

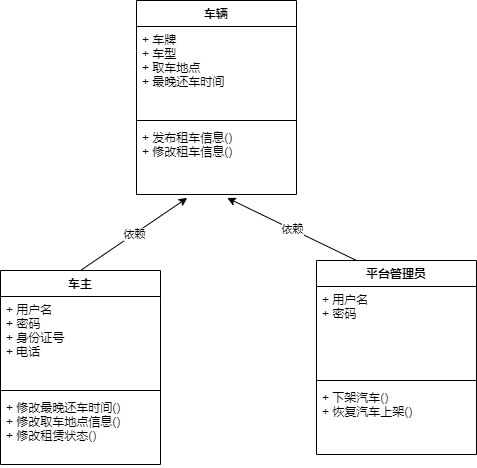


图33-车辆管理模块类图

由类图可知，车辆管理模块主要由车主和平台管理员共同参与，车主对自己的车辆进行基本属性信息的修改，而平台管理员对违规或不适合上架的车辆进行下架，或恢复上架。

车主修改车辆基本信息的修改页面布局大致如下：

### 1.4 订单管理模块设计

订单拥有多项属性信息，包括订单号、车主用户名、租车人用户名、起始时间、完成时间、车型、车牌号、结算金额。订单是车主和租车人进行成功交易的凭证，也是网站流水的重要数据，平台的管理员、车主、租车人都对其有一定权限。

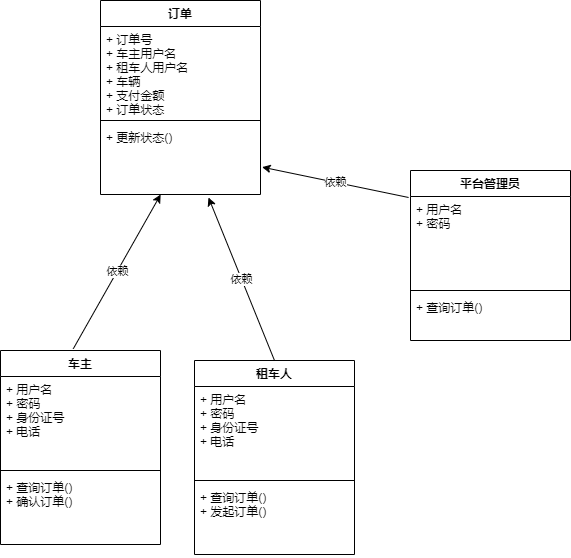


图34-订单管理模块类图

由上图可知，订单管理模块主要用于车主和租车人发起订单并进行交易，车主、租车人以及平台管理员使用不同的权限可对订单进行查询。

### 1.5 留言管理模块设计

拥有多项属性信息，包括留言id、发送方用户名、接收方用户名、留言内容、状态等，该模块主要是作为车主、租车人、平台管理员进行交流的媒介，三者通过这一类可以很容易地获取他人给自己的留言并选择进行回复或删除。

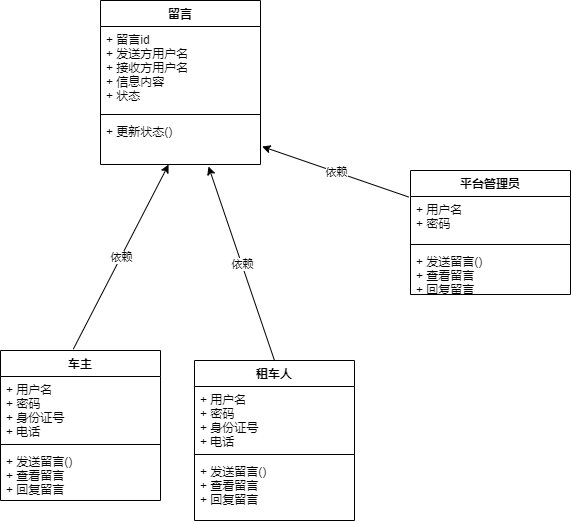


图35-留言模块类图

由上图可见，系统中的各方角色可以使用留言模块进行相互通信，并可实现查看、回复、发送留言的功能。

## 2.系统数据库设计

本小结的工作内容是对《云租车管理系统》后台数据信息存储的数据库进行设计。根据系统业务功能模块设计，构建该租车系统数据信息存储的数据库，当用户点击功能模块进行操作时，系统就会调用业务逻辑程序对应功能模块的数据信息进行处理。本系统采用MySQL作为系统开发的数据库。

系统数据库的设计分为逻辑结构设计和详细数据设计两个方面，数据库逻辑结构设计主要是针对数据库的存储进行数据库实体的提取，数据库详细设计主要是根据数据库实体进行数据库表格的详细设计。

### 2.1 逻辑结构设计

在本系统中，数据库的信息存储是整个平台的核心，各个业务功能模块的点击都需要对对应模块的数据库信息进行操作，因此，为了确保数据库信息存储的完整性以及一致性，首先需要进行逻辑结构的设计，从而确定系统的数据库实体。本文为系统平台系统设计的数据库实体有车主、租车人、平台管理员、车辆、交易订单、留言等，具体的设计结果通过如下的E-R图进行展现，在E-R图中给出每一个数据库实体对应的属性信息以及他们之间的关系，本云租车系统的详细E-R图如下图所示。

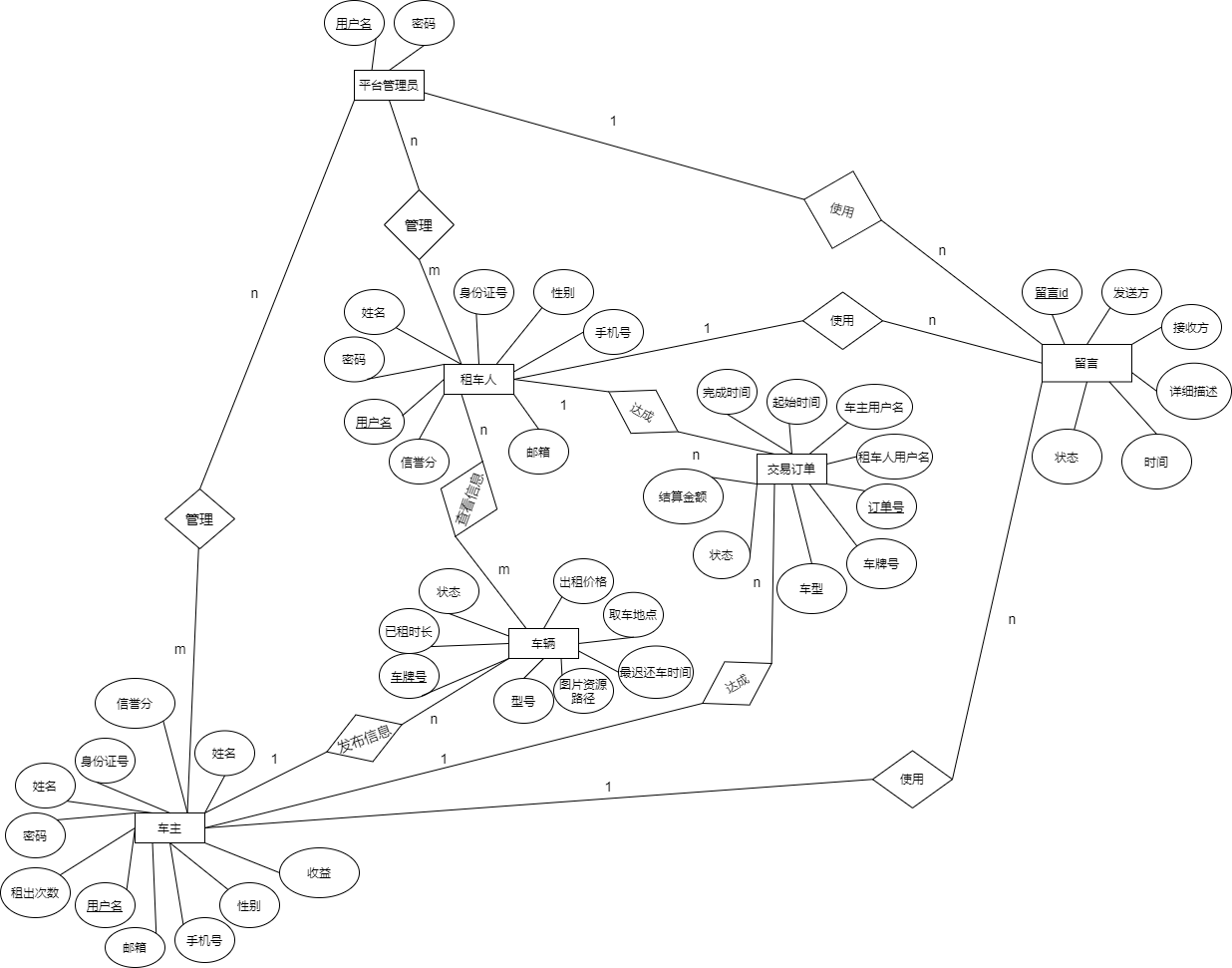


图36-《云租车管理系统》E-R图

**E-R图说明：**

**车主**

拥有多项个人基本属性信息，包括用户名、密码、姓名、性别、身份证号、手机号、账户收益，信誉分、邮箱，其中“用户名”为车主的唯一标识。

**租车人**

拥有多项个人基本属性信息，包括用户名、密码、姓名、性别、身份证号、手机号，其中“用户名“为租车人的唯一标识。

**平台管理员**

拥有用户名和密码两个基本属性，且“用户名“为平台管理员的唯一标识。

**车辆**

拥有多项基本属性，包括车主用户名、车牌号、已租时长、状态、出租价格、取车地点、型号、图片资源路径，其中车牌号为车辆的唯一标识。

对该实体的部分属性解释如下：

已租时长：该车在此平台上已经租出的累计时长。

状态：包括待租/已租/已下架三种状态，以说明该车是否可以租出的情况。

出租价格：该车的出租价格，设定单位为 元/小时。

取车地点：车主就近选择一个取车地点，客户可根据这一地点决策/取车。

型号：该车的品牌及型号，以使客户判断是否适合自己。

图片资源路径：该车的预览图在系统中的存放路径。

**交易订单**

拥有多项属性信息，包括订单号、车主用户名、租车人用户名、起始时间、完成时间、车型、车牌号、结算金额，其中订单号为交易订单的唯一标识。

对该实体的部分属性解释如下：

起始时间：订单经双方确认，正式开始的时间点。

完成时间：该订单最终确认结束的时间点。

车型：此次租赁订单的被租的车辆车型。

车牌号：此次租赁订单的被租的车辆车牌号。

结算金额：此次订单结束后，根据平台规则所支付的费用。

**留言**

拥有多项属性信息，包括留言id、发送方用户名、接收方用户名、留言内容、留言时间、状态。

对该实体的部分属性解释如下：

状态：该条留言的状态，包含已读/未读两种情况。

**实体间的关系**

E-R图中的多个实体间有着密不可分的关系，车主可以发布车辆租赁信息，相应的，租车人可以浏览车辆出租信息并根据需求自由选择；当车主和租车人达成交易时，生成交易订单，记录此次交易的详细信息，若车主和租车人在系统中有问题求助或进行协商沟通，则可以通过留言实体进行留言。

除此之外，平台管理员拥有较高的管理权限，可以对车主和租车人进行管理，促进系统的健康稳定运行。系统中的各个实体相互协调，共同参与并维持着《云租车管理系统》的正常运转。

### 2.2 物理结构设计

#### 2.2.1 数据表

根据上文数据库逻辑结构设计的说明，结合数据库设计理论，本系统平台的主要数据用表如下：

系统主要数据表清单

|  |  |  |
| --- | --- | --- |
| **序号** | **数据表** | **数据表描述** |
| 1 | LOGIN | 用户登录信息表 |
| 2 | OWNER\_OF\_VEHICLE | 车主详细信息主表 |
| 3 | HIRER\_OF\_VEHICLE | 租车人详细信息主表 |
| 4 | VEHICLE | 车辆信息详表 |
| 5 | ORDER | 交易订单表 |
| 6 | MESSAGE | 留言表 |

#### 2.2.2 详细设计

系统关键数据表设计详情如下：

（1）用户登录信息表，表名为LOGIN，数据设计详情如下：

用户登录信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 字段类型 | 可空 | 默认值 | 主键/主键组成 |
| USER\_NAME | 用户名 | VARCHAR(60) | 否 | 无 | 是 |
| PASSWORD | 密码 | VARCHAR(30) | 否 | 无 | 否 |
| LAST\_TIME | 上次登陆时间 | DATETIME | 是 | 无 | 否 |
| TIMES | 总登陆次数 | INT | 否 | 0 | 否 |
| STATE | 用户状态 | INT | 否 | 0 | 否 |
| REGISTER\_TIME | 用户注册时间 | DATETIME | 否 | 无 | 否 |
| USER\_TYPE | 用户类型 | INT | 否 | 1 | 否 |

**说明：**

1. STATUS为用户账户状态，设定0为未登录，1为已登录，-1为已注销；

2. USER\_TYPE为用户类型，设定0为平台管理员，1为租车人，2为车主。

（2）车主详细信息主表，表名为OWNER\_OF\_VEHICLE，数据设计详情如下：

车主详细信息主表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 字段类型 | 可空 | 默认值 | 主键/主键组成 |
| USER\_NAME | 用户名 | VARCHAR(60) | 否 | 无 | 是 |
| NAME | 姓名 | VARCHAR(30) | 否 | 无 | 否 |
| SEX | 性别 | INT | 否 | 1 | 否 |
| ID | 用户身份证号 | VARCHAR(30) | 否 | 无 | 否 |
| CREDIT | 用户信誉分 | INT | 否 | 100 | 否 |
| FREQUENCY | 已租出次数 | INT | 否 | 0 | 否 |
| TEL | 用户手机号 | VARCHAR(30) | 否 | 无 | 否 |
| EMAIL | 用户邮箱 | VARCHAR(30) | 否 | 无 | 否 |
| INCOME | 账户收益 | DOUBLE | 否 | 0 | 否 |

说明：

1. CREDIT为用户信誉分，初始均为100分，最高也是100分，60分为下限；

2. INCOME为用户账户的收益，可用于提现。

（3）租车人详细信息主表，表名为HIRER\_OF\_VEHICLE，数据设计详情如下：

租车人详细信息主表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 字段类型 | 可空 | 默认值 | 主键/主键组成 |
| USER\_NAME | 用户名 | VARCHAR(60) | 否 | 无 | 是 |
| NAME | 姓名 | VARCHAR(30) | 否 | 无 | 否 |
| SEX | 性别 | INT | 否 | 1 | 否 |
| ID | 用户身份证号 | VARCHAR(30) | 否 | 无 | 否 |
| CREDIT | 用户信誉分 | INT | 否 | 0 | 否 |
| FREQUENCY | 已租车次数 | INT | 否 | 0 | 否 |
| EMAIL | 用户邮箱 | VARCHAR(30) | 否 | 无 | 否 |
| TEL | 用户手机号 | VARCHAR(30) | 否 | 无 | 否 |

说明：

1. CREDIT为用户信誉分，初始均为100分，最高也是100分，60分为下限；

（4）管理员详细信息主表，表名为ADMIN，数据库设计详情如下：

管理员详细信息主表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 字段类型 | 可空 | 默认值 | 主键/主键组成 |
| USER\_NAME | 用户名 | VARCHAR(60) | 否 | 无 | 是 |
| NAME | 姓名 | VARCHAR(30) | 否 | 无 | 否 |
| ID | 用户身份证号 | VARCHAR(30) | 否 | 无 | 否 |
| ADMIN\_ID | 工号 | VARCHAR(30) | 否 | 无 | 否 |

（5）车辆信息详表，表名为VEHICLE，数据设计详情如下：

车辆信息详表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 字段类型 | 可空 | 默认值 | 主键/主键组成 |
| VEHICLE\_ID | 车牌号 | VARCHAR(30) | 否 | 无 | 是 |
| VEHICLE\_TYPE | 车型 | VARCHAR(60) | 否 | 无 | 否 |
| OWNER\_ID | 车主用户名 | VARCHAR(60) | 否 | 无 | 否 |
| DURATION | 已租时长 | INT | 否 | 0 | 否 |
| STATUS | 车辆状态 | INT | 否 | 0 | 否 |
| RENTAL\_PRICE | 出租价格 | DOUBLE | 否 | 无 | 否 |
| VEHICLE\_BRAND | 车辆品牌 | VARCHAR(255) | 否 | 无 | 否 |
| DEADLINE | 最迟还车时间 | DATETIME | 是 | 否 | 否 |
| LOCATION | 取车地点 | VARCHAR(255) | 否 | 无 | 否 |

说明：

1. DURATION为已租时长，表示该车在此平台累计租出的总时长，以小时为单位；

2. STATUS为车辆状态，0表示在租，1为已租，2为已下架；

3. RENTAL\_PRICE为出租价格，单位为 元/小时；

4. DEADLINE为最迟还车时间，当最迟还车时间超过仍未还车时，会扣除一定信誉分。

（6）交易订单表，表名为ORDER，数据库设计详情如下：

交易订单表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 字段类型 | 可空 | 默认值 | 主键/主键组成 |
| ORDER\_ID | 订单号 | INT | 否 | 无 | 是（自增） |
| OWNER\_NAME | 车主用户名 | VARCHAR(60) | 否 | 无 | 否 |
| HIRER\_NAME | 租车人用户名 | VARCHAR(60) | 否 | 无 | 否 |
| START\_TIME | 起始时间 | DATETIME | 是 | 无 | 否 |
| END\_TIME | 完成时间 | DATE | 是 | 无 | 否 |
| VEHICLE\_TYPE | 车型 | VARCHAR(60) | 否 | 无 | 否 |
| STATUS | 订单状态 | INT | 否 | 0 | 否 |
| AMOUNT | 结算金额 | DOUBLE | 是 | 无 | 否 |
| VEHICLE\_ID | 车牌号 | VARCHAR(30) | 是 | 无 | 否 |

说明：

1. START\_TIME，表示起始时间，即订单经双方确认，正式开始的时间；

2. END\_TIME，表示完成时间，即订单经双方确认，正式结束的时间；

3. STATUS表示订单状态，0为待开始，1为正在进行，2为正在还车，3为已结束。

（7）留言表，表名为MESSAGE，数据库设计详情如下：

留言表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 字段类型 | 可空 | 默认值 | 主键/主键组成 |
| MESSAGE\_ID | 留言编号 | INT | 否 | 无 | 是（自增） |
| SEND\_NAME | 发送方用户名 | VARCHAR(60) | 否 | 无 | 否 |
| RECEIVE\_NAME | 接收方用户名 | VARCHAR(60) | 否 | 无 | 否 |
| DETAIL | 留言内容 | VARCHAR(255) | 否 | 无 | 否 |
| SEND\_TIME | 留言时间 | DATETIME | 否 | 无 | 否 |
| STATUS | 留言状态 | INT | 否 | 0 | 否 |

说明：

1. STATUS ，留言状态中，0为未读，1为已读。

#### 2.2.3 外键约束

数据库的多种关系之间有着非常密切的关系，有的数据表中的数据也是相互关联的，如果只改变其中一个，那么就会导致其他数据的错误、数据不匹配等问题，因此给数据表加入一定约束是有必要的，一方面能够同步一些关联属性，另一方面能够保护主表，防止误操作导致数据丢失的情况。

**约束1**

令用户登录信息表为主表，车主详细信息表为从表，车主详细信息表的用户名（USER\_NAME）作为外键关联到用户登录信息表的主键（USER\_NAME）。

**约束2**

令用户登录信息表为主表，交易订单表为从表，交易订单表的车主用户名（USER\_NAME）作为外键关联到用户登录信息表的主键（USER\_NAME）。

**约束3**

令用户登录信息表为主表，车辆信息详表为从表，车辆信息详表的车主用户名（OWNER\_ID）作为外键关联到用户登录信息表的主键（USER\_NAME）。

**约束4**

1. 令用户登录信息表为主表，交易订单表为从表，交易订单表的车主用户名（OWNER\_NAME）作为外键关联到用户登录信息表的主键（USER\_NAME）。

2. 令用户登录信息表为主表，交易订单表为从表，交易订单表的租车人用户名（HIRER\_NAME）作为外键关联到用户登录信息表的主键（USER\_NAME）。

**约束5**

1. 令用户登录信息表为主表，留言表为从表，留言表的发送方用户名（SEND\_NAME）作为外键关联到用户登录信息表的主键（USER\_NAME）。

2. 令用户登录信息表为主表，留言表为从表，留言表的接收方用户名（RECEIVE\_NAME）作为外键关联到用户登录信息表的主键（USER\_NAME）。

# 第四章 设计规范

## 1.引言

### 1.1 规范目的

软件编码规范是与特定语言相关的描写如何编写代码的规则集合。编码规范的目的是提高编码质量，避免不必要的程序错误，增强程序代码的可读性、可重用性和可移植性。代码规范不仅使得开发统一，减少审查拿督，而且让代码审查有据可查，大大提高了审查效率和效果，同时代码审查也有助于代码规范的实施。编码规范改善软件的可读性,让工程师更快更彻底地理解新的代码。命名规范使程序更易读，从而更易于理解。它们也可以提供一些有关标识符功能的信息，以助于理解代码。

### 1.2 缩写说明

JSP  
Java Server Page（Java服务器页面）的缩写，一个脚本化的语言。  
  
MVC  
Model-View-Control（模式－视图－控制）的缩写，表示一个三层的结构体系。  
  
EJB  
Enterprise Java Bean（企业级JavaBean）的缩写。

### 1.3 术语定义

Class  
Java程序中的一个程序单位，可以生成很多的实例。  
  
Packages   
由很多的类组成的工作包。

## 2.编码规范

### 2.1 缩进排版

4个空格作为缩进排版的一个单位。

### 2.2 行长度

尽量避免一行的长度超过80个字符，用于文档中的例子应该使用更短的行长，长度一般不超过70个字符。

### 2.3 断行规则

当一个表达式无法容纳在一行内时，可以依据如下一般规则断开：

（1）在一个逗号后面断开

（2）在一个操作符前面断开

（3）新的一行应该与上一行同一级别表达式的开头处对齐

### 2.4 空行

空行将逻辑相关的代码段分隔开，以提高可读性。

下列情况应该总是使用两个空行：

（1）一个源文件的两个片段(section)之间

（2）类声明和接口声明之间

下列情况应该总是使用一个空行：

（1）两个方法之间

（2）方法内的局部变量和方法的第一条语句之间

（3）块注释或单行注释之前

（4）一个方法内的两个逻辑段之间，用以提高可读性

## 3.命名规范

### 3.1 包(Packages)

一个唯一包名的前缀总是全部小写的ASCII字母并且是一个顶级域名，通常是com, edu, gov, mil, net, org。包名的后续部分根据不同机构各自内部的命名规范而不尽相同。

### 3.2 类(Classes)

类名是个一名词，采用大小写混合的方式，每个单词的首字母大写。尽量使你的类名简洁而富于描述。使用完整单词，避免缩写词(除非该缩写词被更广泛使用，像URL，HTML)。

### 3.3 接口(Interfaces)

大小写规则与类名相似。

### 3.4 方法(Methods)

方法名是一个动词，采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。

### 3.5 变量(Variables)

采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。变量名不应以下划线或美元符号开头，尽管这在语法上是允许的。变量名应简短且富于描述。变量名的选用应该易于记忆，即，能够指出其用途。尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为i，j，k，m和n，它们一般用于整型；c，d，e，它们一般用于字符型。

### 3.6 实例变量(Instance Variables)

大小写规则和变量名相似，除了前面需要一个下划线。

### 3.7 常量(Constants)

类常量和ANSI常量的声明，应该全部大写，单词间用下划线隔开。

## 4.声明规范

### 4.1 每行声明变量的数量

推荐一行一个声明，因为这样以利于写注释。不要将不同类型变量的声明放在同一行。  
  
4.2初始化

尽量在声明局部变量的同时初始化。

### 4.3 布局

只在代码块的开始处声明变量。  
4.4包的声明

在多数Java源文件中，第一个非注释行是包语句。  
  
4.5类和接口的声明   
当编写类和接口是，应该遵守以下格式规则：   
（1）在方法名与其参数列表之前的左括号"("间不要有空格   
（2）左大括号"{"位于声明语句同行的末尾   
（3）右大括号"}"另起一行，与相应的声明语句对齐，除非是一个空语句，"}"应紧跟在"{"之后   
（4）方法与方法之间以空行分隔

## 5.语句规范

### 5.1 简单语句

每行至多包含一条语句

### 5.2 复合语句

复合语句是包含在大括号中的语句序列，形如"{ 语句 }"。遵循原则如下：  
（1）被括其中的语句应该较之复合语句缩进一个层次   
（2）左大括号"{"应位于复合语句起始行的行尾；右大括号"}"应另起一行并与复合语句首行对齐。  
（3）大括号可以被用于所有语句，包括单个语句，只要这些语句是诸如if-else或for控制结构的一部分。这样便于添加语句而无需担心由于忘了加括号而引入bug。

## 6.注释规范

在注释里，对设计决策中重要的或者不是显而易见的地方进行说明是可以的，但应避免提供代码中己清晰表达出来的重复信息。

## 7.目录规范

开发环境是IDEA，开发之后需要部署到Tomcat 服务器环境上。所以开发环境的目录结构与运行环境的目录结构是一致的，只是在部署的运行环境中，可以不设置源代码的目录。开发目录如下：

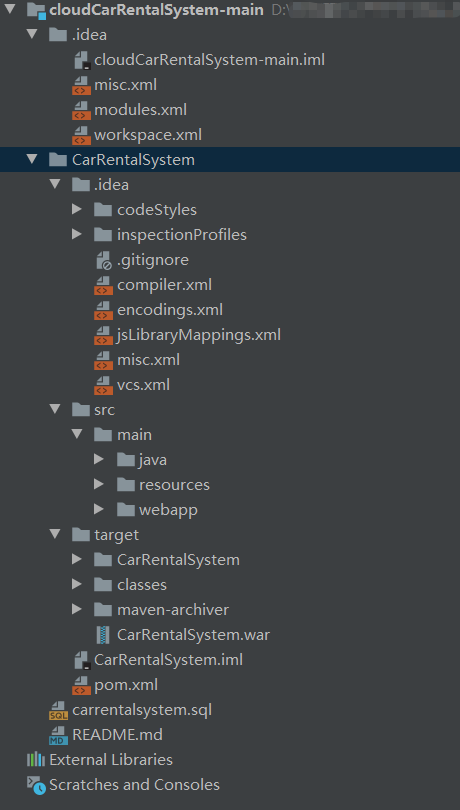


图37

# 第五章 项目实现

## 1.技术方案

**（1）使用Maven构建管理工具**

Maven主要用于解决导入依赖于Java类的jar和编译Java项目的主要问题。

依赖的jar包由pom.xml文件中的dependency属性管理，并且jar包包含类文件和一些必要的资源文件。在使用SSM过程中需要导入很多种包，包括数据库连接、Mybatis等包，使用Maven能很快得搭建好开发环境。

**（2）使用SSM框架（Spring+SpringMVC+Mybatis）**

SSM框架是spring MVC ，spring和mybatis框架的整合，是标准的MVC模式，将整个系统划分为表现层，controller层，service层，DAO层四层

使用spring MVC负责实现MVC设计模式。Spring负责管理Spring MVC和Mybatis相关对象的创建和依赖注入，mybatis作为数据对象的持久化引擎。

mybatis是对jdbc的封装，它让数据库底层操作变的透明。mybatis的操作都是围绕一个sqlSessionFactory实例展开的。mybatis通过配置文件关联到各实体类的Mapper文件，Mapper文件中配置了每个类对数据库所需进行的sql语句映射。在每次与数据库交互时，通过sqlSessionFactory拿到一个sqlSession，再执行sql命令。

**（3）Git和GitHub版本更新迭代**

所有的版本数据都保存在服务器上，协同开发者从服务器上同步更新或上传自己的修改。每一个开发人员是一个客户端，文件和版本信息存储服务器上面的。

## 2.关键模块的实现技术

**(1)路由寻址**

在页面访问路由方面，我们采用了SpringMVC代替了传统的Servlet，可以对不同角色的不同路由需求进行分类处理，前端使用thymleaf，后端使用SpringMVC，两者进行数据交互。例如前端使用thymleaf框架，给<a></a>一个链接th:href=”@{/example}”，则在相应的controller里，通过SpringMVC框架的注解@GetMapping(“/example”)即可捕获该路由请求，并进行处理。

<a id="hid" th:href="@{/example}"></a>

@GetMapping("/toLogin")

public String example() {

return "example";

}

**（2）ajax异步请求处理**

在登录和注册等功能模块下，为了带来更好的用户体验，使用ajax异步处理就很有效果。而jQuery已经为我们封装了ajax的相关功能，我们使用jQuery的ajax封装函数在前端发起请求，然后由后端接收数据并处理后返回给前端，前端根据返回的数据进行相关操作。

前端代码实例：

$(document).ready(function () {

$("#login").click(function () {

const user\_name = document.getElementById("user\_name").value;

const password = document.getElementById("password").value;

var encode = $.base64.encode(password);

const user\_type = document.getElementById("user\_type").value;

$.ajax({

url: '[[@{/login/toIndex}]]',

type: 'POST',

data: {

user\_name: user\_name,

password: encode,

user\_type: user\_type

},

async: true,

success: function (data) {

if (data == 'true') {

document.getElementById("hid").click();

} else {

alert("用户名或密码错误，请重新登录！");

}

}

})

})

});

后端代码实例：

//验证登录,与ajax交互

@RequestMapping("/toIndex")

@ResponseBody

public String toIndex(HttpServletRequest httpServletRequest, HttpSession httpSession) {

String user\_name = (String) httpServletRequest.getParameter("user\_name");

String user\_type = httpServletRequest.getParameter("user\_type");

String password = httpServletRequest.getParameter("password");

String name = httpServletRequest.getParameter("name");

//type记录用户类型，0为管理员，1为租车人，2为车主

int type;

if ("管理员".equals(user\_type)) {

type = 0;

} else if ("租车人".equals(user\_type)) {

type = 1;

} else {

type = 2;

}

System.out.println("解密前：" + user\_name + password + user\_type + name);

//对密码进行解密

password = BASE64Util.decode(password);

System.out.println("解密后：" + user\_name + password + user\_type + name);

//数据库进行查询该用户

Login one = loginService.findOne(user\_name);

System.out.println(one);

if (one != null && one.getUSER\_NAME().equals(user\_name) && one.getPASSWORD().equals(password) && one.getUSER\_TYPE() == type) {

//登录成功,在会话中进行标记

httpSession.setAttribute("user\_status", "1");

httpSession.setAttribute("user\_type", String.valueOf(type));

httpSession.setAttribute("user\_name", String.valueOf(user\_name));

//更新后台log

Login login = new Login();

login.setUSER\_NAME((String) httpSession.getAttribute("user\_name"));

login.setLAST\_TIME(new Date());

Login login1 = loginService.findOne((String) httpSession.getAttribute("user\_name"));

login.setTIMES(login1.getTIMES() + 1);

login.setSTATE(1);

loginService.updateLog(login);

return "true";

} else {

//登录失败

httpSession.setAttribute("user\_status", "0");

return "false";

}

}

**（3）安全加密**

如果登录注册时，使用明文传送密码，那么很容易会被他人从浏览器中拦截，为增加安全性，在前后端使用了BASE64进行加密解密（还有更好的加密方式，但为了演示并未采取），在前端传输前进行密码的加密，传送到后端后再进行解密，从而加大密码破译的难度。

前端：

var encode = $.base64.encode(password);

后端：

password = BASE64Util.decode(password); // 已有一个BASE64Util类

**（4）**使用ModelAndView规范模型数据和视图相关信息，Model是指需要传递的模型数据，View是指将要展示的页面，使用它可以使得代码易读性更强，功能也更加强大。传输数据的形式也可多样化，可以传递对象、List列表等等。

实例：

//返回所有订单信息

@GetMapping("/all\_order")

public ModelAndView all\_order(HttpSession httpSession) {

ModelAndView modelAndView = new ModelAndView();

if ("1".equals(httpSession.getAttribute("user\_status"))) {

List<Order> orderList = adminService.getAllOrders();

// 添加模型数据

modelAndView.addObject("orderList", orderList);  
 // 设定跳转页面

modelAndView.setViewName("orderList");

} else {

modelAndView.setViewName("login");

}

return modelAndView;

}

## 3.特殊问题及解决办法

**1.** 登录验证时，如果登录失败需要重新返回页面重新填写登录信息，显得繁琐且用户体验差。

**解决方案：**结合jQuery和Ajax实现异步登录验证，当提交表单后，后端进行登录验证并将结果返回前端，前端接收到消息后做出反映，同时会保存用户名和密码，以便用户修改。

**解决结果**：问题解决。当登录时，会在不刷新页面的情况下给出反馈。

**2.** jsp太过古老，而使用html时又无法像jsp一样和后端进行直接交互且静态资源引用也会报错。

**解决方案：**使用Thymeleaf把model中的数据渲染到html中进行显示，除此之外，还可以通过Thymeleaf引用静态资源（css/js/img）、实现路由的访问等功能。

**解决结果：**问题解决。使用Thymeleaf后可以和后端SpringMVC很好地结合起来实现路由的跳转寻址等功能，代码清晰，排查错误也更方便。

**3.** 用户注册和登录时，需要输入密码，而如果使用明文直接传输，很容易从浏览器中查看到传输的密码，系统安全性大大降低。

**解决方案：**使用基于Base64的通用加密解密算法，在前端传输前进行密码串的加密处理，然后传输到后端controller，再由创建的工具类进行解密，再进行数据库比对。（此方法依然有安全隐患，可以使用md5等方式将密码加密后存储在数据库，更为安全，但为了便于演示，这里不采用密文存储的方式）

**解决结果：**问题解决。在浏览器后台截取密码，会发现密码是被加密过的数据，不采取一定算法难以得到真正的密码。

**4.** 留言系统中，如果留言内容过长，则会影响页面布局或是不能完整查看消息，带来较差的用户体验。

**解决方案：**固定表格的长度比例，超出的文字部分进行隐藏，在该行留言的末尾加上“查看”按钮，点击后可出现一个模态框显示完整的留言消息，且查看过后，会刷新后台的留言状态。

**解决结果：**问题解决。当文字过多时，会自动隐藏超出的部分，点击查看后，会显示完整的信息。

**5.** 数据库导出后，再导入会发生未知错误，无法完全导入数据。

解决方案：当时使用的是Navicat Premium 15作为数据库客户端，直接导出会存在一些过多的版本约束，导致再次导入会出错；之后查阅资料后采用mysql命令行进行导出，再导入就恢复正常了。

解决结果：问题解决。

**6.** 项目使用Maven打包后，却无法使用tomcat环境直接运行。

**解决方案：**经过排查，发现pom.xml中，打包的选项中，只打包了java相关文件，而并没有打包src/main/resources文件夹下的文件，也就不包含ssm相关的配置文件。重新设置后，再次打包，即可运行成功。

**解决结果：**问题解决。项目的.war包包含了所有运行所需的包和其他资源，可在tomcat下完美运行。

# 第六章 测试计划

## 1.引言

### 1.1 项目背景

说到租车，对于很多人来说想必都不陌生，目前的租车模式不论是生活上、工作上都有着越来越多的政府机构、企事业单位、商务人士及个人都可以享受汽车租赁带来的便利；当下汽车租赁行业进入了规模化、高速发展时期。 汽车租赁行业的资源共享、绿色、低碳的出行方式所带来的行业优势已经显现，汽车租赁行业将保持高速增长，线上租车APP开发也应运而生，优化用户的租车体验。

本项目为线上云租车平台，为车辆空闲的人和需要临时用车的人提供便利，两者将直接进行交易，平台在其中起到维护和保障双方利益的作用，同时也收取一定的服务费。根据对租车市场现状的分析，一个合理可行的租车平台需要车主发布车辆相关信息，如型号、价格、期限、新旧等，租车人可以在平台上看到车主们发布的车辆相关信息，进行租车服务，同时平台管理员起到连接车主和租车人的作用，对车主发布车辆的相关信息进行核实验证，同时对租车人的信用进行考量，设定租车期限，对于租车过程和还车过程中发生的纠纷进行处理。

### 1.2 测试目的

编写本测试计划的目的是：

（1）为整个测试阶段的管理工作和技术工作提供指南同时确定测试的内容和范围，为评价系统提供依据；

（2）开发有效的测试模型，能正确地验证正在开发的软件系统并且确立每个测试阶段测试完成以及测试成功的标准、要实现的目标；

（3）识别出测试活动中各种风险，并消除可能存在的风险，降低由不可能消除的风险所带来的损失。

预期读者对象为测试人员以及软件开发人员。

### 1.3 参考资料

列出编写本计划及测试整个过程中所要参考的文件、资料。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **资料名称** | **作者** | **日期** | **出版单位** |
| 1 | 《软件测试技术》 | 刘文乐 田秋成 | 2012 | 机械工业出版社 |

## 2.任务概述

### 2.1 目标

本测试覆盖范围：

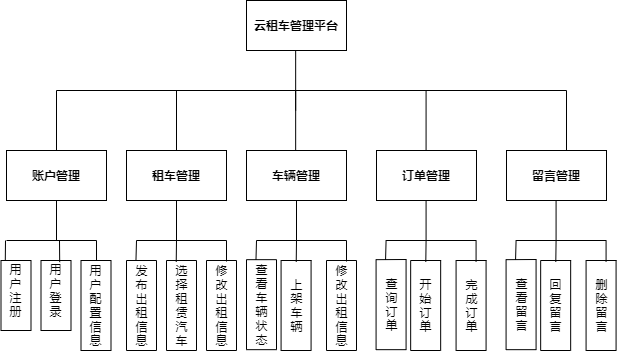


图38-平台功能模块示意图

通过测试，达到以下目标：

（1）测试已实现的项目达到设计的要求。

（2）项目规定的操作和运行稳定。

（3）Bug数和缺陷率控制在可接受的范围之内。

本项目的验收标准：

执行完所有系统的功能测试、性能测试的用例，无重大的导致系统不能运行的问题。如果进行系统测试时，存在严重的质量问题，导致无法继续，并且在可接受的时间范围内无法修复，系统测试终止。

### 2.2 测试环境

1. 硬件环境
2. 操作系统： Microsoft Windows 10

（2）软件环境

服务器：Tomcat 8.5

数据库：MYSQL 2021

开发软件：IDEA 2019

### 2.3 需求概述

#### 2.3.1 数据需求

（1）用户登录信息表，表名为LOGIN，数据设计详情如下：

CREATE TABLE `login` (

`USER\_NAME` varchar(60) NOT NULL,

`PASSWORD` varchar(60) NOT NULL default '',

`LAST\_TIME` datetime default NULL,

`TIMES` int(11) NOT NULL,

`STATE` int(11) NOT NULL,

`REGISTER\_TIME` datetime NOT NULL,

`USER\_TYPE` int(11) NOT NULL,

PRIMARY KEY (`USER\_NAME`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（2）车主详细信息主表，表名为OWNER\_OF\_VEHICLE，数据设计详情如下：

CREATE TABLE `owner\_of\_vehicle` (

`USER\_NAME` varchar(60) NOT NULL,

`NAME` varchar(30) NOT NULL,

`SEX` int(11) NOT NULL,

`ID` varchar(30) NOT NULL,

`CREDIT` int(11) NOT NULL,

`FREQUENCY` int(11) NOT NULL,

`TEL` varchar(30) NOT NULL,

`EMAIL` varchar(60) NOT NULL default '',

`INCOME` double NOT NULL,

PRIMARY KEY (`USER\_NAME`),

CONSTRAINT `fk\_3` FOREIGN KEY (`USER\_NAME`) REFERENCES `login` (`USER\_NAME`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（3）租车人详细信息主表，表名为HIRER\_OF\_VEHICLE，数据设计详情如下：

CREATE TABLE `hirer\_of\_vehicle` (

`USER\_NAME` varchar(60) NOT NULL,

`NAME` varchar(30) NOT NULL,

`SEX` int(11) NOT NULL,

`ID` varchar(30) NOT NULL,

`CREDIT` int(11) NOT NULL,

`FREQUENCY` int(11) NOT NULL,

`EMAIL` varchar(60) NOT NULL default '',

`TEL` varchar(30) NOT NULL,

PRIMARY KEY (`USER\_NAME`),

CONSTRAINT `fk\_2` FOREIGN KEY (`USER\_NAME`) REFERENCES `login` (`USER\_NAME`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（4）管理员详细信息主表，表名为ADMIN，数据库设计详情如下：

CREATE TABLE `admin` (

`USER\_NAME` varchar(60) NOT NULL,

`NAME` varchar(30) NOT NULL,

`ID` varchar(30) NOT NULL,

`ADMIN\_ID` varchar(30) NOT NULL,

PRIMARY KEY (`USER\_NAME`),

CONSTRAINT `fk\_1` FOREIGN KEY (`USER\_NAME`) REFERENCES `login` (`USER\_NAME`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（5）车辆信息详表，表名为VEHICLE，数据设计详情如下：

CREATE TABLE `vehicle` (

`VEHICLE\_ID` varchar(30) NOT NULL default '',

`VEHICLE\_TYPE` varchar(60) NOT NULL,

`OWNER\_ID` varchar(60) NOT NULL,

`DURATION` int(11) NOT NULL,

`STATUS` int(11) NOT NULL,

`RENTAL\_PRICE` double NOT NULL,

`DEADLINE` datetime default NULL,

`LOCATION` varchar(255) NOT NULL,

`VEHICLE\_BRAND` varchar(255) NOT NULL,

PRIMARY KEY (`VEHICLE\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（6）交易订单表，表名为ORDER，数据库设计详情如下：

CREATE TABLE `orders` (

`ORDER\_ID` int(11) NOT NULL auto\_increment,

`OWNER\_NAME` varchar(30) NOT NULL,

`HIRER\_NAME` varchar(30) NOT NULL,

`START\_TIME` datetime default NULL,

`END\_TIME` datetime default NULL,

`VEHICLE\_TYPE` varchar(60) NOT NULL,

`STATUS` int(11) NOT NULL,

`AMOUNT` double default NULL,

`VEHICLE\_ID` varchar(30) NOT NULL,

`VEHICLE\_BRAND` varchar(255) NOT NULL,

PRIMARY KEY (`ORDER\_ID`),

KEY `fk\_4\_1` (`OWNER\_NAME`),

KEY `fk\_4\_2` (`HIRER\_NAME`),

CONSTRAINT `fk\_4\_1` FOREIGN KEY (`OWNER\_NAME`) REFERENCES `login` (`USER\_NAME`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `fk\_4\_2` FOREIGN KEY (`HIRER\_NAME`) REFERENCES `login` (`USER\_NAME`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

（7）留言表，表名为MESSAGE，数据库设计详情如下：

CREATE TABLE `message` (

`MESSAGE\_ID` int(11) NOT NULL auto\_increment,

`SEND\_NAME` varchar(60) NOT NULL,

`RECEIVE\_NAME` varchar(60) NOT NULL,

`DETAIL` varchar(255) NOT NULL,

`SEND\_TIME` datetime NOT NULL,

`STATUS` int(11) NOT NULL,

PRIMARY KEY (`MESSAGE\_ID`),

KEY `fk\_5` (`SEND\_NAME`),

KEY `fk\_5\_2` (`RECEIVE\_NAME`),

CONSTRAINT `fk\_5` FOREIGN KEY (`SEND\_NAME`) REFERENCES `login` (`USER\_NAME`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `fk\_5\_2` FOREIGN KEY (`RECEIVE\_NAME`) REFERENCES `login` (`USER\_NAME`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

本项目的内部数据均需按照上述数据设计进行组织。

#### 2.3.2 事务需求

|  |  |
| --- | --- |
| 步骤 | 动作 |
| 1 | 将测试用例进行打包和编译。 |
| 2 | 对测试用例进行审核，并最终提交测试。 |
| 3 | 接收测试。 |
| 4 | 开始测试。 |

测试过程和处理要求按以上步骤进行。

### 2.4 条件和限制

1. 硬软件设备固定
2. 测试时间有限

测试开始时间：2021-12-20 10:00am

测试结束时间：2021-12-30 10:00am

## 3.测试计划

### 3.1 测试方法

1. 数据和数据库完整性测试

将数据库和数据库进程应作为一个子系统来进行测试，确保数据库访问方法和进程正常运行，数据不会遭到损坏；检查数据库，确保数据已按预期的方式填充，并且所有的数据库事件已正常发生；或者检查所返回的数据，确保正当的理由检索到了正确的数据。

1. 接口测试

所有软件、硬件接口，记录输入输出数据，确保接口调用的正确性。

1. 功能测试

利用有效的和无效的数据来执行各个用例、用例流或功能，核实是否在使用有效数据时得到预期的结果；在使用无效数据时显示相应的错误消息或警告消息，确保测试的功能正常。

1. 用户界面测试

通过测试进行的浏览可正确反映项目的功能和需求，包括窗口与窗口之间、字段与字段之间的浏览，以及使用窗口的菜单、大小、位置、状态符合标准。确保用户界面会通过测试对象的功能来为用户提供相应的访问或浏览功能。

1. 性能测试

对响应时间、事务处理速率和其他与时间相关的需求进行评测和评估。

### 3.2 测试手段

（1）手工测试

由测试人员逐个输入用例，观察结果。

1. 自动测试

由测试人员使用脚本，代码和自动化工具（计算机）完成。

### 3.3 测试内容

#### 3.3.1 账户管理

在云租车系统中，账户管理模块主要负责对系统账户的注册、登录以及信息配置进行管理。系统账户管理类图如下所示，从图中可以看出此业务功能模块的详细参与人员，具体业务类包括车主、租车人、平台管理员。

#### 3.3.2 租车管理

用户租车与被租车都需要在租车管理系统中进行，车主需要发布车辆出租信息，并拥有出租信息修改、收益查询、问题申诉等功能，而租车人有选择租赁汽车、问题申诉等主要功能。

#### 3.3.3车辆管理

车辆拥有多项基本属性，包括车主用户名、车牌号、已租时长、状态、出租价格、取车地点等，车辆是本系统的重要组成部分，车主是车辆的拥有者，可以发布车辆信息；租车人是车辆的租赁者，可以选择租赁某辆在租的汽车。而对于车辆管理，主要由车主和管理员负责。

#### 3.3.4订单管理

订单拥有多项属性信息，包括订单号、车主用户名、租车人用户名、起始时间、完成时间、车型、车牌号、结算金额。订单是车主和租车人进行成功交易的凭证，也是网站流水的重要数据，平台的管理员、车主、租车人都对其有一定权限。

#### 3.3.5留言管理

拥有多项属性信息，包括留言id、发送方用户名、接收方用户名、留言内容、状态等，该模块主要是作为车主、租车人、平台管理员进行交流的媒介，三者通过这一类可以很容易地获取他人给自己的留言并选择进行回复或删除。

### 3.4 测试准备

在测试前，与各模块的主要负责人进行共同协商讨论；阅读软件需求分析说明书，并以此作为总的提纲；选择合适的输入输出数据；编写测试用例。

### 3.5 测试机构以及人员

项目测试组测试人员以及部分开发人员。

## 4.测试项目

### 4.1 测试用例

**Test\_001:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_001 |
| 测试标题 | 登录功能（管理员、租车人、车主） |
| 重要级别 | 高 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 转向对应的系统界面 |
| 完成情况 | 已完成 |
| 操作步骤 | 分别输入正确对应的账户信息/分别输入错误对应的账户信息，选择用户类型，点击登录按键 |
| 实际输入 | 账号以及密码 |
| 实际输出 | 转向对应的系统界面 |
| 评价 | 符合要求 |



图39

**Test\_002:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_002 |
| 测试标题 | 忘记密码功能（管理员、租车人、车主） |
| 重要级别 | 中 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 转向对应的重置密码界面，重置密码更新至数据库 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击忘记密码按键链接，输入邮箱、用户名、新密码，点击重置密码按键 |
| 实际输入 | 邮箱、用户名、新密码 |
| 实际输出 | 转向对应的重置密码界面，重置密码更新至数据库 |
| 评价 | 符合要求 |



图40

**Test\_003:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_003 |
| 测试标题 | 注册功能（管理员、租车人、车主） |
| 重要级别 | 高 |
| 预制条件 | 无 |
| 预期输出 | 更新信息至数据库 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击注册按键，输入用户名，姓名，性别，身份证号，邮箱，手机号码，用户类别，密码，点击注册按键 |
| 实际输入 | 用户名，姓名，性别，身份证号，邮箱，手机号码，用户类别，密码 |
| 实际输出 | 更新信息至数据库 |
| 评价 | 符合要求 |



图41

**Test\_004:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_004 |
| 测试标题 | 信息查看（管理员） |
| 重要级别 | 高 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 可视化的信息查看,转向对应的系统界面 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击查看个人信息/历史登录/管理员/我的留言,滑动对话框进行浏览 |
| 实际输入 | 无 |
| 实际输出 | 可视化的信息查看,转向对应的系统界面 |
| 评价 | 符合要求 |

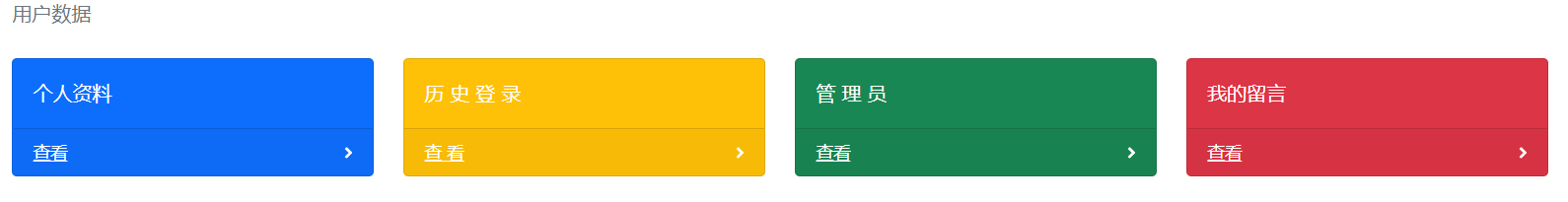


图42



图43

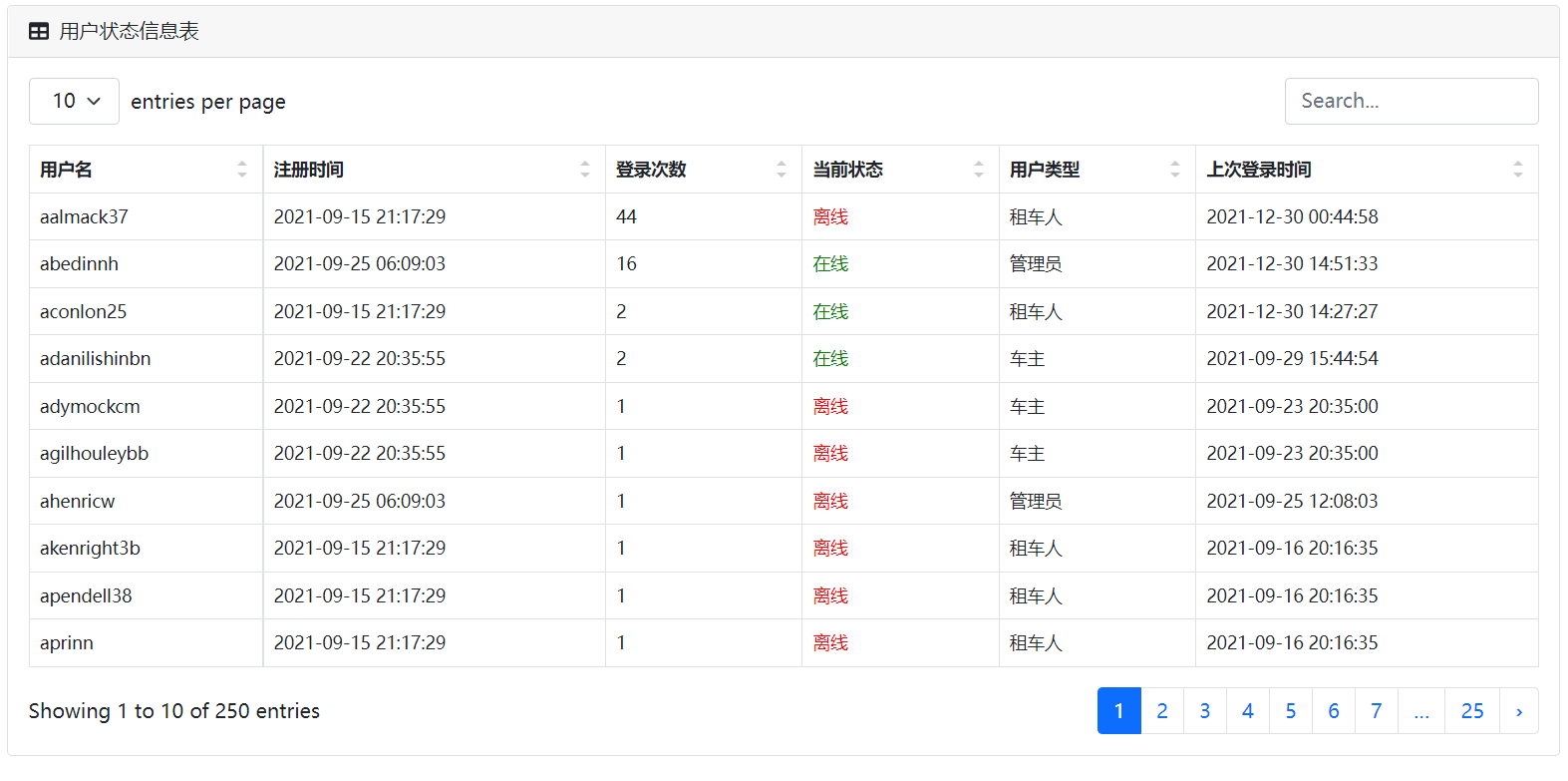


图44

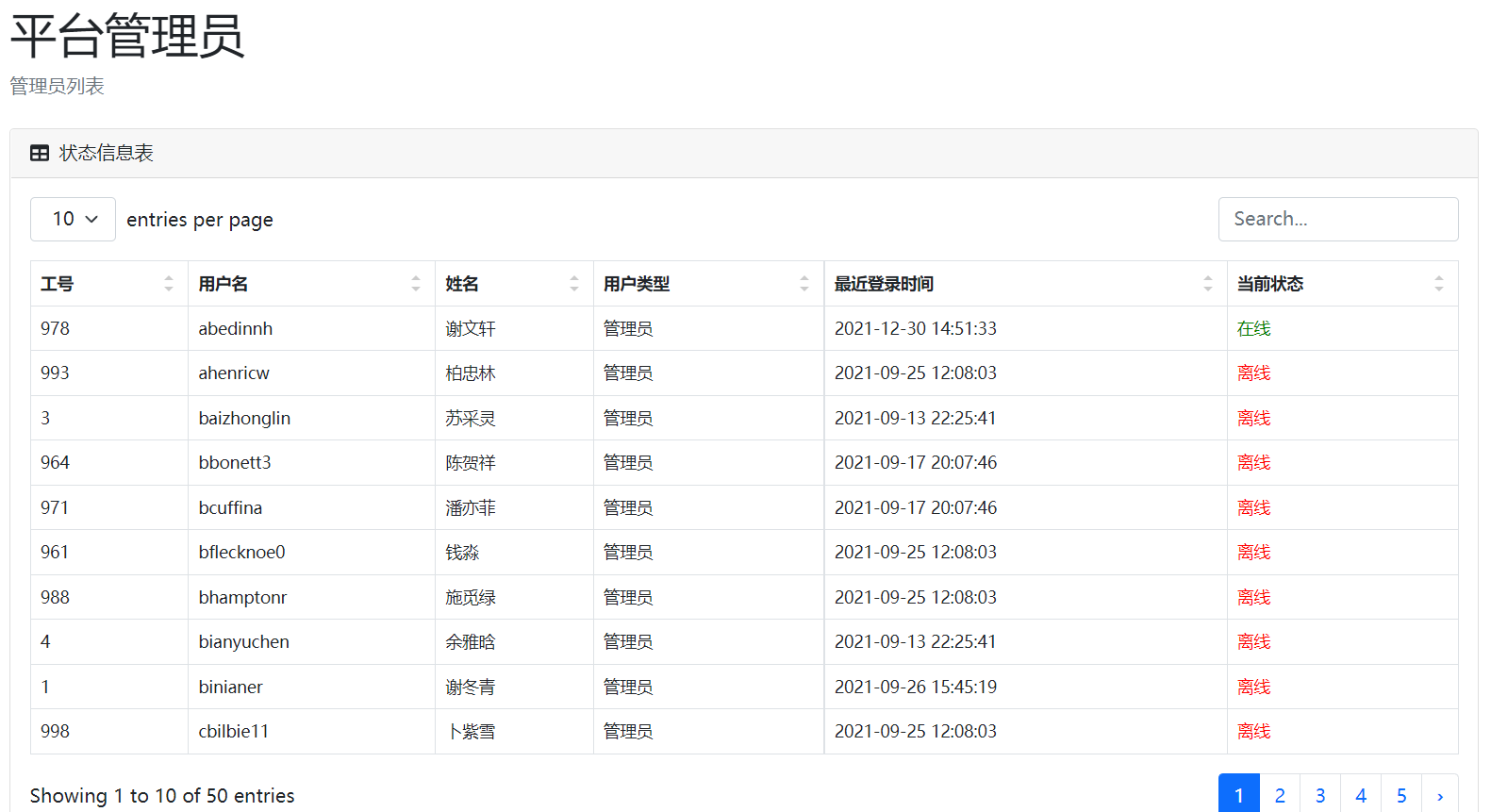


图45



图46

**Test\_005:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_005 |
| 测试标题 | 柱状统计图的更新与显示（管理员） |
| 重要级别 | 中 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 可视化的信息查看，柱状统计图按照数据库中的数据完成更新与显示 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击查看，刷新后进行更新 |
| 实际输入 | 无 |
| 实际输出 | 可视化的信息查看，柱状统计图按照数据库中的数据完成更新与显示 |
| 评价 | 符合要求 |

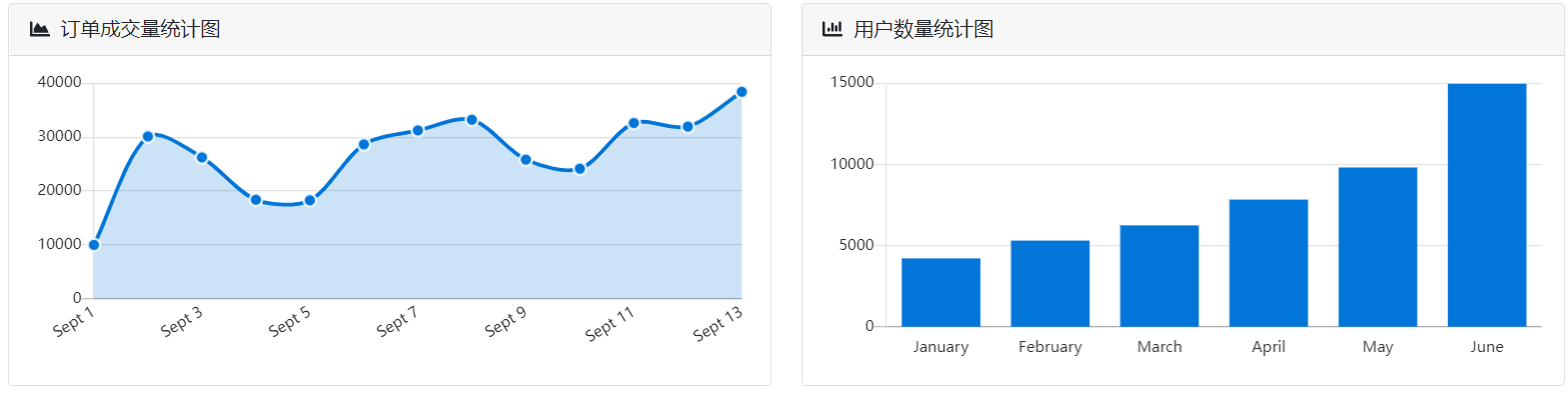


图47

**Test\_006:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_006 |
| 测试标题 | 用户管理以及黑名单（管理员） |
| 重要级别 | 高 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 用户的相关信息显示以及数据库的相关信息更新，转向对应的系统界面 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击用户管理，选择具体的用户类型（租车人或者车主），查看详细信息，选择其中用户点击加入黑名单的按键，跳转到黑名单信息，在黑名单页点击移除黑名单按键。 |
| 实际输入 | 无 |
| 实际输出 | 用户的相关信息显示以及数据库的相关信息更新，转向对应的系统界面 |
| 评价 | 符合要求 |

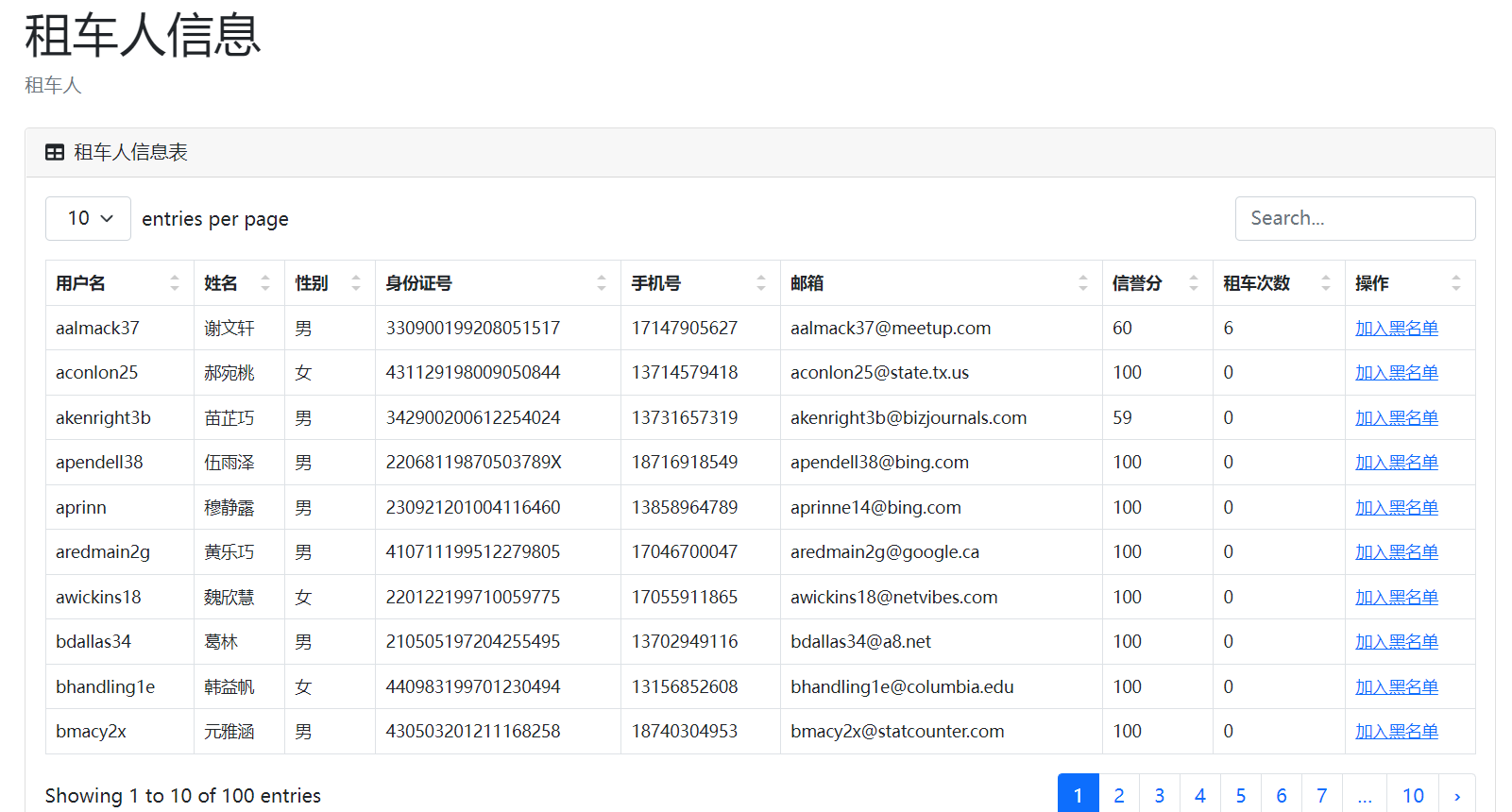


图48

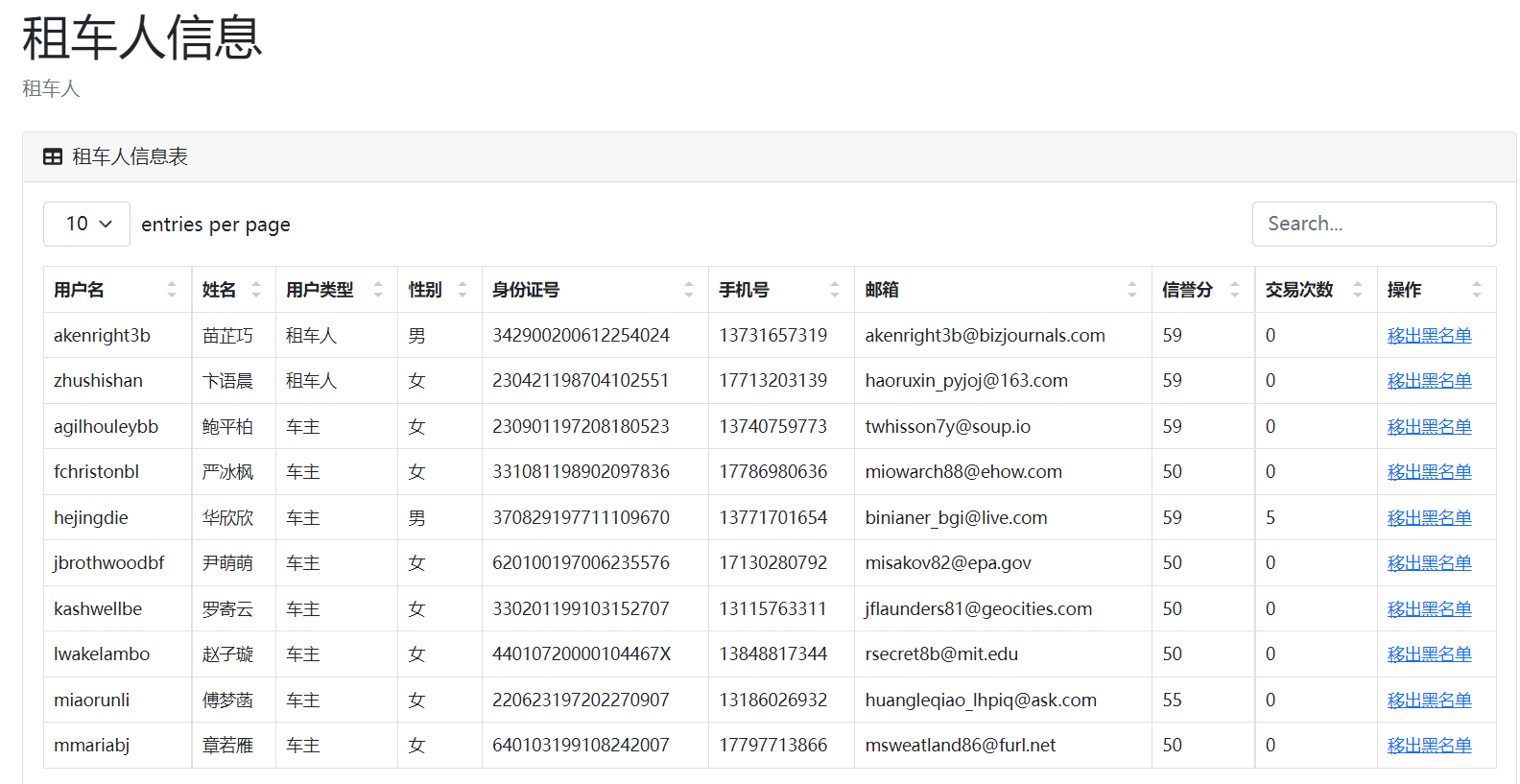


图49

**Test\_007:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_007 |
| 测试标题 | 车辆管理以及上下架（管理员） |
| 重要级别 | 高 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 车辆的相关信息显示以及数据库的相关信息更新，转向对应的系统界面 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击车辆管理，选择具体的车辆类型（已租，待租，已下架），查看信息/点击车辆上下架，点击上架或者下架按键。 |
| 实际输入 | 无 |
| 实际输出 | 车辆的相关信息显示以及数据库的相关信息更新，转向对应的系统界面 |
| 评价 | 符合要求 |

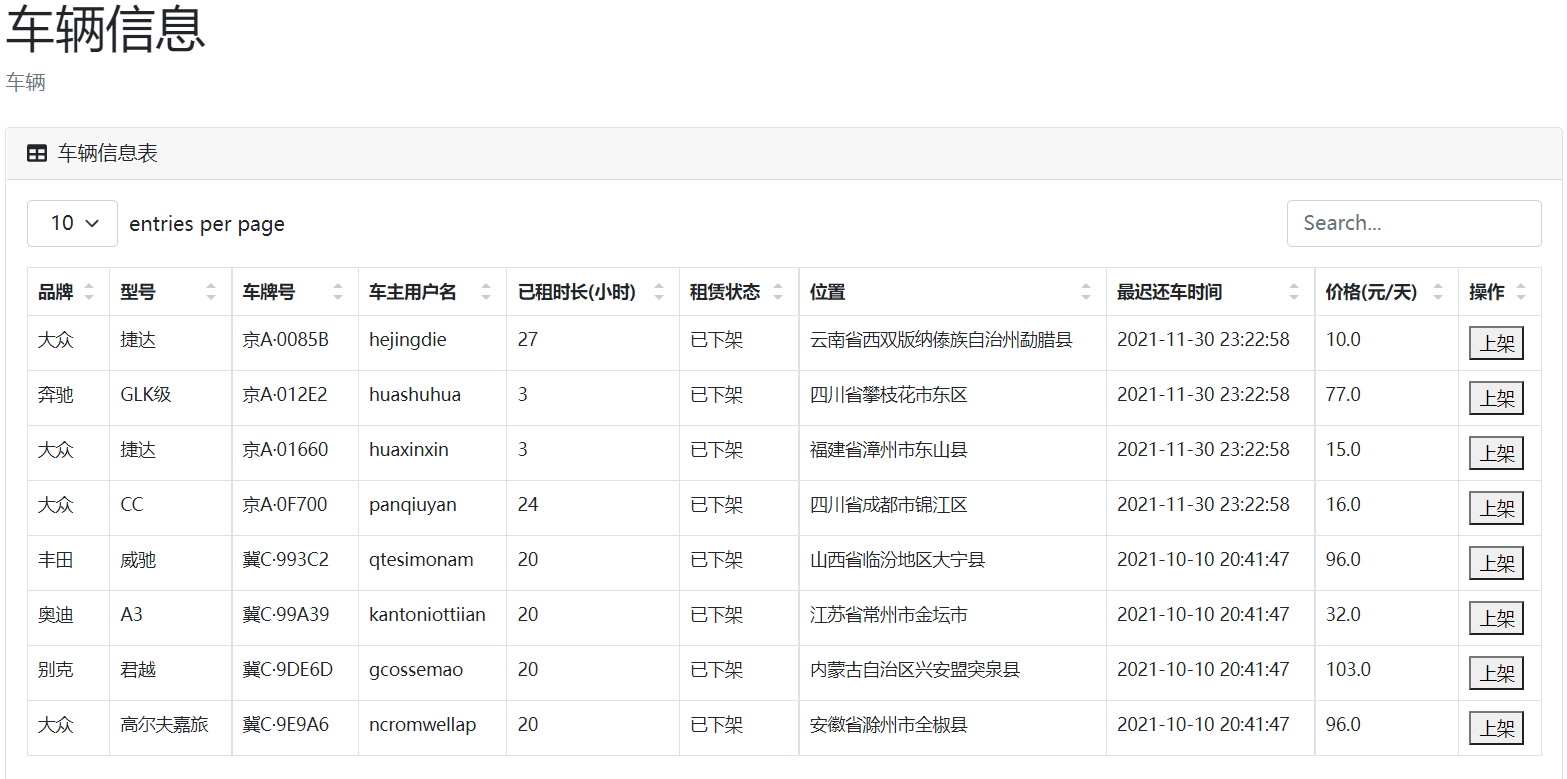


图50

**Test\_008:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_008 |
| 测试标题 | 留言（管理员，租车人，车主） |
| 重要级别 | 中 |
| 预制条件 | 无 |
| 预期输出 | 数据库的相关信息更新，转向对应的系统界面 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击留言，输入留言内容，点击发送留言或者放弃发送按键。 |
| 实际输入 | 留言内容 |
| 实际输出 | 数据库的相关信息更新，转向对应的系统界面 |
| 评价 | 符合要求 |



图51

**Test\_009:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_009 |
| 测试标题 | 租车功能（租车人） |
| 重要级别 | 高 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 数据库的相关信息更新，转向对应的系统界面 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击租车按键并且确认。 |
| 实际输入 | 无 |
| 实际输出 | 数据库的相关信息更新，转向对应的系统界面 |
| 评价 | 符合要求 |

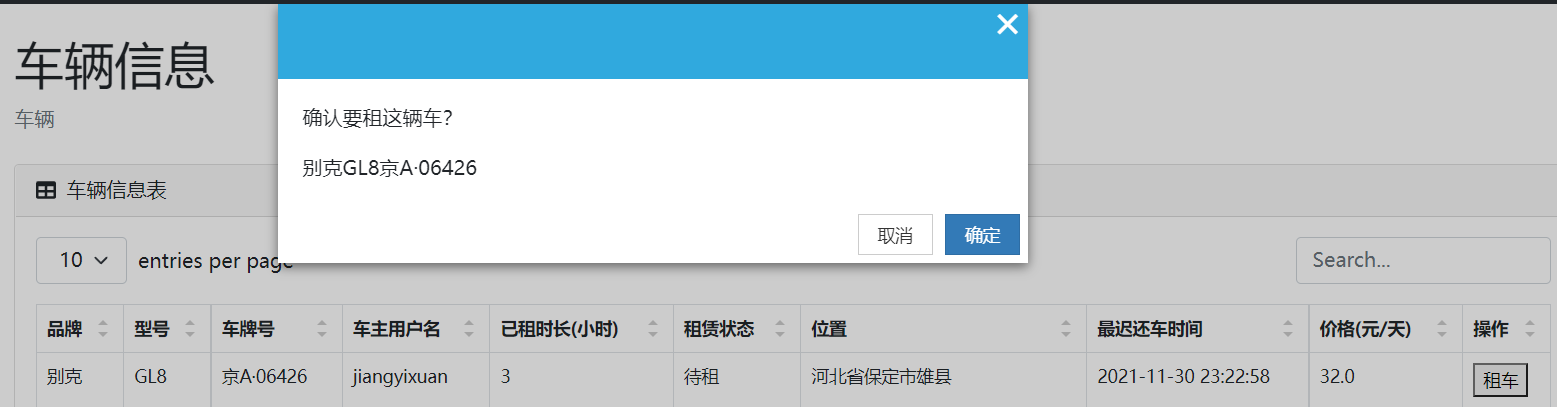


图52

**Test\_010:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_010 |
| 测试标题 | 租车，还车，车辆上下架，新增车辆功能（车主） |
| 重要级别 | 高 |
| 预制条件 | 数据库已存在相应账户及信息 |
| 预期输出 | 数据库的相关信息更新，转向对应的系统界面 |
| 完成情况 | 已完成 |
| 操作步骤 | 点击按键并且确认/输入车牌号，厂商，车型，时间，单价，所在地，点击提交按键 |
| 实际输入 | 车牌号，厂商，车型，时间，单价，所在地 |
| 实际输出 | 数据库的相关信息更新，转向对应的系统界面 |
| 评价 | 符合要求 |

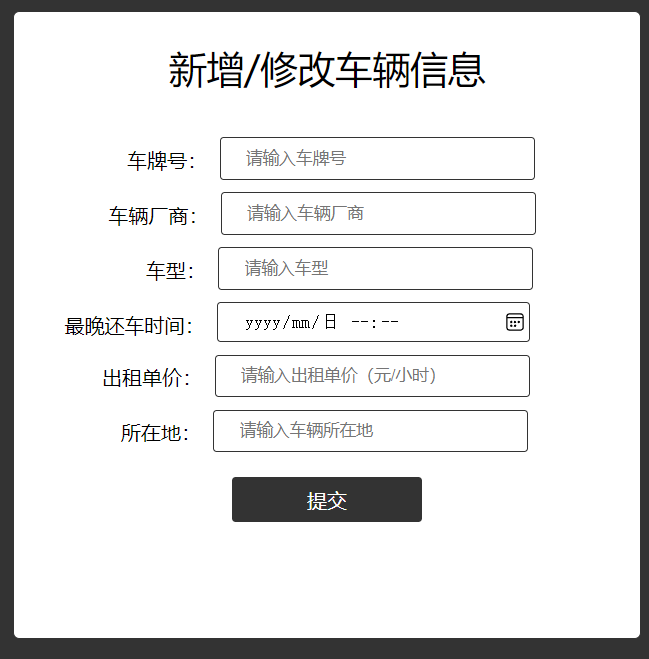


图53

**Test\_011:**

|  |  |
| --- | --- |
| 测试用例编号 | Test\_011 |
| 测试标题 | 时间性能测试 |
| 重要级别 | 中 |
| 预制条件 | 无 |
| 预期输出 | 无 |
| 完成情况 | 已完成 |
| 操作步骤 | 进行切换界面，修改信息等相关操作，记录相关时间延迟 |
| 实际输入 | 无 |
| 实际输出 | 无 |
| 评价 | 时间延迟在一秒以内，符合要求 |

## 5.评价

### 5.1 准则

质量准则：错误率尽可能低，效率尽可能高，具有可靠性。

覆盖准则：用例的覆盖度要高。

### 5.2 结果

结果偏差在允许范围内，测试通过。