软件工程实验报告

实验名称：停车场管理系统文档撰写

项目名称：停车场管理系统

项目组成员：

姓名	学号	角色	工作内容	评分
蒋芷昕	2017180202005		组织策划、文档撰写	A
冯朗	2017060102022		文档撰写、代码编写	B

注：评分共分为 5 档：A、B、C、D、E，每组中评分须拉开差距，要求每个档次都有。

项 目 编 号	200602006
文 档 编 号	1 0
密 级	内部

停车场管理系统需求规格

V1.0

软件工程实验

评 审 日 期: 2019 年 10 月 10 日

目 录

1 导言

1.1 目的

该文档是关于用户对于停车场管理系统的功能和性能的要求，重点描述了停车场管理系统的设计需求，将作为对该工具在概要设计阶段的设计输入。

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，解决整个项目系统的“做什么”的问题。在这里，对于开发技术并没有涉及，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渠道。

1.3 缩写说明

UML

Unified Modeling Language（统一建模语言）的缩写。

1.4 术语定义

Use Case

用例。一种通过用户的使用场景来获取需求的技术。

1.5 引用标准

[1] 《企业文档格式标准》 V1.1
北京长江软件有限公司
[2] 《需求规格报告格式标准》 V1.1
北京长江软件有限公司软件工程过程化组织

1.6 参考资料

[1] 《UML》 V1.1
北京长江软件有限公司
[2] 《需求规格报告格式标准》 V1.1
北京长江软件有限公司软件工程过程化组织

1.7 版本更新信息

本文档的更新记录如表 A-1。

表 A-1 版本更新记录

修改编号	修改日期	修改后版本	修改位置	修改内容概述
001	2019.10.5	0.1	全部	初始发布版本
002	2019.11.10	0.2	4.3 章节	修改
003	2019.11.15	1.0	4.4 章节	修改

2 系统定义

我们分别阐述一下项目的来源、背景和项目的目标。

2.1 项目来源及背景

本项目是为停车场管理者以及需要泊车的用户开发的一个简易停车场管理系统，由于在日常生活中，人们因为外出需求常常需要将车停泊在各类中大型停车场中。对于需要泊车的车主而言，如何简明快速地找到最近的停车位，以及在停车结束前，快速地在路线复杂的停车场中找寻到自己的车成为较为棘手的问题；对于停车场管理者而言，还需进行车位管理以应对突发事件（例如因装修原因需要占用某些车位而不向普通车主提供泊车服务），以及泊车费用、时长、车牌等相应信息的记录。为此我们希望有一个智能化的停车场管理系统能够给用户提供便利，提高对停车场的使用满意度，以及提高管理者的工作效率。

为现实协调管理者与用户的关系，以及充分发挥停车场作用的目的，来弥补车位管理中存在的种种不足。在这种条件下，我们开发了停车场管理系统。车库使用者可以通过系统在入场前查看当前车位余量，离场前查看泊车费用与时长信息，并同时获去取前往泊车位的最优路径。停车场内的收费人员通过这个管理系统可以根据离场车辆的时长收取相应的泊车费用。管理停车场的工作人员通过这个管理系统可以实时更新发布当前停车场的信息，并随时查看停车场状态；根据需求，查看特定的泊车车辆信息；统计整理以往某一时间段的总体泊车情况，并根据停车场总体的使用率进行相应的运营调整。

2.2 项目要达到的目标

本项目设定的目标如下：

1. 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
2. 系统具有良好的运行效率，能够得到提高生产率的目的
3. 系统应有良好的可扩充性，可以容易的加入其它系统的应用。
4. 平台的设计具有一定的超前性，灵活性，能够适应企业生产配置的变化。
5. 通过这个项目可以锻炼队伍，提高团队的开发能力和项目管理能力

2.3 系统整体结构

根据观察日常生活中的需求，可以确定本项目分为客户端和管理端，客户端主要功能是提供停车场的余量显示、检索车辆的泊车时长、费用，前往泊车位的最优路径等。管理端的功能提供停车场管理人员进行的停车场状态管理、车位管理、车辆管理等。他们的关系如图A-1。

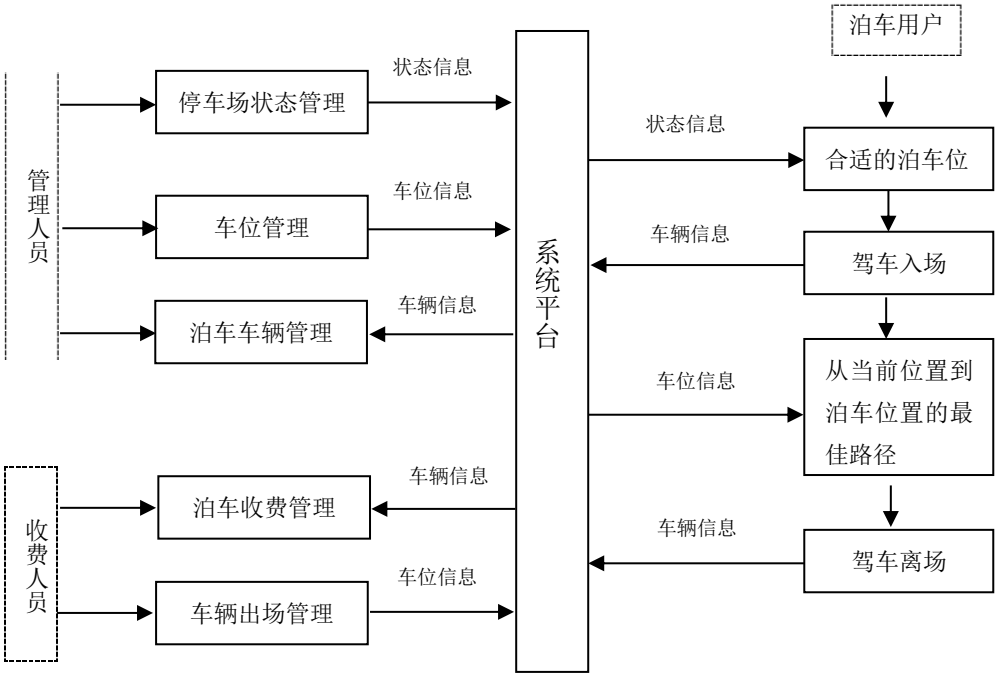


图 A-1：停车场管理系统流程图

3 应用环境

本项目的应用环境可以分硬件环境、软件环境和网络环境来描述。

3.1 系统运行网络环境

本系统的网络运行图如图 A-2，无论是客户端的泊车用户还是管理端的管理人员等都可以通过网络登录到本系统中。泊车用户通过网络查看停车场相关信息，管理人员通过网络管理停车场，获得相关车辆及车位状态信息，进行管理。

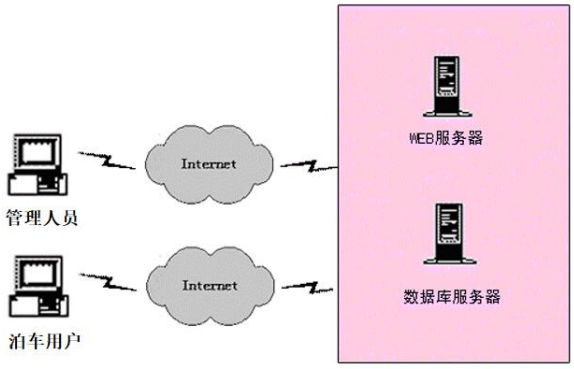


图 A-2：网络拓扑图

3.2 系统运行硬件环境

本系统的硬件环境如下：

- 客户机：普通 PC
  - CPU：P4 1.8GHz
  - 内存：256MB 以上
  - 分辨率：推荐使用 1024*768 像素
- WEB 服务器
  - CPU：P4 1.8GHz
  - 内存：256MB 以上
- 数据库服务器
  - CPU：P4 1.8GHz
  - 内存：256MB 以上

3.3 系统运行软件环境

- 操作系统：Windows 10
- 数据库：SQL Server 2017
- 开发工具：JDK Version 1.4.2
- JSP 服务器：Tomcat
- 浏览器：IE6.0

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 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。

Use Case 描述的是“actor”(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。Use Case 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 Use-case 时，我们遵循下列步骤：

第一步，识别出系统的“Actor”。Actor 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(Actor)是谁。尽可能地确保所有 Actor 都被完全识别出来。

第二步，描述主要的 Use Case。可以采取不断提问“这个 Actor 究竟想通过系统做什么？”的方式，来准确地描述 Use Case。

第三步，重新审视每个 Use Case，为它们下个详尽的定义。

4.1 角色（Actor）定义

角色或者执行者（Actor）指与系统产生交互的外部用户或者外部系统。

4.1.1 管理用户

管理用户是指管理端的用户，这个此 Actor 派生三个子类，停车场管理者、收费人员和系统管理员。停车场管理者是指在这个停车场管理系统中通过服务端管理泊车信息的人员，作为整个停车场系统的统筹管理者，进行远程的监控和一些重要信息的决策，主要参与停车系统车位管理、车辆信息查询、泊车管理等功能。收费人员是指对每一次入场、泊车、离场等活动进行具体的管控，他们是对出入停车场系统的每一位用户进行相关收费，导引的人员。系统管理员是指对停车场管理系统进行相关设置、维护的人员，它也是通过管理端登录对管理端的用户进行设置，分配权限等。它们的关系如图 A-3：

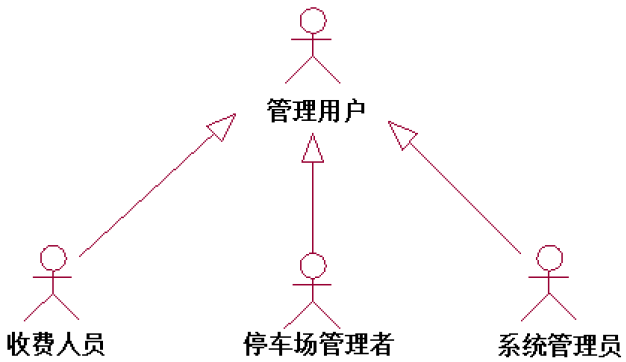


图 A-3：管理用户的关系图

管理用户具体说明如下：

- 停车场管理者
  —管理车位、车辆信息。
- 收费人员
  —根据停车场数据库记录信息核实准备出场的车辆是否成功缴费，并更改车辆的出场状态，结束本次泊车，管理停车场的现场引导活动。
- 系统管理员
  —通过管理端对系统用户进行管理的人员，这个 Actor 主要负责对管理端用户的增加，权限的设置等功能。

4.1.2 泊车人员

泊车人员是指具体的使用本停车场管理系统，出入停车场并进行相关行动的用户。他们作为该系统面向的真正服务者，需要知道该停车系统是否有空位，泊车时长和产生的费用，以及最优离场路径等信息。

4.2 系统主Use Case图

停车场管理系统可以分为两个主要的组成部分：一个是客户端子系统，一个是管理端子系统。客户端子系统主要是指泊车用户通过登录停车场管理系统进行相关操作的功能，即泊车功能。管理端子系统是停车场的管理人员运行查看车位及车辆信息，进行停车收费，修改停车场状态等功能。系统的主 Use Case 图如图 A-4 所示。

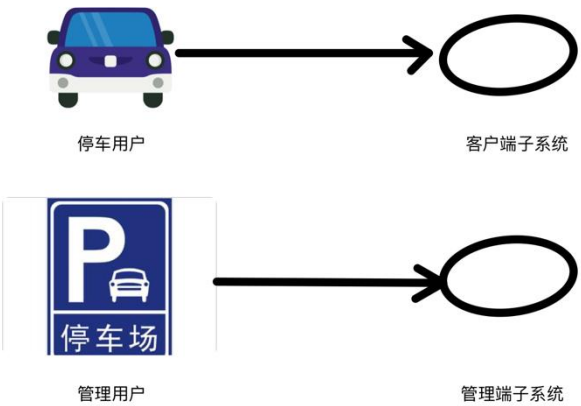


图 A-4：系统的主 Use Case 图

4.3 客户端子系统

需要泊车的用户通过客户端泊车系统进行相关的信息查询，这就是客户端子系统的主要功能。在客户端，用户可以看到车位余量，泊车后所产生的时长及对应费用，以及要查询的特定车辆的位置路径这几项。当输入车牌号，且系统检索成功（即当前数据库中存储有该车牌号所对应的车辆），即可显示该车辆的相关信息，以及从停车场的某一指定位置到该车位的最优路径。用户根据该路径可以在较短的时间内到达该车位。最后，用户驾车离场时，根据系统记录的信息结算本次泊车时长和使用该停车场所产生的费用。它的活动图如图 A-5 所示。

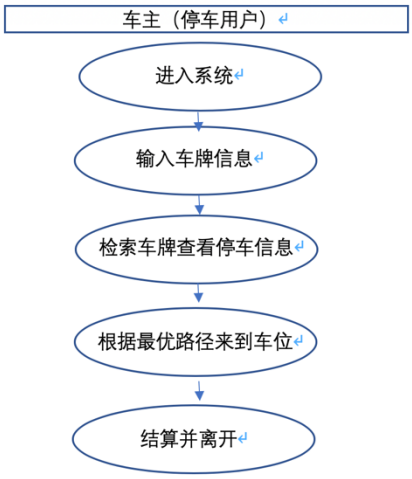


图 A-5：客户端子系统的活动图

客户子系统的功能主要包括余量显示、停车时长及费用显示、车位与路径指示等功能。它的用例图如图 A-6。

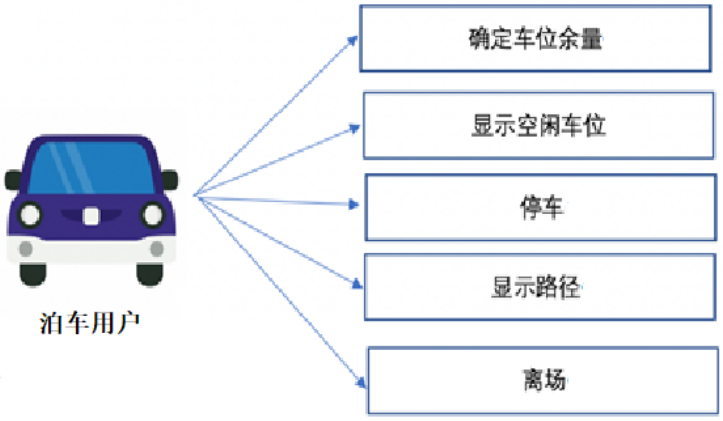


图 A-6：客户端子系统的功能用例

客户端管理的功能描述如下：

F-C-1：确定空闲车位

泊车用户通过停车场入口处的实时信息，或登录到客户端，可以看到停车场当前的车位余量信息。当有空闲车位时，车主即可驾车入场，满足其泊车需求。

安装在每个车位上方的红绿指示灯会根据数据库中的车位信息确定该车位是否已被占用。若是，则显示红灯；否则显示绿灯。车主驶入停车场内后，即可根据每个车位上的指示灯显示，较为清晰简便地找到空闲车位进行泊车。

F-C-2：泊车信息录入

根据用户的泊车行为，更新数据库中相应的车位及车辆信息。

F-C-3：路径等信息显示

泊车用户登录停车场客户端系统，输入需要查找的车牌号，系统检索成功后，返回相关时长、费用和最优路径等信息。

F-C-4：泊车结束

泊车用户根据记录的费用信息完成缴费。可通过客户端线上缴费，或由收费人员收取。缴费成功后修改该车辆状态，并更新相关车辆及车位信息。

4.3.1 空闲车位显示

空闲显示是泊车用户通过系统即可看到停车场当前的车位余量信息。具体描述如下：

用例描述：空闲信息显示

执行者：泊车用户

前置条件：泊车用户已登录到客户端

后置条件：确定有车位余量后，则可入场泊车

基本路径：

a) 泊车用户通过停车场入口处的实时信息显示，或登录到客户端，查看当前停车场的车位余量；

b) 确定有空闲车位后，车主即可驶入停车场寻找空闲车位。停车场内每个车位上的指示灯会根据数据库中信息确定此车位是否已被占用。若是，则显示红

灯；否则显示绿灯；

c) 车主即可根据每个车位上的指示灯显示，较为清晰简便地找到空闲车位进行泊车。

4.3.2 记录泊车

车主选定空闲车位后熄火泊车，待设备识别车牌后，将该车辆的相关信息录入数据库中，同时更新数据库中相关的其他信息。具体描述如下：

用例描述：记录泊车信息

执行者：泊车用户

前置条件：泊车用户已选定空闲车位

后置条件：成功泊车后自动更新记录

基本路径：

a) 车主选定空闲车位后熄火泊车，待安装在每个车位后方的车牌识别设备识别当前车牌后，将该车辆的相关信息录入到数据库中，同时启动该车位计时器执行计时功能；

b) 根据更新后的车位占用状态，该停车位的指示灯由绿变红，指示该车位已被占用。

4.3.3 查询泊车记录

车主结束泊车前，登录停车场客户端系统。输入需要查找的车牌号，系统在数据库中进行检索。检索成功后，显示相关信息。具体描述如下：

用例描述：泊车信息查询

执行者：泊车用户

前置条件：泊车用户已泊车且已登录到客户端

后置条件：根据返回信息完成缴费、离场等后续操作

基本路径：

a) 用户登录到客户端，输入需要查找的车牌号，数据库检索；

b) 系统根据数据库中记录的信息，给出该车辆泊车时长、车位信息、费用，以及从用户当前所在的指定位置到达该车位的最优路径等信息；

c) 车主可选择在线缴费，在客户端完成泊车费用结算；

d) 车主可根据系统提供的最优路径，在较短的时间内找到自己的车辆。当车牌识别设备检测到该车辆已驶离当前车位，更新数据库中该车辆与车位的状态，车位指示灯转为绿色。

4.3.4 记录离场

泊车用户根据记录的信息完成缴费，结束此次泊车动作。同时更新相关车辆及车位信息。具体描述如下：

用例描述：记录离场信息

执行者：泊车用户

前置条件：泊车用户已缴费

后置条件：车辆离场，泊车结束

基本路径：

a) 车主驾车前往停车场出口，收费人员根据系统记录核实该车辆是否缴费成功；若成功，放离车辆离场，更新数据库；若不成功，则根据系统记录收取费用；

b) 数据库中将该车辆相关的数据转为历史记录，客户端不再显示此次泊车记录。

4.4 管理端子系统

管理端主要是指提供给停车场管理人员使用的功能部分，它的功能分为显示剩余车位，提供最优路径智能算法，收费，车位上锁、解锁等部分。每位管理端的登录者经过身份认证后，根据相应的权限实现系统相应的操作和功能。它的用例图如图 A-7。

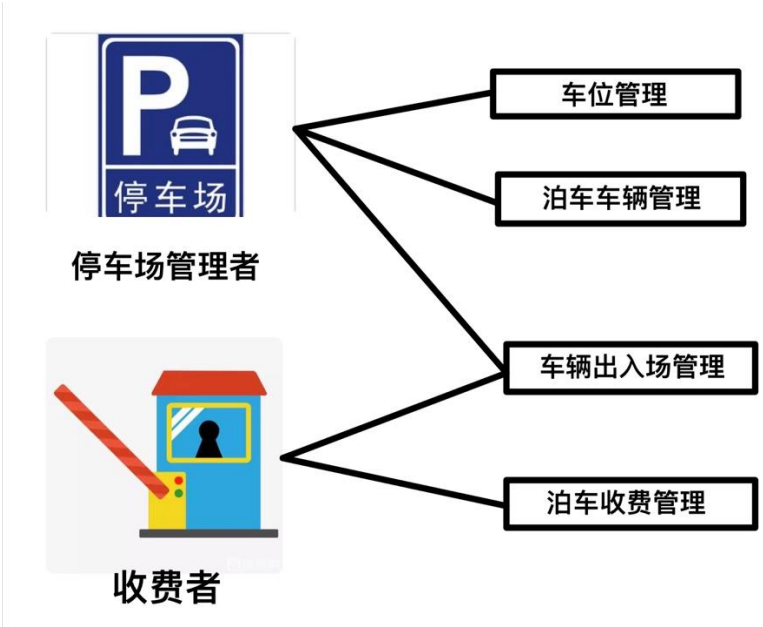


图 A-7：管理端用例图

管理端的这些 Use case（用例）描述如下：

F-L-1：登录管理

登录管理是负责所有的管理端的登录，管理端的人员要登录到管理端必须经过登录界面，输入自己的用户名和密码，通过判断这个用户的权限信息，不同的登录人可能具有不同的权限，根据不同的权限现实不同的功能。

F-M-1 停车场状态管理：

停车场状态管理用例是管理员登录到系统，设置停车场费用情况等的功能，及提供或查看某一车位的详细信息。

F-M-2 车辆管理：

车辆管理用例是管理员登录到系统，查看历史纪录中某一车辆泊车情况的详细信息。

F-M-3 出入场管理：

出入场管理用例是收费人员登录到系统，核实停车场和相关车辆信息，进行车辆的准入和放行，同时更新数据库。

4.4.1 登录管理

登录到管理端的所有人都需要通过登录界面进入相应的管理界面，不同的登录人具有不同的权限，根据登录人具有的权限将相应的功能现实在登录到的管理界面，没有权限操作的功能将在现实在这个界面上。活动视图如图 A-8：

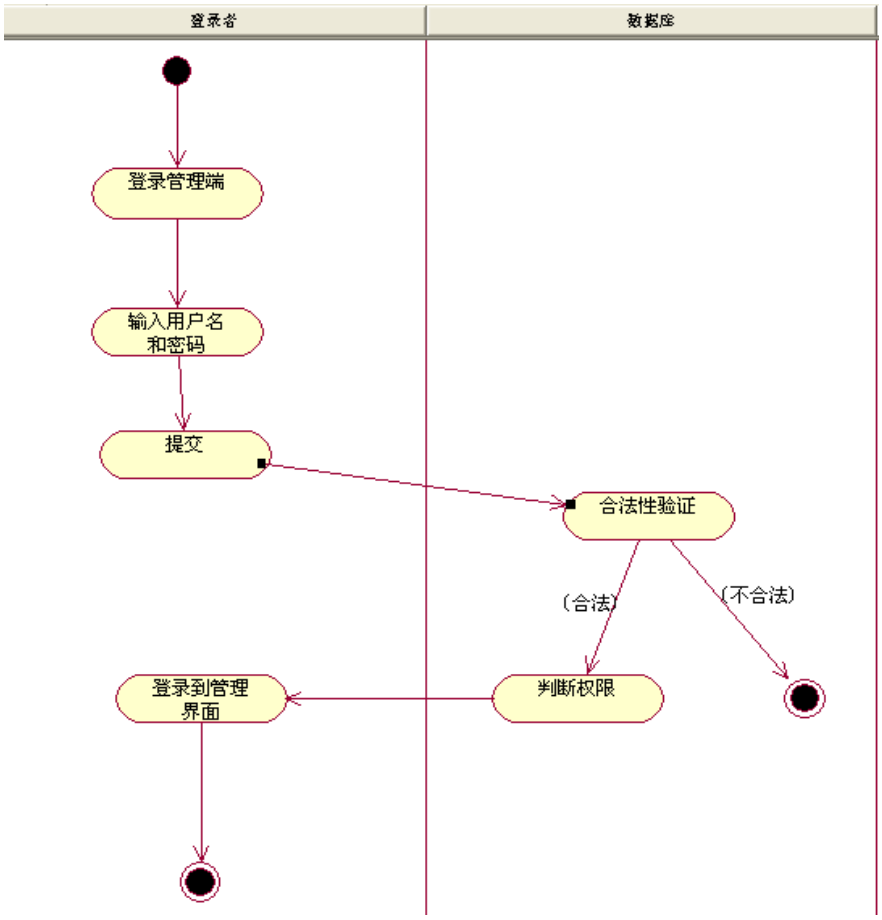


图 A-8：登录管理活动视图

4.4.2 停车场状态管理

登录到管理端的管理人员可以查看并管理当前停车场的总体车位情况，进行各个车位管理操作，查看停车场中各个车位的状态，该系统将停车系统的情况反馈给管理系统。具体描述如下：

用例描述：停车场状态管理

执行者：停车场管理者

前置条件：停车场管理者已登录系统

后置条件：若状态维护成功后，则停车场泊车价格，或某车位的状态随之变化

基本路径：

a) 进入停车场管理界面，首先展示目前停车场的状态，包括使用中车位数、空闲车位数、收费情况；

b) 选择每个车位可以详细浏览该车位在记录时间段的使用情况，包括占用过该车位的车辆信息、收费情况等。同时提供车位的个性化管理，可以根据实际需求手动修改该车位的占用状态，关闭此车位，使其不在客户端显示为空闲状态；

c) 提供修改泊车价格按钮，更改收费计算，并反应在客户端；

d) 提供最优路径算法结合停车场实地情况，为每一个车位规划从特定入口到该车位的

最优路径。

4.4.3 车辆管理

停车场管理者登录到系统后，检索并查看数据库中记录的某一车牌号对应的泊车详细情况。具体描述如下：

用例描述：车辆管理

执行者：停车场管理者

前置条件：停车场管理者已登录系统

后置条件：根据系统提供的车辆记录进行后续运营管理

基本路径：

a) 进入系统管理界面，输入需要查询的车牌号；

b) 系统检索记录，成功则返回该车辆的泊车详细记录。

4.4.4 出入场管理

收费人员在各个出入口处进行车辆的准入和放行。当有车辆要求进入停车场时，收费人员首先核实当前停车场中是否还有余位，有则放行，准许进入；当有车辆离场时，收费人员核实缴费情况后，准许其离场。具体描述如下：

用例描述：出入场管理

执行者：收费人员

前置条件：有车辆要求进入或准备离开

后置条件：如果收费人员确认信息后，则泊车车辆的相应信息记录到数据库中。

基本路径：

a) 进入管理界面，首先查看当前停车场中的车位情况；

b) 若有空闲车位则准许进入，并开始计时计费，同时更改该车辆状态。

c) 当有车辆准备离场时，核实该车辆的缴费信息，准许放行并更改该车辆状态。

5 性能需求

根据用户对本系统的要求，确定系统在响应时间、可靠性、安全等方面有较高的性能要求。

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、版面形象清新悦目、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

无论是客户端和管理端,当用户登录,进行任何操作的时候,系统应该及时的进行反应，反应的时间在 5 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，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 7X24 内不当机，保证 20 人可以同时在客户端登录，系统正常运行，正确提示相关内容。

5.4 开放性需求

系统应具有十分的灵活性，以适应将来功能扩展的需求。

5.5 可扩展性需求

系统设计要求能够体现扩展性要求，以适应将来功能扩展的需求。

5.6 系统安全性需求

系统有严格的权限管理功能，各功能模块需有相应的权限方能进入。系统需能够防止各类误操作可能造成的数据丢失，破坏。防止用户非法获取网页以及内容。

6 产品提交

提交产品为：
a) 应用系统软件包
b) 数据库初始数据

c） 系统开发过程文档
d） 系统使用维护说明文档
提交方式：CD 介质等其他存储器

7 实现约束

系统的实现约束如下：
a） 操作系统为 Windows 10
b） 开发平台为 eclipse-SDK-3.1.2-win32
c） 数据库为 SQL Server 2017

8 签字

本需求规格经过双方认可，特签字如下表Ａ－２。

表Ａ－２：需求规格签字

用户签署信息		企业签署信息	
单位名称	四川 XXX 公司	单位名称	电子科技大学学生 XXX
签署人姓名	ＸＸＸ	签署人姓名	ＸＸＸ
签署日期	2019.10	签署日期	2019.10

项 目 编 号	**200602006**
文 档 编 号	**11**
密　　级	内部

停车场管理系统概要设计

V1.0

软件工程实验

评　　审　　日　期：　2019 年 10 月 17 日

目 录

1.导言

1.1 目的

该文档的目的是描述停车场管理系统项目的概要设计，其主要内容包括：

- 系统功能简介
- 系统结构设计
- 系统接口设计
- 数据设计
- 模块设计
- 界面设计

本文档的预期的读者是：

- 开发人员
- 项目管理人员
- 测试人员

1.2 范围

该文档定义了系统的结构和单元接口，但未确定单元的实现方法，这部分内容将在详细设计/实现中确定。

1.3 缩写说明

UML

Unified Modeling Language（统一建模语言）的缩写，是一个标准的建模语言。

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。

1.4 术语定义

JSP Mode12

Servlet/JSP 规范的0．9 2版本中描述的术语，定义了如何在同一个应用程序中联合使用Servlet和JSP的体系结构。

JavaBean

用Ｊ ａ ｖ ａ语言实现的满足一定功能的类。

1.5 引用标准

[1]《企业文档格式标准》
北京长江软件有限公司
[2]《软件概要设计报告格式标准》
北京长江软件有限公司软件工程过程化组织

1.6 参考资料

[1]《实战ｓ ｔ ｒ ｕ ｃ ｔ》〔美〕Ｔ ｅ ｄ Ｈ ｕ ｓ ｔ ｅ ｄ
机械工业出版社
[2]《软件重构》
清华大学出版社

1.7 版本更新信息

本文档的更新记录如表Ｂ－1所示。

表B-1 版本更新记录

修改编号	修改日期	修改后版本	修改位置	修改内容概述
000	2019.10.15	0.1	全部	初始发布版本
001	2019.10.16	1.0	全部	修改

2.系统分析

本系统可以实现停车场的管理与车辆信息采集。泊车用户进入停车场后，由入口处地磁以及安装在闸门和车位旁的摄像头采集入场并成功泊车入位的车辆信息，并保存到数据库中。管理人员通过系统身份认证，并授予不同级别的权限后，访问数据库进行各项增删改查的操作，亦即修改停车场相关状态信息，并针对性地查看有记录的车辆信息。同时，管理单位可以不定期汇总泊车记录，浏览泊车记录，测评某时间段内停车场的车辆流动以及实际收益情况。方便运营企业管理。系统包括管理端子系统和客户端子系统。

管理端子系统包括车位管理、车辆管理、收费等信息管理、用户管理。客户端子系统包括余量显示，泊车时长及费用查询，移动支付缴费，最优路径显示。图 B—1 和 B—2 为客户端和管理端的组成结构图。

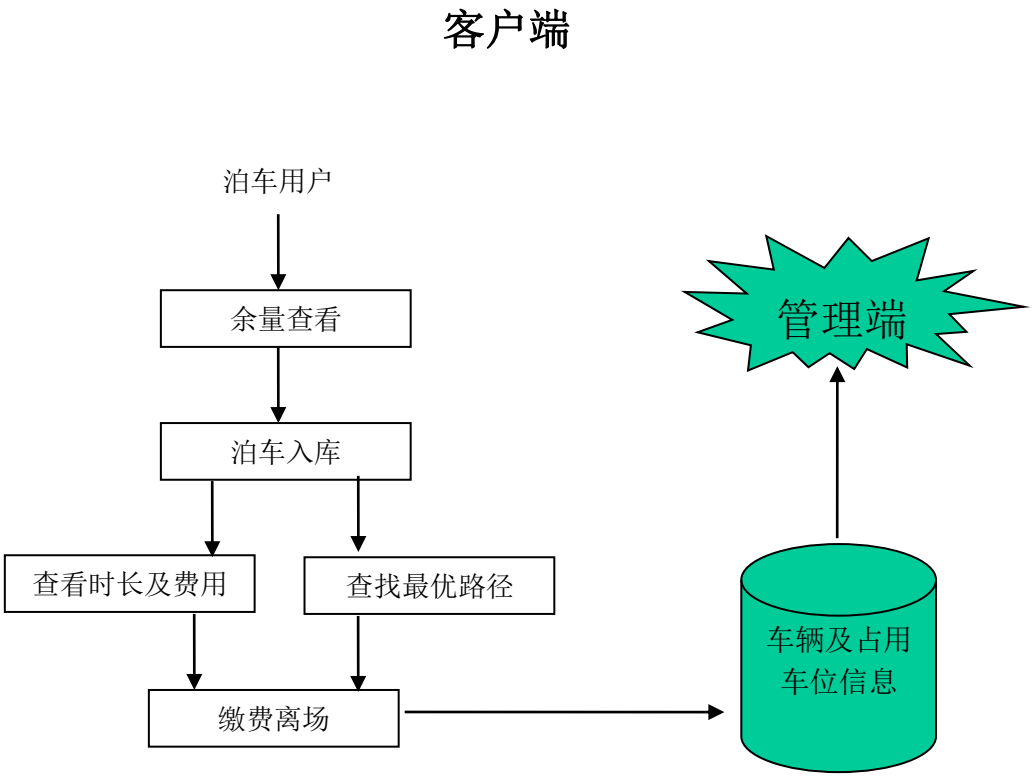
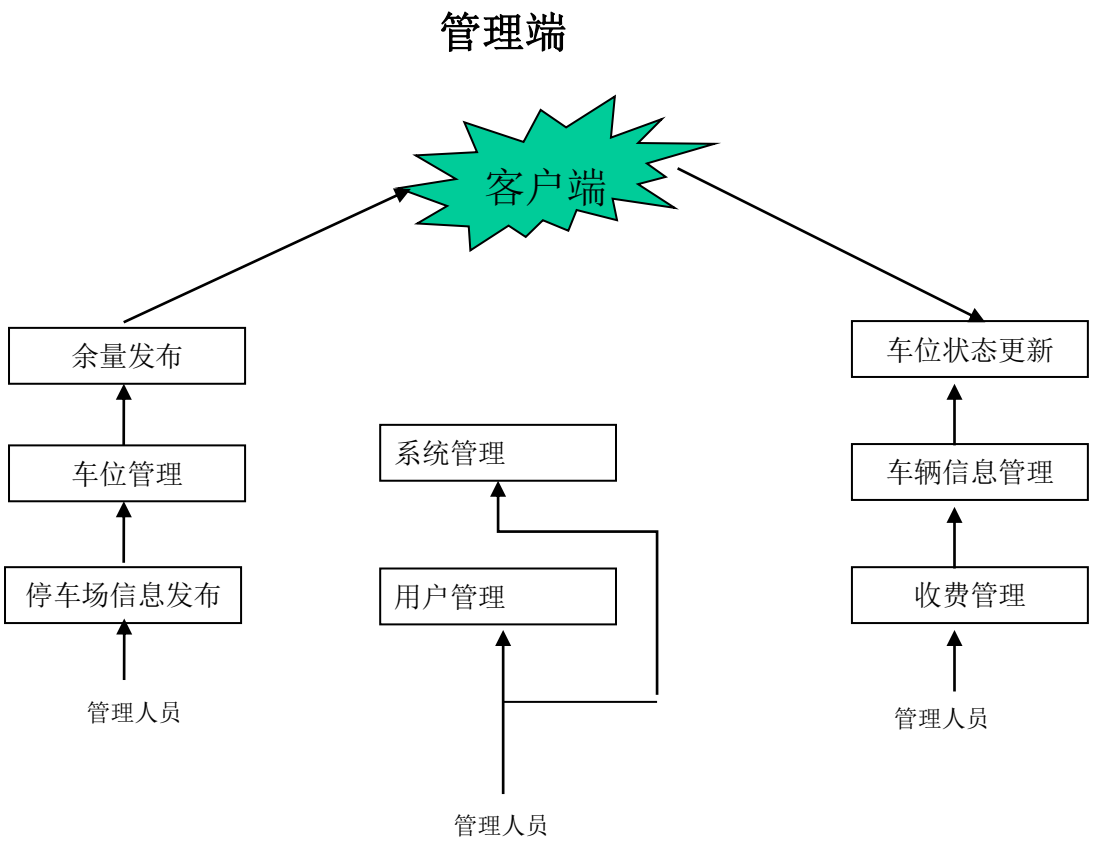


图 B — 1 ：客户端子系统图示



图Ｂ－２：管理端子系统

3.界面设计

本系统的用户界面按功能分客户端和管理端。

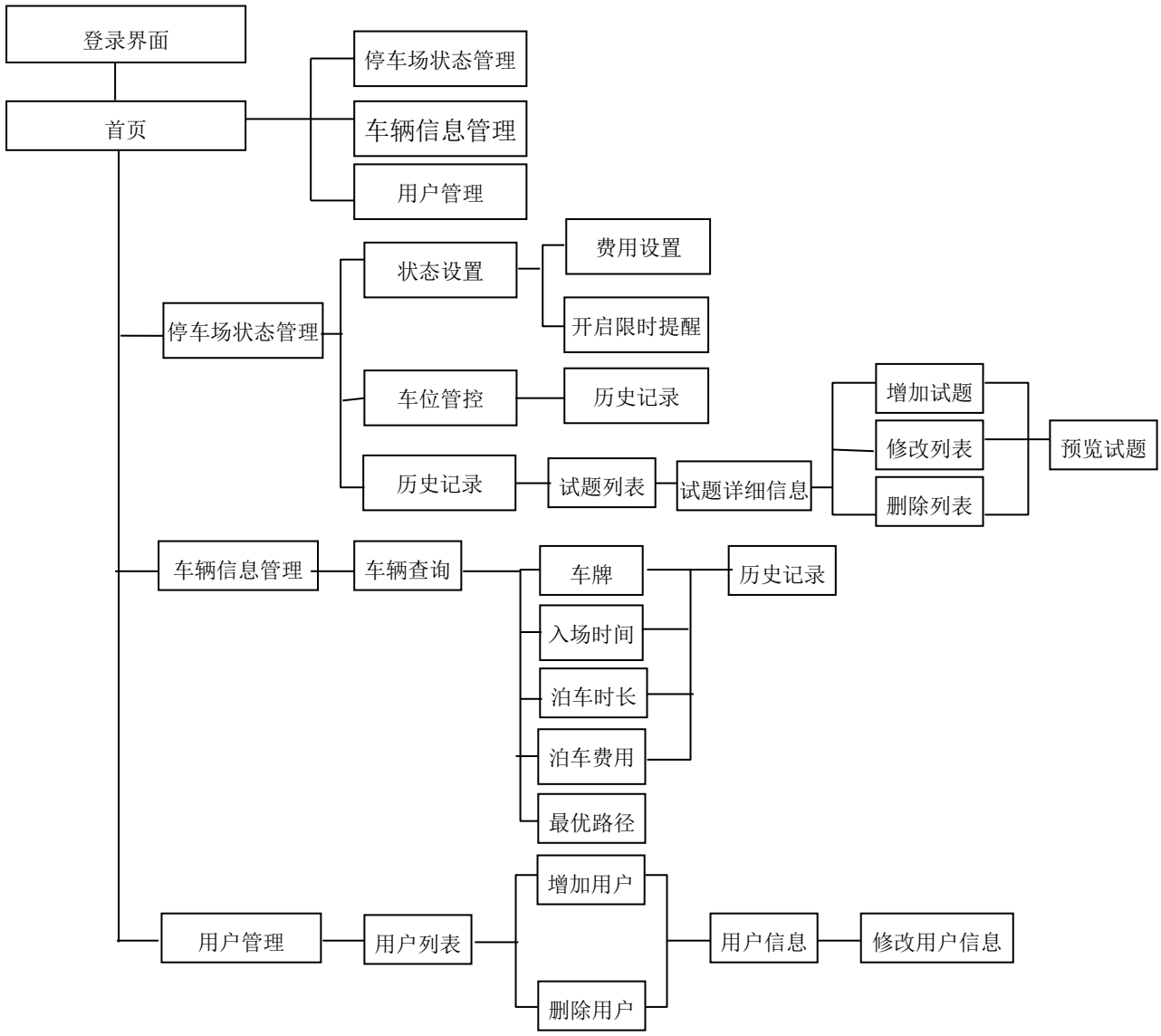
3.1 管理端界面设计

管理端主要实现车位管理、车辆管理、收费等信息管理以及用户管理等功能。主要界面设计如下：

- 登录界面
  - ◆ 通过用户名和密码实现用户登录，并判断用户的权限
- 管理首页
  - ◆ 根据用户的权限，进入首页，并在首页中展示此用户相应可以操作的权限功能。
- 停车场状态管理
  - ◆ 包括“费用设置”、“车位管控”、“开启限时提醒”、“历史记录”等页面。
- 车辆信息管理
  - ◆ 包括 “车辆查询”、“历史记录”等页面。
- 用户管理
  - ◆ 包括“用户列表”、“用户信息”、“修改用户信息”、“添加用户”和“删除

用户”、“用户权限设置”等页面。

具体页面流如下图Ｂ－3 所示：



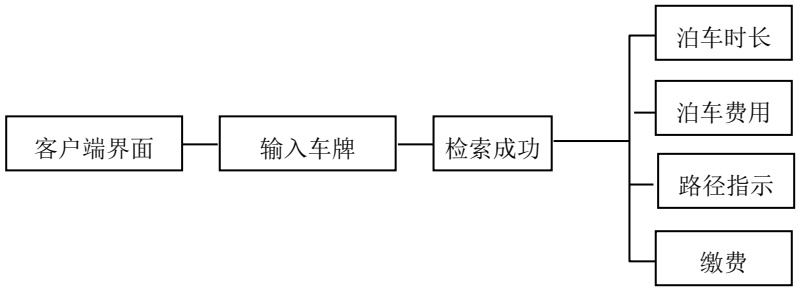
图Ｂ－３：管理端的页面流程

3.2 客户端界面设计

客户端主要为车主提供泊车相关服务的过程，车主通过查看余量，确定当前有空闲车位以供使用，驾车入场，由摄像头以及地磁系统自动采集车牌等信息，并录入数据库中。当车辆泊如某车位时，由车位旁的摄像头采集此车位上停放的车辆信息，录入此车位的数据库信息，并修改该车牌对应的车辆信息。准备结束泊车的用户，在客户端界面输入需要查询的车牌信息，检索成功后，可根据个人需求查看泊车时长、费用，以及从当前位置到达泊车位置的最优路径。

首先进入客户端界面，点击“车牌查询”进入检索界面，输入要查询的车牌后进入子界面，包括“泊车时长信息”、“泊车费用信息”、“路径指引”和“停车缴费”等页面。

具体页面流如下图Ｂ－４所示：



图Ｂ－４：客户端的页面流程

4.体系结构

系统的总体结构设计遵循如下原则：

1）系统应具有良好的适应性：能适应用户对系统的软件环境、管理内容、模式和界面的要求；

2）系统应具有可靠性：采用成熟的技术方法和软件开发平台，以保证在以后的实际应用中安全、可靠；

3）系统应具有较好的安全性：应提高完善的安全机制和用户权限限制机制，确保数据的受限访问；

4）系统应具有良好的可维护性：系统应易于维护、安装；

5）系统应具有良好的可扩展性：系统应适应未来信息化建设的要求，能方便得进行功能扩展，以建立完善的信息集成管理体系。

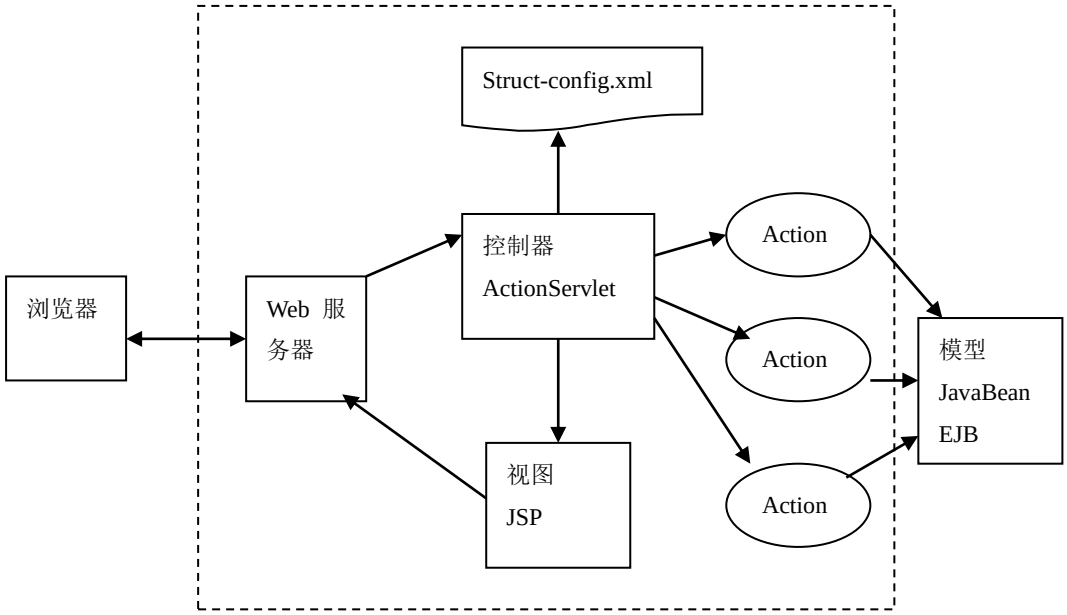
本系统采用 struts 体系结构，Struts 是一个基于模型 (Model) －视图 (View) －控制器 (Controller)(MVC) 模式的应用架构的开源框架。

4.1 体系结构

目前软件项目中有很多的体系结构，其中ｓｔｒｕｃｔ是比较流行的一种。

４.１.１Ｓｔｒｕｃｔ体系结构

对于开发 Web 应用，要从头设计并开发出一个可靠、稳定的框架不是一件容易的事情，随着 Web 开发技术的日趋成熟，在 web 开发领域出现了一些现成的优秀的框架，开发者可以直接使用它们，ｓｔｒｕｃｔ就是一个很好的框架结构，它是在 JSP Model2 基础上实现的一个 MVC 框架,它可以使你不必要从头开始全部开发组件,对于大项目更是很好的。在 Strcut 框架中模型由实现业务逻辑的 JavaBean 或者 EJB 组件构成，控制器由 ActionSevelet 和 Action 来实现，视图由一组 JSP 文件组成，图Ｂ－５显示了 Struct 实现的 MVC 框架。



图Ｂ－５:Struct 实现的 MVC 框架

其中：
视图：就是一组 JSP 文件，这些 JSP 文件没有业务逻辑，也没有模型信息，只有标签，这些标签可以是标准的 JSP 标签或者是客户化标签，如 Struct 标签库的标签。此外，通常将 Strcut 框架中的 ActionForm Bean 也划为视图模块中，ActionForm Bean 是一种 JAVABEAN，除了具有一些 JAVABean 的常规方法外，还包含了一些特殊的方法，用于验证 HTML 表单数据以及将其属性重新设置默认值。Struct 框架利用 ActionForm Bean 来进行视图和控制器之间表单数据的传递。Strcut 框架将用户输入的表单数据保存在 ActionForm Bean 中，将它

传递给控制器，控制器可以对 ActionForm Bean 中的数据进行修改，JSP 文件使用 Strcut 标签读取修改后的 ActionForm Bean 的信息，重新设置 HTML 表单。
控制器：控制器由 ActionServlet 类和 Action 类实现，ActionServlet 类是 Struct 框架中的核心组件。是这个 MVC 的中央控制器的角色，ActionServlet 主要负责接收 HTTP 请求的信息，根据配置文件 struct-config.xml 的配置信息，将请求转发给适当的 Action 对象，如果该 Action 对象不存在，ActionServlet 会先创建这个 Action 对象。Action 类负责调用模型的方法，更新模型的状态，并帮助控制应用程序的流程，对于小型简单的应用，Action 类本身也可以完成一些实际的业务逻辑。
模型：模型表示应用程序的状态和业务逻辑，业务逻辑常常由 JavaBean 或者 EJB 组件实现。

如果在 Web 应用开发中套用现成的 Struct 框架，可以简化每个开发阶段的工作，开发人员可以更加有针对性地分析应用需求，不必重新设计框架，只需在 Struct 框架的基础上，设计 MVC 各个模块包含的具体组件，在编码过程中，可以充分利用 Struct 提供的各种实用类和标签库，简化编码工作。

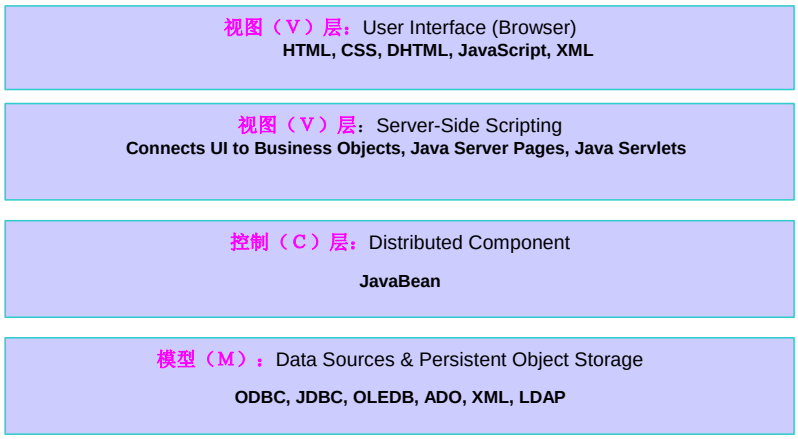
Struct 框架可以方便迅速地将一个复杂的应用划分成模型、视图和控制器组件，而 Struct 的配置文件 struct-config.xml 可以灵活地组装这些组件，简化开发过程。

4.1.2 系统体系结构

根据系统分析结果，该系统从结构上应满足：

- 基于浏览器进行显示以方便用户使用；
- 采用 MVC 的三层体系结构，分化各个功能组件；
- 采用 JDBC 技术与数据库通讯以便于数据库的转换；
- 采用标签技术完成动态页面的简单逻辑。

本系统的体系结构如图Ｂ－６，它基本遵循了 struct 体系的 MVC 框架规范。

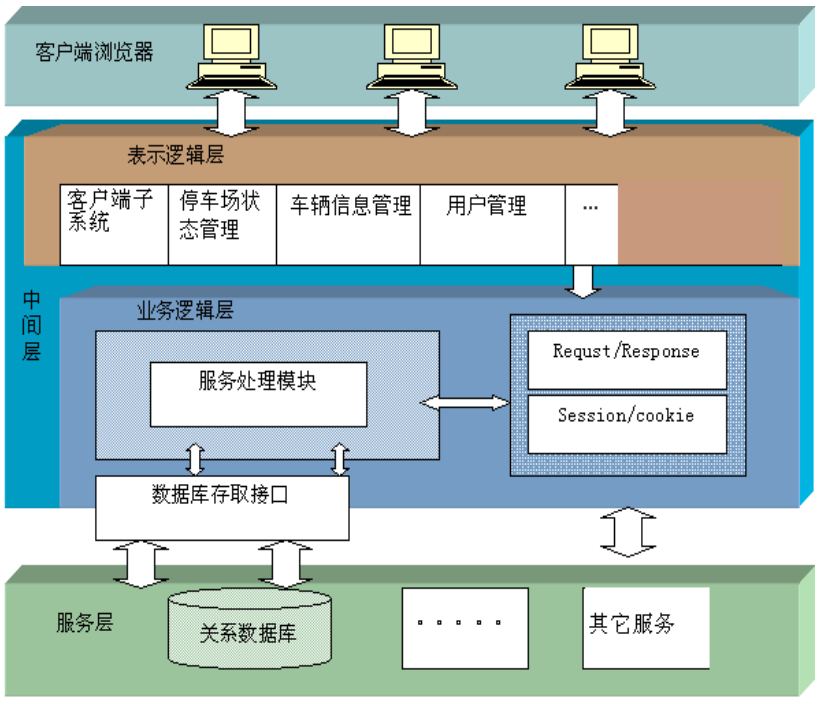


图Ｂ－６：系统的体系结构

其中：

表示层：用于与用户进行交互以及显示结果。包括所有的 JSP，提供用户界面，接受用户输入， 还包括相应的 ActionFrom Bean，用来存放表单数据，并进行表单数据验证。
控制层：包括所有的 Action 类，它完成三项任务：一是进行业务逻辑验证，二是调用模型组件，三是决定将合适的视图组件返回给用户。
模型：进行逻辑处理的 JavaBean 等。数据库采用 ODBC 技术以提供数据库的可移植性

体系结构的具体拓扑图示如图Ｂ－７。



图Ｂ－７：结构拓扑图

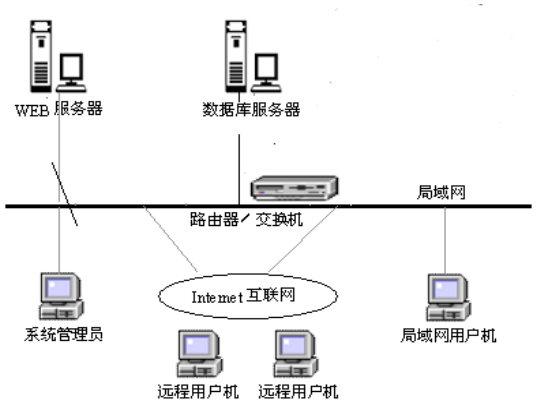
客户层主要是指用户登录的 Web 浏览器；中间层负责平台的业务逻辑处理和表示逻辑生成；服务层提供底层的信息数据库服务器。
1.客户层：用于与企业信息系统的用户进行交互以及显示根据特定业务规则进行计算后的结果。本系统将完全采用基于 WEB 的（B/S 架构）客户端，即用户可以直接通过浏览器来访问和使用本系统。
2.中间层：这相当于三层标准架构中的 Web 应用服务层，支持诸如响应客户请求以及查询等功能。并且由中间层进行逻辑处理，再处理的结果反馈给客户或者发送到数据库中。
3.服务层：主要是数据库系统，这里的数据库系统主要是关系数据库系统(RDMS)。

4.2 系统运行环境

系统运行的网络结构图、硬件软件环境图如下。

4.2.1 网络结构图

本系统的网络拓构图如图Ｂ－８：



图Ｂ－８：系统的网络拓扑图

其中的局域网用户机主要是公司内部的人员可以使用的机器,远程用户机主要是指通过互联网登录系统的人员使用的机器，主要是公司内部的管理人员。

4.2.2 硬件环境

本系统的硬件环境如下：

- 客户机：普通 PC
  - CPU：P4 1.8GHz 以上
  - 内存：256MB 以上
  - 能够运行 IE5.0 以上或者 Netscape4.0 以上版本的机器
  - 分辨率：推荐使用 1024*768 像素
- WEB 服务器
  - CPU：P4 2.0GHz
  - 内存：1G 以上
  - 硬盘：80G 以上
  - 网卡：千兆
- 数据库服务器
  - CPU：P4 2.0GHz
  - 内存：1G 以上

- 硬盘：80G 以上

4.2.3 软件环境

本系统的软件环境如下：

- 操作系统： windows10
- 数据库：SQL Server 2000
- 开发工具包：JDK Version 1.4.2
- 开发环境：eclipse-SDK-3.1.2-win32
- Web 服务器：Tomcat
- 浏览器：IE6.0 以上

（1）数据库及操作系统:

对于核心数据库来说，选择一个合适的数据库系统对我们的系统运行是很重要的，选择数据库的关键因素是要考虑预计会有多少人同时访问数据库；正常工作时间的级别；用来访问数据库的应用程序的类型；运行数据库的服务器的硬件和操作系统类型；以及管理人员的专业技术水平。目前市场上适用于中小型企业的数据库产品有 IBM DB2 、Microsoft SQL Server 系列、 Oracle 系列。所有这些产品都基于 SQL 语言。同时，它们还拥有精密复杂的安全控制以适应不同的商业需要。服务器操作系统使用 Windows10 系统。

考虑到价格因素、易用性，我们使用 SQL SERVER 2000 作为系统后台数据库系统，服务器操作系统采用 Windows10。

（2）WEB 服务软件：

目前的 WEB 服务器软件有很多种，成熟而且稳定有 Apache、Tomcat 和 Microsoft 的 IIS ，它们也是占据着 Web 服务器市场最大的份额。Tomcat 是 Sun 和 Apache 合作做出来的 JSPServer, 支持 Servlet2.2 及 JSP1.1 等版本。而且 Tomcat 未来将会取代 Jserv，成为 Apache 主要的 Servlet&JSPEngine。Tomcat 在设计上是以独立的 Server 执行,而不像 Jserv 是附在 Apache 中,这样就更可以发挥在 servlet 中，非 HttpServlet 的能力。Tomcat 是 Java 程序，所以只要有 JDK 就可以使用,不需要考虑操作系统平台。因此选择 Tomcat 作为 WEB 服务器。

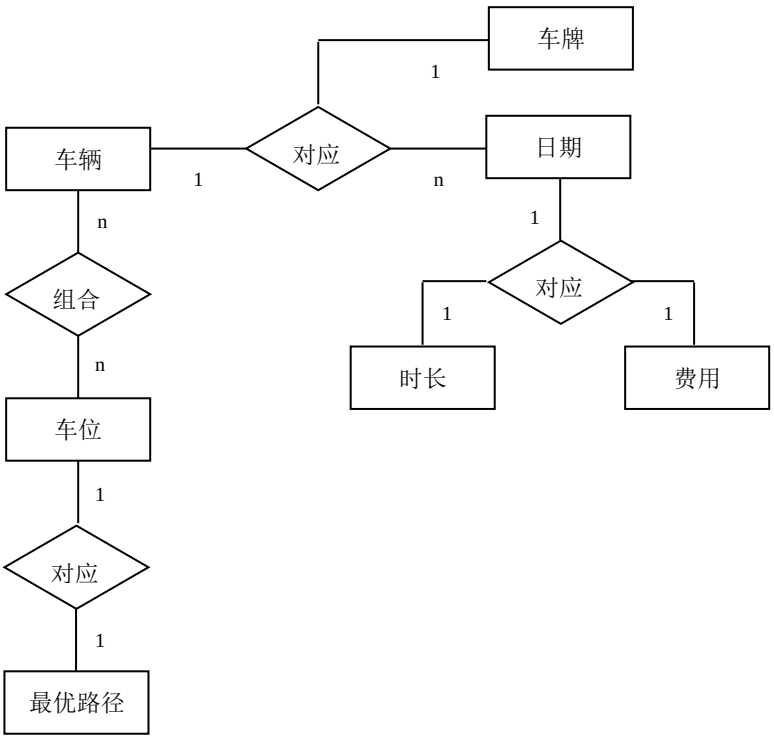
5.数据模型

本系统的数据模型主要是进行数据库的设计。

5.1 数据库的概念结构模型设计

概念设计以反映现实世界中的实体、属性和它们之间的关系等的原始数据形式，建立数

据库的每一幅用户视图。图Ｂ－９是系统 E-R 图。其中系统中的管理用户



图Ｂ－９：系统的Ｅ－Ｒ图

5.2 数据库的逻辑结构模型设计

数据库的逻辑设计是将各局部的 E-R 图进行分解、合并后重新组织起来形成数据库全局逻辑结构，包括所确定的关键字和属性、重新确定的记录结构、所建立的各个数据之间的相互关系。根据本系统需求分析,系统的数据库包括了停车场管理、车辆管理、车位管理、用户管理以及需要的基本数据字典等部分。

停车场管理包括的库表：

- history-历史纪录
- Park_Info－停车场相关状态

车辆管理包括的库表：

- car － 车辆相关信息
- park － 泊车相关信息

车位管理包括的库表：

- spot － 车位相关信息

用户管理包括的库表：

- users－管理端用户的信息

基本字典包括的库表：

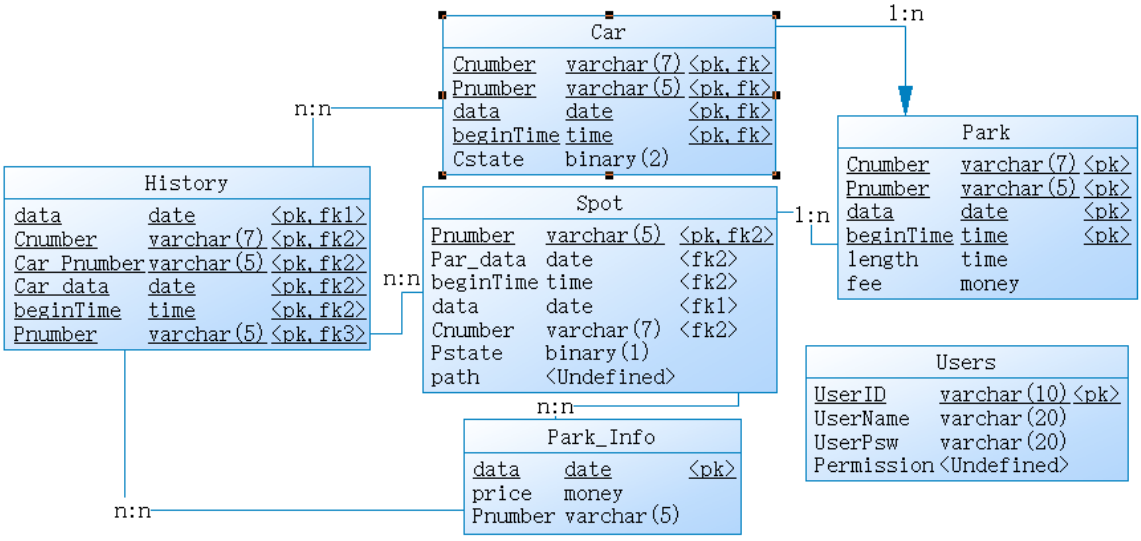
- number－车牌信息，格式固定，例如川 A11111、渝 BABCDE 等等
- Pnumber-车位标号，根据停车场具体车位规划，为每一个车位编制唯一的编号，可以时字母、数字或二者组合，例如 01、02、A1、C10 等等
- Cstate-车辆状态，包括入场、出场、占用三种状态
- Pstate-车位状态，包括空闲、占用两种状态

表Ｂ－2 是对表 Park 的设计，其它库表的设计详细见数据表设计文件。

表Ｂ－２：Park（泊车相关信息）表

字段名	字段代码	字段类型	关键字	可否为空	代码字典表
车牌号	Cnumber	VARVARCHAR(7)	Y		number
日期	data	VARVARCHAR(6)	Y	N	
车位号	Pnumber	VARVARCHAR(5)	Y	N	location
起始时刻	beginTime	time			
时长	length	double			
泊车费用	fee	money			

在确定了各个表主键字段的基础上,依据表与表相关字段之间的联系建立了各表之间的关系，如图 B-1 0 所示。



图Ｂ－１０：系统数据库表关系图

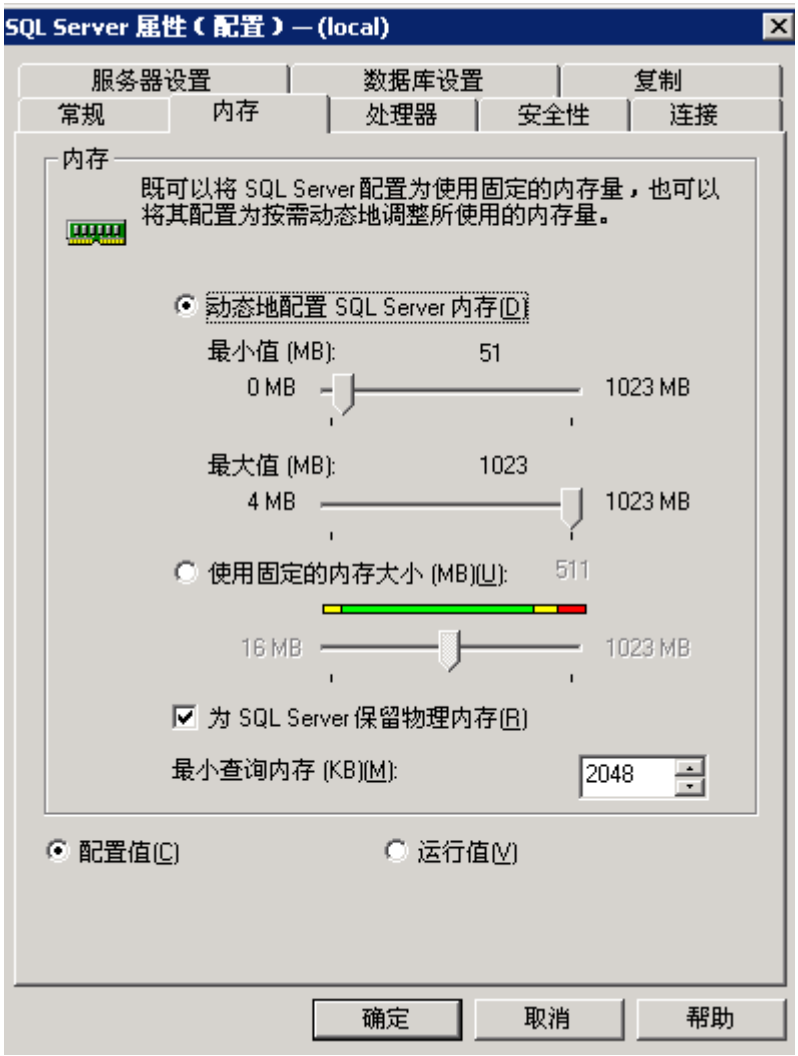
5.3 数据库管理物理结构模型设计

信息存储结构的设计在系统的设计中至关重要，要考虑到数据冗余、系统执行效率、信

息控制以及维护等方面的要求。信息的管理离不开数据库的支持，我们采用ＳＱＬ　Ｓｅｒｖｅｒ　２０００数据库管理系统。

数据库的物理设计主要是对数据在内存中的安排，包括对索引区、缓冲区的设计；对使用的外存设备及外存空间的组织，包括索引区、数据块的组织与划分；设置访问数据的方式方法。需在非系统卷（操作系统所在卷以外的其他卷）上安装 SQL Server 程序及数据库文件。内存是影响 Microsoft SQL Server 系统性能的一个重要因素，应在 Microsoft SQL Server 数据库安装后进行内存选项(Memory)设置，最大配置值为 2GB。

为了确定 SQL Server 系统最适宜的内存需求，可以从总的物理内存中减去 Windows 10 需要的内存（120M）以及其它一些内存需求后综合确定，理想的情况是给 SQL Server 分配尽可能多的内存，而不产生页面调度。设置服务器的虚拟内存为 1G。对 Sql 属性配置如图Ｂ－１１所示。



图Ｂ-１１ SQL 内存配置图

通过 ADO 对象提供的 OLEDB 接口与数据库连接。

1、首先建立创建 Connection 对象

SqlDatabaseName = "PMS"

SqlPassword = "xxxxxxxx"

```
SqlUsername = "sa"
SqlLocalName = "xx.xx.xx.xx"
ConnStr = "Provider = Sqloledb; User ID = " & SqlUsername & "; Password = " &
SqlPassword & "; Initial Catalog = " & SqlDatabaseName & "; Data Source = " &
SqlLocalName & ";"
set cnn = server.createobject("ADODB.Connection")
cnn.open ConnnStr
```

2、不直接打开 recordset 记录集，充分利用连接池

```
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open SQL,cnn
Set rs1 = Server.CreateObject("ADODB.Recordset")
rs1.Open SQL,cnn
```

这种打开连接池的方式可以节省数据库服务器的内存资源。

3、使用记录集后，最早的时间释放数据库资源

```
Rs.close
set rs=nothing
cnn.close
set cnn=nothing
```

本项目中后台数据库采用 SQL Sever 数据库系统。数据库各库表的脚本代码见数据库脚本文件。

6.模块设计

按照功能分解，本系统分为客户端管理系统和管理端系统。根据页面流的设计，管理端系统又分为用户登录、用户管理、车位管理、车辆管理、停车场管理 5 个模块，如图 B-12。

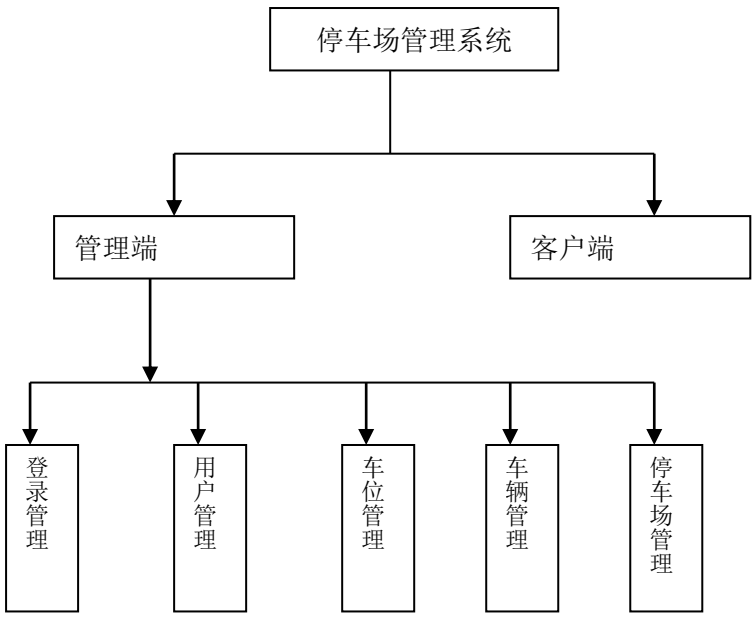
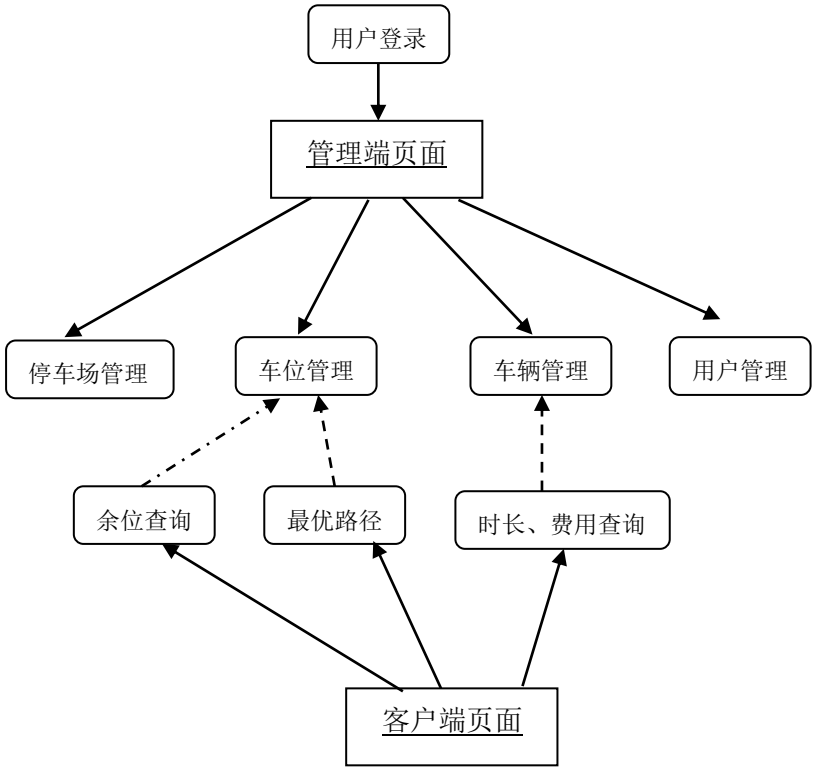


图 B-12：模块设计

各个部分的关系如图Ｂ－１３下：



图Ｂ－１３：模块之间的关系

针对不同用户，提供了两类功能登录接口：即泊车用户登录接口和停车场管理人员登录接口，这些接口都以 WEB 页面的形式提供。通过各自的页面，用户和管理人员可以从事各自的活动。

以下将分小节对各个部分进行设计

6.1 客户端模块设计

客户端运行在公网上，可以显示停车场相关信息，和车辆相关的泊车信息。泊车车主可以查看车位的详细信息。

6.1.1 表示层设计

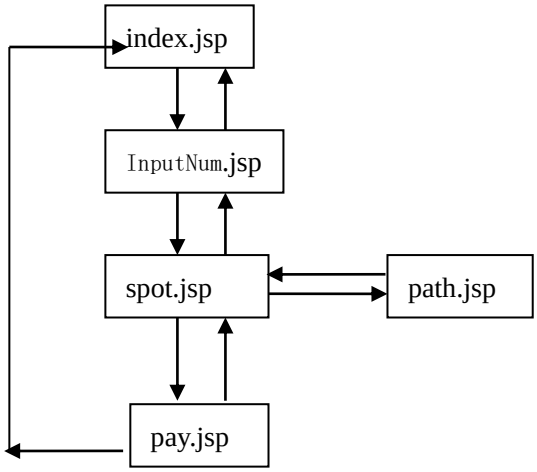
根据上述的功能介绍，总结出客户端的页面设计如表Ｂ-３。

表Ｂ－３：客户端的页面设计

界面	JSP	功能描述

主页面	index.jsp	客户端的主页面
车牌号	InputNum.jsp	输入需要检索的车牌号
车位详细信息	spot.jsp	显示对应车位的详细信息
显示路径	path.jsp	最优路径界面
移动支付	pay.jsp	显示移动支付成功界面

根据界面流的设计可以确定各个界面的访问入口，以及界面之间切换关系，流程图如图B-1 4所示。



ActionForm Bean用于在视图组件和控制组件之间传递HTML表单数据，通常每个HTML表单对应一个ActionForm Bean。此外，ActionForm Bean的validate（）方法用于对用户输入的数据进行合法性验证，由于ActionForm Bean工作于视图组件和控制器组件之间，不会访问模型组件，因此，validate（）方法通常不涉及对数据的业务逻辑验证，只是完成简单的数据格式和语法检查。表B-4列出了ActionForm以及进行相应的验证。Html表单中的字段和ActionForm Bean中的属性是一一对应的。

表B－4：客户端的ＡｃｔｉｏｎＦｏｒｍ

界面	JSP	ActionForm
主页面	Index.jsp	
输入车牌号	InputNum.jsp	InputNumForm
车位详细信息	spot.jsp	SpotForm
显示路径	path.jsp	PathForm
移动支付	pay.jsp	PayForm

6.1.2 控制层

控制层主要是设计 Action 组件，Action 负责单个事件的流程控制， Action 映射决定了 Action 与其它 Web 组件之间的关联关系. 客户端的事件主要包括输入车牌号、浏览该车辆的泊车详细信息，查看最优路径，进入移动支付界面，完成缴费，所有页面的上级返回动作等动作。表Ｂ-５列出了每个 Action 的入口（即调用 Action 的组件）、传递 Action 的 ActionForm，以及出口（即 Action 将请求转发到目标组件）。

表Ｂ-５：客户端应用的 Action 映射

事件	Action	入口	ActionForm	出口
进入输入界面	EnAction	index.jsp	InputNumForm	InputNum.jsp
浏览本次泊车详细信息	SpotAction	InputNum.jsp	SpotForm	spot.jsp
进入路径显示界面	PathAction	spot.jsp	PathForm	path.jsp
选择移动支付	PayAction	spot.jsp	PayForm	pay.jsp
返回泊车详细信息页面上一步	BackSpotAction	spot.jsp		InputNum.jsp
返回路径显示页面上一步	BackPathAction	path.jsp		spot.jsp
返回支付页面上一步	BackPayAction	pay.jsp		spot.jsp
缴费成功返回	BackExitAction	pay.jsp		InputNum.jsp

6.1.3 模型层

在 struct 框架中，模型组件负责完成业务逻辑，模型组件可以是 JavaBean、EJB 和实用类。客户端的业务逻辑主要是完成数据库的操作，修改车辆的缴费状态到数据库中。具体的模型组件见表Ｂ－６所示。

表Ｂ－６：客户端的模型组件

模型组件	描述
DBUtil	数据库的基本操作,为复用组件
CVDataBean	修改车辆的缴费状态到数据库中。

6.2 登录管理模块设计

登录管理模块负责管理端用户的登录。管理端用户都是通过登录界面进入管理端的，用户输入用户名和密码进入管理界面首页，提供了进入功能面板的接口，并根据用户的权限在

首页中列出相应的操作功能。

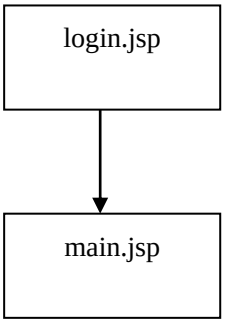
6.2.1 表示层设计

根据上述的功能介绍，总结出用户登录的页面如表Ｂ－７所示。

表Ｂ－７：登录管理模块的页面设计

界面	JSP	功能描述
登录界面	Login.jsp	登录的主页面
主页面	main.jsp	管理主页面
页面中部	center.jsp	复用页面：页面中心部分
页面上端	Top.jsp	复用页面：页面上部分
页面左端	Left.jsp	复用页面：页面的左部分
页面下端	bottom.jsp	复用页面：页面的下部分

根据界面流的设计可以确定各个界面的访问入口，以及界面之间切换关系，页面的流程图如图Ｂ－１５所示。



图Ｂ－１５：登录管理的页面流程

6.2.2 控制层设计

登录管理的控制层主要是设计用户的登录事件的流程控制（Action）。表Ｂ－８列出了每个 Action 的入口（即调用 Action 的组件）、传递 Action 的 ActionForm，以及出口（即 Action 将请求转发到目标组件）。

表Ｂ－８：登录管理的控制层设计

事件	Action	入口	ActionForm	出口

用户登录	LoginAction	login.jsp	LoginForm	main.jsp

6.2.3 模型层设计

登录管理模型组件负责完成用户信息的数据库操作的业务逻辑模型,建立封装了用户信息的 bean，这个 Bean 主要完成验证用户相关信息是否存在，并判断其权限。模型组件见表 B－9 。

表Ｂ－９：客户端的模型组件

模型组件	描述
DBUtil	数据库的基本操作,为复用组件
Permission	判断用户权限的类。
Userbean	Bean 主要完成验证用户信息是否存在。

6.3 用户管理模块设计

在用户管理中系统管理员对用户进行增删改查，可以进行权限设置。具有不同权限的用户进入不同的主界面，界面左侧栏中的图标数有所不同，具体的图标与用户所具有的权限对应。在用户管理中可以增加或删除用户，编辑用户名，用户密码，修改用户权限等。

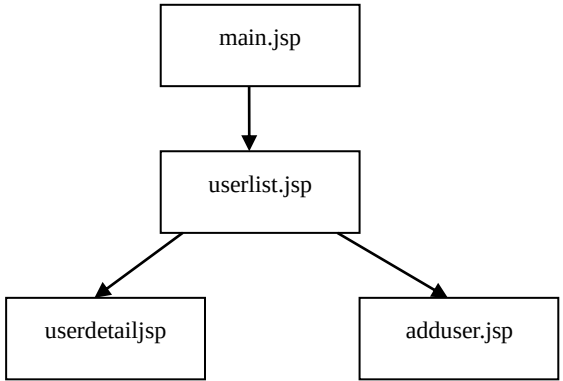
6.3.1 表示层设计

根据上述的功能介绍，总结出用户管理功能的页面如表Ｂ－１０。

表Ｂ－１０：用户管理的页面设计

界面	JSP	功能描述
用户列表	main.jsp	用户管理主界面
用户详细信息	userlist.jsp	用户详细信息的界面
修改用户	userdetail.jsp	完成特定用户信息的维护,即修改删除等
添加用户	adduser.jsp	增加用户信息

用户管理模块各个表示页面之间的关系如图Ｂ-１６。



图Ｂ－１６：用户管理页面之间的关系图

6.3.2 控制层设计

用户管理的控制层主要负责进入用户信息界面、进入浏览用户信息界面、增加用户信息、修改用户信息、删除用户信息等事件的流程控制，即设计用户管理的Ａｃｔｉｏｎ。表Ｂ－１１列出了用户管理的每个 Action 的入口（即调用 Action 的组件）、传递 Action 的 ActionForm，以及出口（即 Action 将请求转发到目标组件）。

表Ｂ－１１：用户管理的控制层设计

事件	Action	入口	ActionForm	出口
进入用户信息界面	UserListAction	main.jsp	UserListForm	userlist.jsp
进入浏览用户信息界面	UserDetailAction	userlist.jsp	UserDetailForm	userdetail.jsp
修改用户信息界面	ModifyUserAction	userlist.jsp	ModifyUserForm	userdetail.jsp
提交修改用户信息界面	SubmitModifyAction	userdetail.jsp	ModifyUserForm	userlist.jsp
增加用户信息	AddUserAction	userlist.jsp	AddUserForm	userdetail.jsp
提交增加用户信息	SubmitAddAction	userdetail.jsp	AddUserForm	userlist.jsp
删除用户信息	DelUserAction	userlist.jsp	DelUserForm	userlist.jsp

6.3.3 模型层设计

用户管理的模型层主要是完成用户信息的浏览、增加用户信息、用户信息维护等业务逻辑，并于完成相应的数据库操作。模型组件见表 B-12。

表Ｂ－12：用户管理模块的模型组件

模型组件	描述
DBUtil	数据库的基本操作,为复用组件
Userbean	主要完成管理用户信息。

6.4 停车场状态管理模块设计

停车场状态管理是对车位信息库和车辆信息库的维护和管理,对状态库的车位信息进行的增、删、改的功能，及提供泊车库中各车位的详细信息，。

6.4.1 表示层设计

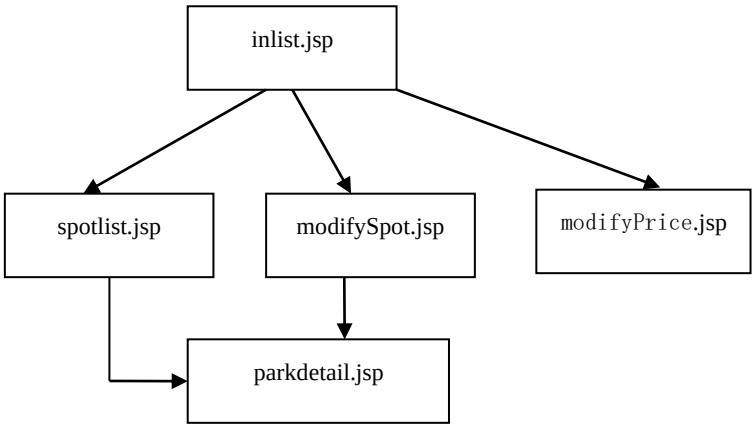
根据上述的功能介绍，总结出停车场状态管理功能的页面设计如表Ｂ－13 所示。

表Ｂ－13：停车场状态管理模块的界面

界面	JSP	功能
停车场管理首页	inlist.jsp	停车场状态管理主页面
车位信息列表	spotlist.jsp	显示各车位列表
车位泊车记录详细信息	parkdetail.jsp	某一车位泊车记录详细信息
修改车位状态	modifySpot.jsp	修改车位状态的页面
修改费用	modifyPrice.jsp	修改收费信息的页面

注：停车场状态管理模块中也用到前面提到的公共页面，例如
left.jsp ,top.jsp,bottom.jsp

各个界面流的基本流程关系图Ｂ－17 所示。



图Ｂ－１7：停车场状态管理的页面流程图

6.4.2 控制层设计

停车场状态管理的控制层主要负责进入停车场信息界面、进入车位详细信息界面、进入修改车位状态、修改费用信息界面、进入车辆详细信息界面等事件的流程控制。表Ｂ－14列出了题库管理控制层每个 Action 的入口（即调用 Action 的组件）、传递 Action 的 ActionForm，以及出口（即 Action 将请求转发到目标组件）。

表Ｂ－14：控制处理层

事件	Action	入口	ActionForm	出口
车位信息列表	showSpotAction	inlist.jsp	showSpotForm	spotlist.jsp
车位泊车记录详细信息	parkDetailAction	spotlist.jsp	parkDetailForm	parkdetail.jsp
修改车位状态	modifySpotAction	spotlist.jsp	DelQuestionForm	modifySpot.jsp
修改费用	modifyPriceAction	inlist.jsp	modifyPriceForm	modifyPrice.jsp

6.4.3 业务逻辑层设计

停车场状态管理业务逻辑层设计主要包括建立封装了车位信息的 bean:Spot.java，建立封装了泊车信息的 bean:Park.java， 以及封装答案的购物车 KnCart.java，完成将车位状态和相应的泊车信息存放数据库的操作，同时也提供了数据维护的操作等逻辑。模型组件见表 B-15。

表Ｂ－15：状态模块的模型组件

模型组件	描述
DBUtil	数据库的基本操作,为复用组件
Spotbean	封装了车位信息的 bean。
Parkbean	封装了详细泊车信息的 bean。

6.5 车辆信息管理模块设计

车位管理是管理员根据历史纪录调取查看各车位的占用记录,以及某一车辆对应的泊车记录。

6.5.1 表示层设计

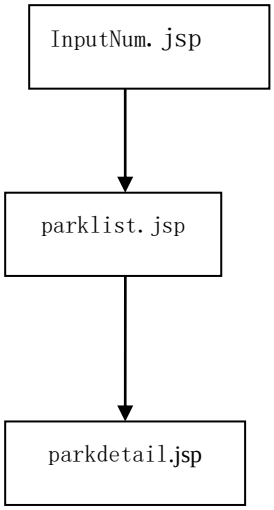
根据上述的功能介绍，总结出车辆信息管理功能的页面如表Ｂ－１６所示。

表Ｂ－１６：车辆信息管理模块的页面

页面	JSP	功能描述
车牌号	InputNum.jsp	输入需要检索的车牌号
停放记录列表	parklist.jsp	显示该车辆停放的历史记录列表
泊车详细信息	parkdetail.jsp	某次泊车的详细记录

注：问卷管理页面中也使用到了公共的页面，例如 left.jsp ,top.jsp,bottom.jsp 等。

各个界面基本流程图如图Ｂ－１８所示。



图Ｂ-１８：车辆信息管理的页面流程图示

6.5.2 控制层设计

车辆信息管理控制层主要负责进入车辆查询界面、进入车辆停放记录列表界面、进入某条泊车记录的详细信息等事件的流程控制。表Ｂ－１７ 列出了车辆信息管理控制层每个 Action 的入口（即调用 Action 的组件）、传递 Action 的 ActionForm，以及出口（即 Action 将请求转发到目标组件）。

表Ｂ－１7：控制处理层关系表

事件	Action	入口	ActionForm	出口
输入车牌号	InputNumAction	index.jsp	InputNumForm	InputNum.jsp
停放记录列表	parklist Action	InputNum.jsp	ExamDetailForm	parklist.jsp
泊车详细信息	parkdetailAction	parklist.jsp	AddExamForm	parkdetail.jsp
删除泊车记录	deleteAction	parkdetail.jsp	deleteForm	parklist.jsp

6.5.3 模型层设

车辆信息管理的业务逻辑主要是完成信息的查询，并完成相应数据库的操作。车辆信息管理的模型层主要是建立封装了车辆信息的 bean，以及封装了数据库操作的组件。模型组件见表 B-18。

表Ｂ－18：题库模块的模型组件

模型组件	描述
DBUtil	数据库的基本操作,为复用组件
Carbean	封装了问卷信息的 bean。

项 目 编 号	**200602006**
文 档 编 号	**12**
密　　级	内部

停车场管理系统详细设计

V1.0

软件工程实验

评　　审　　日　期：　2019 年 11 月 3 日

目 录

1. 导言

1.1 目的

该文档的目的是描述《停车场管理系统》项目的详细设计，其主要内容包括：

- 系统功能简介
- 系统详细设计简述
- 各个模块的三层划分
- 最小模块组件的伪代码

本文档的预期的读者是：

- 开发人员
- 项目管理人员
- 测试人员

1.2 范围

该文档定义了系统的各个模块和模块接口，但未确定单元的具体实现，这部分内容将在实现中确定。

1.3 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。

1.4 术语定义

Ｓｔｒｕｃｔ：一种框架体系结构。

1.5 引用标准

[1]《企业文档格式标准》 V1.1
[2]《软件详细设计报告格式标准》 V1.1

1.6 参考资料

[1]《实战ｓｔｒｕｃｔ》〔美〕Ｔｅｄ Ｈｕｓｔｅｄ
机械工业出版社

1.7 版本更新信息

本文档版本更新记录如表Ｃ-1：

表Ｃ-1 版本更新记录

修改编号	修改日期	修改后版本	修改位置	修改内容概述
000	2019.11.1	1.0	全部	初始发布版本

2 系统设计概述

根据《停车场管理系统》的概要设计，本系统按照功能角度分解，可以分为客户端子系统和管理端子系统。根据页面流的设计，管理端系统分为用户登录、用户管理、车位管理、车辆管理、停车场管理 5 个模块，他们的关系如图Ｃ－１，以下将分小节对各个部分分别进行详细设计。

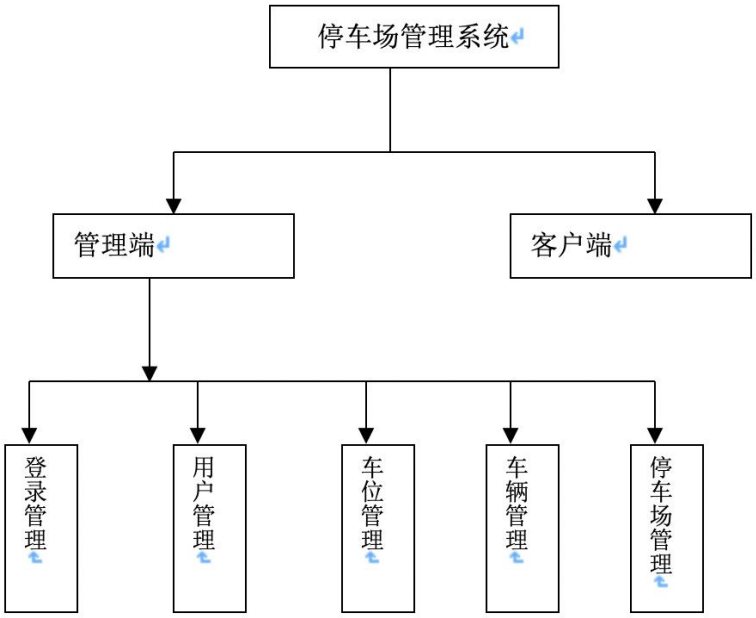


图 c-1： 模块设计

3 详细设计概述

由于本系统采用了基于ｓｔｒｕｃｔ体系结构的设计，即采用ＭＶＣ的三层设计模式，采用面向对象的Ｊａｖａ语言以及ＪＳＰ的脚本语言。所以，基本采用面向对象的设计方法。在整个的开发过程中，尽可能采用复用的原则，例如采用标签库，统一数据库的基本操作，统一结果显示等。

本文档的详细设计主要是按照Ｓｔｒｕｃｔ的ＭＶＣ的三个层次分别编制视图层、控制层和模型层模块的伪代码。为下一步的编码提供基础。伪代码(Pseudocode)是一种算法描述语言。使用伪代码的目的是为了使被描述的算法可以容易地以任何一种编程语言实现。因此，伪代码必须结构清晰，代码简单，可读性好，并且类似自然语言。

4 客户端模块的详细设计

客户端模块主要实现用户泊车的功能，客户端模块的Ｓｔｒｕｃｔ视图层、控制层和模型层三个层次的模块如表Ｃ－２所示。

表Ｃ－２：客户端的三层模块

视图	控制器	模型

JSP 组件	ActionFormBean	客户标签		
index.jsp InputNum.jsp spot.jsp path.jsp pay.jsp	 InputNumForm SpotForm PathForm PayForm	Struct Html 标签 Struct Bean 标签 Struct Logic 标签 用户自定义 CV 标签库的标签	EnAction SpotAction PathAction PayAction BackSpotAction BackPathAction BackPayAction BackExitAction	DBUtil CVDataBean

4.1 视图层

客户端管理视图层包括ＪＳＰ组件、Form 组件以及标签库等。共有 6 个ＪＳＰ文件，共有 4 个 Form 文件，标签库详见 12.1 描述。

4.2 控制层

客户端管理控制层共有 8 个 Action 文件。

4.3 控制层

客户端管理模型层共有 2 个 Bean 文件。

5 登录管理模块的详细设计

登录管理模块负责管理端用户的登录。管理端用户都是通过登录界面进入管理端的，用户输入用户名和密码进入管理界面首页，提供了进入功能面板的接口，并根据用户的权限在首页中列出相应的操作功能。

表Ｃ－６：登录管理模块的三层模块

视图			控制器	模型
JSP 组件	ActionFormBean	客户标签		

Login.jsp main.jsp center.jsp Top.jsp Left.jsp bottom.jsp	LoginForm	Struct Html 标签 Struct Bean 标签 Struct Logic 标签 用户自定义 CV 标签库的标签	LoginAction	DBUtil Permission Userbean

5.1 视图层

登录管理视图层包括ＪＳＰ组件、Form 组件以及标签库等。共有 6 个ＪＳＰ文件,共有 1 个 Form 文件。

5.2 控制层

登录管理控制层共有 1 个 Action 文件。

5.3 模型层

登录管理模型层共有 3 个Ｂｅａｎ文件。

5.4 分析及功能实现

功能描述：登陆页面需要用户名、密码进行登录。登录的同时对输入的用户名和密码进行验证，首先我们要保证用户名和密码不能为空。其次是对数据库的验证，系统在数据

库中搜索用户输入的用户名是否存在，若不存在，则提示出错。并且重新登录，系统不允许两个用户同时登录，该操作在一定程度上使系统的安全性有所提高。

登录主要代码位置：TestPark/src/DAL/login.java。主要封装了对登录数据的操，该类中有三个方法分别用于检查用户登录信息是否合法,根据用户编号和角色编号获取用户名和角色信息。

```
Public class Login {
    publicboolean checkLogin(String user_id,String user_pwd)
    {
        String sqlCmd="select count(*) from user where user_id=? and user_pwd=?";

       Object[] objList=new Object[2];
        objList[0]=user_id;
        objList[1]=user_pwd;
        String result=SQLUtil.excuteScalar(sqlCmd,objList).toString();
    if(result.equals("1"))
        {
            returntrue;
        }
        else {
            returnfalse;
        }
    }
    public String  getName(String user_id)
    {
        String sqlCmd="select user_name from user where user_id='"+user_id+"'";
        String result=SQLUtil.excuteScalar(sqlCmd, null).toString();
        return result;
    }
    public String  getSysLevel(String user_id)
    {
        String sqlCmd="select role_id from user where user_id='"+user_id+"'";
        String result=SQLUtil.excuteScalar(sqlCmd, null).toString();
        return result;
```

```
    }
}
```

6 用户管理模块的详细设计

在用户管理中系统管理员对用户进行增删改查，可以进行权限设置。具有不同权限的用户进入不同的主界面，界面左侧栏中的图标数有所不同，具体的图标与用户所具有的权限对应。在用户管理中可以增加或删除用户，编辑用户名，用户密码，修改用户权限等。

表Ｃ－７：用户管理模块的三层模块

视图			控制器	模型
JSP 组件	ActionFormBean	客户标签		
main.jsp userlist.jsp userdetail.jsp adduser.jsp	UserListForm UserDetailForm ModifyUserForm AddUserForm DelUserForm	Struct Html 标签 Struct Bean 标签 Struct Logic 标签 用户自定义 CV 标签库的标签	UserListAction UserDetailAction ModifyUserAction SubmitModifyAction AddUserAction SubmitAddAction DelUserAction	DBUtil Userbean

6.1 视图层

用户管理视图层包括 JSP 组件、Form 组件以及标签库等。共有 4 个ＪＳＰ文件，共有 5 个 Form 文件。

6.2 控制层

用户管理控制层共有 7 个 Action 文件。

6.3 模型层

用户管理模型层共有 3 个 Bean 文件。

6.4 分析及功能实现

功能描述：该模块是对系统信息的一个管理，添加角色输入编号和名称，后台数据库进行验证，若角色没有存在则添加数据成功，跳转到管理界面。管理角色信息可以根据用户的编号，角色名称进行查询操作、编辑、和删除操作。添加用户信息输入用户编号、名称、昵称、姓名、密码电话点击确定进行后台数据库的验证，若成功添加则跳转到管理界面。管理用户界面可根据用户编号、角色名称、用户名称、真实姓名进行查询、编辑和删除操作。

7 停车场状态管理模块的详细设计

停车场状态管理是对车位信息库和车辆信息库的维护和管理，对状态库的车位信息进行的增、删、改的功能，及提供泊车库中各车位的详细信息。

停车场状态管理的Ｓｔｒｕｃｔ视图层、控制层和模型层三个层次的模块如表Ｃ－８所示。

表Ｃ－８：简历管理模块的三层模块

视图			控制器	模型
JSP 组件	ActionFormBean	客户标签		
inlist.jsp spotlist.jsp parkdetail.jsp modifySpot.jsp modifyPrice.jsp	showSpotForm parkDetailForm DelQuestionForm modifyPriceForm	Struct Html 标签 Struct Bean 标签 Struct Logic 标签 用户自定义 CV 标签库的标签	showSpotAction parkDetailAction modifySpotAction modifyPriceAction	DBUtil Spotbean Parkbean

7.1 视图层

停车文件，共有 4 个

7.2 控制层

停车场状态管理控制层共有 4 个 Action 文件。

7.3 模型层

停车场状态管理模型层共有 3 个 Bean 文件。

7.4 分析及功能实现

添加停车位信息如图所示：

管理停车位信息如图所示：

功能描述：该模块是对车位的一个管理。点击添加车位信息，输入编号选择是 A 区还是 B 区，添加成功则会跳转到管理界面。管理界面可根据车位 ID、车位编号、所属区域、车位备注进行查询，还可对车位信息进行编辑和删除。

对车位数据的操作封装在：seat.java 类中。该类中主要封装了分页和对车位信息的增、

删、改、查等方法。首先获取车位信息列表，然后获取未分配的车位列表，根据查询条件获取获取分页后的信息列表，数据的更新、插入、删除，获取分页总数，根据查询条件获取分页总数。

```
publicclass Seat {
    public List<Object> getEntity()
        {
            String sqlCmd="select *from Seat";
            return DBUtil.SQLUtil.executeQuery(sqlCmd, null);
        }
        public List<Object> getNoUseSeat()
        {
            String sqlCmd="SELECT *FROM Seat WHERE seat_id NOT IN(SELECT seat_id
FROM card)";
            return DBUtil.SQLUtil.executeQuery(sqlCmd, null);
        }
        public List<Object> getEntity(int page)
        {
            int size=(page-1)*15;
            String sqlCmd="select *from Seat limit "+size+",15";
            return DBUtil.SQLUtil.executeQuery(sqlCmd, null);
        }
        public List<Object> getEntityByWhere(String sqlWhere,int page)
        {
            int size=(page-1)*15;
            String  sqlCmd="select  *from  Seat  where  "+sqlWhere+"  limit  "+
size+",15";
            return DBUtil.SQLUtil.executeQuery(sqlCmd, null);
        }
publicint deleteEntity(String seat_id)
        {
            String sqlCmd="delete from Seat where seat_id='"+seat_id+"'";
return DBUtil.SQLUtil.executeNonQuery(sqlCmd, null);
        }
public List<Object> getEntityById(String seat_id)
```

```
        {
            String sqlCmd="select *From Seat where seat_id='"+seat_id+"'";
return DBUtil.SQLUtil.executeQuery(sqlCmd, null);
        }
publicint updateEntity(String seat_id,String seat_num,String seat_section,String
seat_state,String seat_tag)
        {
            String sqlCmd="Update Seat set seat_num='" + seat_num +
"',seat_section='" + seat_section + "',seat_state='" + seat_state +
"',seat_tag='" + seat_tag + "' where seat_id='"+seat_id+"'";
return SQLUtil.executeNonQuery(sqlCmd, null);
        }
publicint insertEntity(String seat_id,String seat_num,String seat_section,String
seat_state,String seat_tag)
        {
            String sqlCmd="Insert into Seat values('" + seat_id + "','" +
seat_num + "','" + seat_section + "','" + seat_state + "','"+seat_tag+"')";
return SQLUtil.executeNonQuery(sqlCmd, null);
        }
publicboolean checkExist(String seat_id)
        {
            String sqlCmd="select count(*) from Seat where
seat_id='"+seat_id+"'";
if(1==Integer.parseInt(SQLUtil.excuteScalar(sqlCmd, null).toString()) )
            {
returntrue;
            }
returnfalse;
        }
        public Object getPageCount()
        {
            String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM Seat ";
            return SQLUtil.excuteScalar(sqlCmd, null);
        }
```

```
        public Object getPageCountByWhere(String sqlWhere)
        {
            String  sqlCmd="SELECT  CEIL(  COUNT(*)/15.0)  FROM  Seat  where
"+sqlWhere;
            return SQLUtil.excuteScalar(sqlCmd, null);
        }
            }
```

8 车辆信息管理模块的详细设计

车辆信息管理的Ｓｔｒｕｃｔ视图层、控制层和模型层三个层次的模块如表Ｃ－９所示。

表Ｃ－９：车辆信息管理管理的三层模块

视图			控制器	模型
JSP 组件	ActionFormBean	客户标签		
InputNum.jsp parklist.jsp parkdetail.jsp	InputNumForm ExamDetailForm AddExamForm deleteForm	Struct Html 标签 Struct Bean 标签 Struct Logic 标签 用户自定义 CV 标签库的标签	InputNumAction parklist Action parkdetailAction deleteAction	DBUtil Carbean

8.1 视图层

车辆信息管理视图层包括 JSP 组件、Form 组件以及标签库等。共有 3 个ＪＳＰ文件，共有 4 个 Form 文件。

8.2 控制层

车辆信息管理控制层共有 4 个 Action 文件。

8.3 模型层

车辆信息管理模型层共有 3 个 Bean 文件。

8.4 分析及功能实现

零时编号	零时编号	车牌号码	入场日期	出场日期	停车费用
20150925173007	JIIB23333	JIIB23333	2015-09-25 17:30:07	2015-11-28 21:29:26	20.0
20150925203021	JIIB23333	JIIB23333	2015-09-25 20:30:21	2015-11-28 21:29:26	15.0
20151201190239	JIIB11111	JIIB11111	2015-12-01 19:02:39	2015-12-01 19:04:24	3.0
20151201190418	JIIF22222	JIIF22222	2015-12-01 19:04:18	2015-12-01 19:04:28	3.0
20170424204735	54456213254	A6262626	2017-04-24 20:47:35	2017-04-26 23:39:43	153.0
20170427000148	111	46546	2017-04-27 00:01:48	2017-04-27 00:01:54	3.0
20170430170351	6516546546565	赣K9981	2017-04-30 17:03:51	2017-04-30 17:14:45	3.0
20170430171958	566	546	2017-04-30 17:19:58	2017-04-30 21:14:07	12.0

9 配置文件

Struct 应用采用两个基于 XML 的配置文件来配置应用，这两个配置文件为 Web.xml 和 strcut-config.Xml。Web.xml 适用于所有的 Java Web 应用,他是 Web 应用的发布描述文件，在 Java Servlet 规范中对它做了定义，对于 Struct 应用，在 Web.xml 文件中除了配置 Java Web 应用的常规信息，还应该配置和 Struct 相关的特殊信息。strcut-config.Xml 是 Struct 应用专有的配置文件，事实上，也可以根据需要给这个配置文件换为其它的文件名。

下面给出本项目中 Web.xml 配置文件和 strcut-config.Xml 配置文件的配置步骤和范围，在编码实施过程中，由开发经理指派专人负责所有文件的具体配置和协调。

9.1 Web.xml 配置文件

Web.xml 配置文件的配置步骤和范围

一、配置 Struct 的 ActionServlet

在 Web.xml 中配置 ActionServlet 主要包括声明 ActionServlet（即配置<Servlet>元素）、运行时环境的初始化配置（即配置<Servlet>元素的<init-param>的子元素）和指定 ActionServlet 可以处理哪些 URL（即配置<Servlet-mapping>元素）。例如

```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.structs.action.ActionServlet</servlet-class>
    <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/struct-config.xml</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

二、配置欢迎文件清单

当客户访问 Web 应用时，如果没有指定具体的文件名，Web 容器可以自动调用 Web 应用的欢迎文件。<welcome-file-list>元素用来设置欢迎文件清单。例如：

```
<welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>login.jsp</welcome-file>
</welcome-file-list>
```

三、配置错误处理

在系统运行过程中如果错误发生而Struct框架不能处理这种错误时,就将错误抛给Web容器,为了避免让用户看到原始的错误信息,可以配置<error-page> 元素及其子元素<error-code>和<exception-type>等.例如:

```
<error-page>
        <error-code>404</error-code>
        <location>/common/404.jsp</location>
</error-page>

<error-page>
        < exception-type>java.io.IOException</ exception-type >
        <location>/common/system_ioerror.jsp</location>
</error-page>
```

四、配置标签库

Strcut 框架提供了一些实用的客户化标签库,如果在应用中使用了标签库,那么必须在配置元素<taglib>中配置它们,它有两个子元素,<taglib-url>和<taglib-location>.例如:

```
<taglib>
    <taglib-url>/WEB-INF/struts-bean.tld</taglib-url>
    <taglib-location>/WEB-INF/struts-bean.tld </taglib-location>
</taglib>
```

13.2 strcut-config.Xml 配置文件

Struts 框架在启动的时候会读入其配置文件,根据它来创建和配置各种组件. Struts配置文件使得开发者可以灵活地组装和配置各个组件,提高了应用软件的可扩展性和灵活性.

<struts-config>有 8 个子元素:

- <data-sources>
- <form-beans>,
- <global-exceptions>,
- <global-forwards>,
- <action-mappings>,
- <controller>,
- <message-resources>,
- <plug-in>

在配置文件中,必须按照这个先后顺序来配置各个子元素.

一、<data-sources>元素

<data-sources>元素用来配置应用所需要的数据源,数据源负责建立和特定数据库的连接。<data-sources>元素包含零个、或者多个<data-source>子元素,它配置特定的数据源,可以包含多个<set-property>。例如:

```
<data-sources>
    <data-source key="sqlserver" type="org.apache.commons.dbcp.BasicDataSource">
        <set-property property="driverClassName" value="com.microsoft.jdbc.sqlserver.SQLServerDriver"
/>

        <set-property property="url"
value="jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=onlineCV" />
        <set-property property="maxActive" value="5" />
        <set-property property="username" value="sa" />
        <set-property property="password" value="xiaobo633601" />
        <set-property property="autoCommit" value="true" />
    </data-source>
</data-sources>
```

二、<form-beans>元素

<form-beans>元素用来配置多个 ActionForm Bean。<form-beans>元素包含零个或者多个<form-bean>子元素,每个<form-bean>元素又包含多个属性。例如:

```
<form-beans>

    <form-bean name="AddJobForm" type="org.apache.struts.action.DynaActionForm">

        <form-property name="JobName" type="java.lang.String" />
        <form-property name="PubDate" type="java.lang.String" />
        <form-property name="EndDate" type="java.lang.String" />
        <form-property name="Requirements" type="java.lang.String" />
        <form-property name="Description" type="java.lang.String" />
        <form-property name="JobNum" type="java.lang.Integer" />
    </form-bean>
    <form-bean name="JobDetailForm" type="org.apache.struts.action.DynaActionForm">
        <form-property name="JobID" type="java.lang.Integer" />
        <form-property name="JobName" type="java.lang.String" />
        <form-property name="PubDate" type="java.lang.String" />
        <form-property name="EndDate" type="java.lang.String" />
        <form-property name="Description" type="java.lang.String" />
        <form-property name="Requirements" type="java.lang.String" />
        <form-property name="JobNum" type="java.lang.Integer" />
    </form-bean>
</form-beans>
```

三、<global-exceptions>元素

<global-exceptions>元素用于配置异常处理,<global-exceptions>元素可以包含零个或者多个< exception>元素。例如:

```
<global-exceptions>
    < exception>
    key=” error.invalidlogin”
    path=” com/relogin.jsp”
    scope=” request”
    type=” netstore.framework.exceptions.InvalidLoginException” />
</global-exceptions>
```

四、<global-forwards>元素

<global-forwards>元素用来声明全局的转发关系,实现 Web 组件之间的相互转发。<global-forwards>元素由零个或者多个<forward>元素组成。<forward>元素用于将一个逻辑名映射到特定的 URL,这样 Action 类或者 JSP 文件无需指定实际的 URL,减弱了控制组件和视图组件之间的耦合。例如:

```
<global-forwards>
        <forward name="toaddjobpre" path="/position/addjob.jsp" />
        <forward name="tojobdetail" path="/position/jobdetail.jsp" />
        <forward name="tojobsearch" path="/position/jspsearchdetail.jsp" />
        <forward name="toaddjob" path="/position/joblist.do" />
        <forward name="towrong" path="/position/wrong.jsp" />
        <forward name="tojoblist" path="/position/joblist.jsp" />
        <forward name="tomodifyjob" path="/position/modifyjob.jsp" />
        <forward name="toaddjobvalueexampre" path="/position/selectexam.jsp" />
        <forward name="toaddjobvalueexam" path="/position/addjob1.jsp" />
        <forward name="toreturn" path="/position/addjob1.jsp" />
        <forward name="toaddjobexampre" path="/position/selectjobexam.jsp" />
        <forward name="toaddjobexam" path="/position/addjob1.jsp" />
        <forward name="todeljobexam" path="/position/addjob1.jsp" />
        <forward name="toaddfulljob" path="/position/joblist.do" />
        <forward name="tomodifyjobsubmit" path="/position/joblist.do" />
        <forward name="toupdate" path="/position/addjob1.jsp" />
        <forward name="todeljob" path="/position/joblist.do" />
        <forward name="toupdatemodifyjsp" path="/position/modifyjobsubmit.jsp" />
        <forward name="tomodifyjobvalueexampre" path="/position/modifyselectvalueexam.jsp"
/>
        <forward name="tomodifyjobexampre" path="/position/modifyselectexam.jsp" />
        <forward name="tomodifyjobvalueexam" path="/position/modifyjobsubmit.jsp" />
        <forward name="tomodifyjobexam" path="/position/modifyjobsubmit.jsp" />
```

```
        <forward name="tomodifyupdatejobexam" path="/position/modifyjobsubmit.jsp" />
        <forward name="tomodifydeljobexam" path="/position/modifyjobsubmit.jsp" />

    </global-forwards>
```

五、<action-mappings>元素

<action-mappings>元素包含零个或者多个<action>元素，<action>元素描述了从特定的请求路径到相应的 Action 类的映射。<action>元素可以包含多个<exception>和<forward>子元素，它们分别配置局部的异常处理及请求转发仅被当前的 Action 所访问。例如：

```
<action-mappings>
        <action attribute="JobDetailForm" name="JobDetailForm" path="/position/jobdetail"
scope="request" type="com.changjiangcompany.struts.action.JobDetailAction"></action>
        <action attribute="JobSearchForm" input="/position/joblist.jsp" name="JobSearchForm"
path="/position/jobsearch" scope="request"
type="com.changjiangcompany.struts.action.JobSearchAction"></action>

</action-mappings>
```

六、<controller>元素

<controller>元素用于配置 ActionServlet。例如

```
    <controller processorClass="share.MyRequestProcessor" />
```

七、<message-resources>元素

<message-resources>元素用于配置 Resource Bundle，Resource Bundle 用于存放本地化消息文本。例如：

```
<message-resources parameter="com.yourcompany.struts.ApplicationResources" />
```

八、<plug-in>元素

<plug-in>元素用于配置 Struts 插件，如果没有插件可以不写

项 目 编 号	**200602006**
文 档 编 号	**17**
密　　级	内部

停车场管理系统测试设计

V1.0

软件工程实验

评　　审　　日　期：　2019 年 11 月 10 日

目 录

1 导言

1.1 目的

该文档的目的是描述停车场管理系统项目客户端的系统测试设计，其主要内容包括：

- 测试总体设计
- 测试用例设计

本文档的预期的读者是：

- 项目管理人员
- 测试人员

1.2 范围

该文档为停车场管理系统客户端的系统测试设计,其中包括功能测试和性能测试的用例描述以及性能测试的测试脚本，为测试人员进行功能测试和性能测试提供标准和依据，以及详尽的测试步骤和方法。

1.3 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。

1.4 术语定义

LoadRunner

Mercury Interactive 的一个对 Windows 和 UniX 环境的负载测试工具。

功能性测试

按照系统需求定义中的功能定义部分对系统实行的系统级别的测试。

非功能性测试

按照系统需求定义中的非功能定义部分（如系统的性能指标，安全性能指标等）对系统实行的系统级别的测试。

测试用例

测试人员设计出来的用来测试软件某个功能的一种情形。

1.5 引用标准

[1]《企业文档格式标准》
北京长江软件有限公司
[2]《软件测试设计报告格式标准》
北京长江软件有限公司软件工程过程化组织

1.6 参考资料

[1]《LoadRunner 使用手册》
北京长江软件有限公司编制
[2]《停车场系统客户端需求说明》
软件工程实验编制
[3]《软件测试技术概论》
古乐 史九林编著 /清华大学出版社
[4]《软件测试：第二版》
Paul C.Jorgensen 著/机械工业出版社

1.7 版本更新信息

本文档的更新信息如表Ｆ-1.

表Ｆ-1 版本更新记录

修改编号	修改日期	修改后版本	修改位置	修改内容概述
000	2019．11．5	1.0	全部	初始发布版本

2.测试设计

由于本次测试主要是针对需求进行的系统测试，包括功能测试和性能测试的技术，功能

测试是执行指定的工作流程，性能测试是将功能测试过程中的单独用户改为 20 人同时执行以验证系统的性能。

2.1 测试范围

系统测试依据的系统的应用工作流：

1) 余量查询：在主页面实时显示当前停车场的空闲车位数。
2) 车牌检索：从主页面的输入框中输入指定格式的车牌号后，点击“查找”，系统开始检索相关信息。若数据库中没有保存相关记录，则返回检索失败提示信息，并自动返回主页面，由用户重新输入；若检索成功，进入“车辆信息”页面。
3) 车辆信息页面：用户即可查看该车辆信息。信息列表中显示开始车牌号、泊车位号、开始泊车时间、泊车时长、当前费用、路径显示、结算状态、缴费等。点击”上一步”按钮返回主页面，当点击”路径显示”按钮时进入“最优路径”页面，当点击“缴费”按钮时进入“支付订单”界面。
4) 最优路径页面：在最优路径页面下方为路径显示区，从停车场各个指定入口到达该泊车位的最优路径会显示在页面该区域中。点击”上一步”按钮返回车辆信息页面。
5) 支付订单页面：在支付页面上方的表格中显示了车牌号、车位号、起始泊车时间、泊车时长、费用信息。页面下方的支付方式选择框中，由用户勾选微信支付或支付宝支付。勾选好方式后，当点击”立即支付”时进入“微信支付”页面或“支付宝支付”页面。
6) 微信支付页面：用户在该页面完成支付，点击”完成”进入“支付成功”页面。
7) 支付宝支付页面：用户在该页面完成支付，点击”完成”进入“支付成功”页面。
8) 支付成功页面：若支付状态成功，则该页面显示订单已完成的提示信息；若返回状态失败，则该页面显示订单失败的提示信息。5 秒后进入主页面继续下一轮查询。

2.2 测试覆盖设计

由于本次测试是系统测试，测试的依据是系统需求，测试的设计应该满足对需求的覆盖，所以，采用的测试方法主要是黑盒测试，包括等价类划分（有效测试和无效测试）、边界值和错误猜测法等。表 F－ 2 就是测试用例覆盖矩阵。

表 F-2：测试用例功能/性能覆盖矩阵

序号	功能项	测试用例	优先级
01	所有基本页面的链接正确	TestCase-FUNC-01	中

02	所有页面的转移正确	TestCase-FUNC-02	中
03	余量信息正确	TestCase-FUNC-03	高
04	正常查询车辆信息的流程－有（无）效数据	TestCase-FUNC-04	高
05	车辆详细信息正确	TestCase-FUNC-05	高
06	最优路径信息显示正确	TestCase-FUNC-06	高
07	支付订单信息正确	TestCase-FUNC-07	高
08	正常支付的流程－有（无）效数据	TestCase-FUNC-08	高
09	访问安全性	TestCase-Perf-1	高
10	并发访问的性能测试	TestCase-Perf-2	高

3.测试用例

按照上面的测试矩阵表，设计相应的测试用例如下。

3.1 用例一：基本页面的链接

这个测试用例的测试编号是 TestCase-FUNC-01，测试内容是验证所有基本页面链接的正确性，同时所有的页面都按照需求有正确的显示。表 F-3 是这个测试用例的具体设计。

表 F-3：TestCase-FUNC-01 测试用例

<table>
<tr><td colspan="3">测试项目名称： 停车场管理系统－客户端</td></tr>
<tr><td>测试用例编号： TestCase-FUNC-01</td><td rowspan="2">测试人员：
蒋芷昕</td><td rowspan="2">测试时间：
2019/11/1</td></tr>
<tr><td>测试项目标题：所有基本页面的正确链接</td></tr>
<tr><td colspan="3">测试内容：
验证网站首页所有链接有效；
验证网站首页中图片能正确装入；
验证网站首页中的超连接的连接页面与页面中指示（或 图示）相符；
验证网站中各个页面的显示的信息都符合需求。</td></tr>
<tr><td colspan="3">测试环境与系统配置：
详见《测试计划》</td></tr>
<tr><td>测试输入数据</td><td colspan="2">脚本见 TC-F-01.c （见测试开发文档）</td></tr>
<tr><td colspan="3">测试次数：每个测试过程做 2 次。</td></tr>
<tr><td colspan="3">预期结果：
可以正确显示图片，每个链接有效，超连接的连接页面与页面中指示（或 图示）相符。</td></tr>
</table>

测试过程： 对于首页每个操作链接，点击进入。察看链接的页面是否相符合。 对于首页每个链接，点击看能否进入相应页。
测试结果：每个链接有效，超链接的链接页面与页面中指示（或图示）相符。
测试结论： 基本页面的链接与显示正确。
实现限制：无
备注：无

3.2 用例二：页面转移的正确性

这个测试用例的测试编号是 TestCase-FUNC-02，测试内容是测试所有转移页面链接的正确性，同时所有的页面都按照需求有正确的显示。表 F-4 是这个测试用例的具体设计。

表 F-4： TestCase-FUNC-02 测试用例

测试项目名称： 停车场管理系统－客户端		
测试用例编号：CV-FUNC-02	**测试人员：** 蒋芷昕	**测试时间：** 2019/11/1
测试项目标题：转移页面的正确性		
测试内容： 验证网站每页输入“转到 ”的输入框能正确处理输入		
测试环境与系统配置： 详见《测试计划》		
测试输入数据	异常数据：0，1．4，6 正常数据：1	
测试次数：每个测试过程做 2 次。		
预期结果： 对于正常数据能够转到相应页面，异常数据能够报错。		
测试过程： 对于首页“转到 ”的输入框，依次输入相应数据。		
测试结果： 对于正常数据能够转到相应页面；对于异常数据，能够给出错误提示，并且刷新当前页面。		
测试结论：对于正常数据能够转到相应页面，异常数据能够报错。		
实现限制：无		
备注：无		

3.3 用例三：余量信息

这个测试用例的测试编号是 TestCase-FUNC-03，测试内容是测试主页面中所显示的余量车位个数的正确性。表 F-5 是这个测试用例的具体设计。

表 F-5：TestCase-FUNC-03 测试用例

测试项目名称： 网上招聘系统－客户端		
测试用例编号： TestCase-FUNC-03	**测试人员：** 蒋芷昕	**测试时间：** 2019/11/1
测试项目标题： 主页面车位余量信息的显示		
测试内容： 验证主页面是否正确显示当前停车场车中的空闲车位个数		
测试环境与系统配置： 软件环境：Micrsoft windowXP Professional 硬件环境：P4 1.7GHz CPU + 1.7GHz 512MB 内存 网络环境：6 人共享 1M 带宽		
测试输入数据	无	
测试次数： 刷新 2 次		
预期结果： 主页面正确显示当前停车场车中的空闲车位个数。		
测试过程： 登录系统客户端，并刷新。		
测试结果： 主页面正确显示当前停车场车中的空闲车位个数。		
测试结论： 识别正常，数据更新及显示正常。		
实现限制： 无		
备注： 无		

3.4 用例四：完整的车辆查询测试

这个测试用例的测试编号是 TestCase-FUNC-04，测试内容是测试泊车用户在正常(非正常)输入的条件下是否可以成功查询，同时所有的页面都按照需求有正确的显示。表 F-6 是这个测试用例的具体设计。

表 F-6：TestCase-FUNC-04 测试用例

测试项目名称： 停车场管理系统－客户端		
测试用例编号： TestCase-FUNC-04	**测试人员：** 蒋芷昕	**测试时间：** 2019/11/1
测试项目标题： 正常查询流程的功能测试		

测试内容： -- 验证填写特殊车牌号时是否能正确执行查询操作。 -- 验证不填写或格式错误时是否在提交时正确显示错误提示。 -- 验证支付失败（中止支付）时是否能正确显示错误提示。 -- 验证支付成功后再次检索，车辆缴费状态是否更新。	
测试环境与系统配置： 软件环境：Micrsoft windowXP Professional + Micrsoft IE6.0 硬件环境：P4 2.8GHz CPU + 2.79GHz 512MB 内存 网络环境：10 人共享 1M 带宽 详见《测试计划》	
测试输入数据	脚本 TC-F-02.c（见测试开发文档）
测试次数：每个测试过程做 2 次。	
预期结果： -- 用户不填写或者格式错误时，提交时应有错误提示，页面刷新，可重新输入。 -- 用户填写有效的特殊牌照时能正确运行，进入下一页面。 -- 用户因中止支付或其他原因导致支付失败时能正确显示错误提示。 --支付成功后再次检索该车辆信息，缴费状态显示已更新。	
测试过程： 登陆客户端； 输入车牌号； 点击“查找”； 根据提示信息选择“缴费”； 执行支付操作； 支付成功后再次查找同一车牌号； 查看其“结算状态”；	
测试结果： -- 用户不填写或者格式错误时，提交时有错误提示，页面刷新，可重新输入。 -- 用户填写有效的特殊牌照时能正确运行，进入下一页面。 -- 用户因中止支付或其他原因导致支付失败时能正确显示错误提示。 --支付成功后再次检索该车辆信息，缴费状态显示已更新。	
测试结论： 车辆查询流程正常，对于正常数据能够正常跳转下一步操作；对于异常数据能够给出错误提示，跳转到对应处理页面。	
实现限制：无	
备注：无	

3.5 用例五：车辆详细信息页面

这个测试用例的测试编号是 TestCase-FUNC-05，测试内容是测试车辆详细信息页面中所显示的信息的正确性。表 F-7 是这个测试用例的具体设计。

表 F-7：TestCase-FUNC-05 测试用例

测试项目名称：停车场管理系统－客户端		
测试用例编号：TestCase-FUNC-05	**测试人员：**蒋芷昕	**测试时间：**2019/11/1
测试项目标题：车辆详细信息查询		
测试内容： 验证页面是否正确显示了车牌号、泊车位号、开始泊车时间、泊车时长、当前费用、路径显示、结算状态、缴费几项。 验证车辆详细信息页面上的信息是否与输入要查询的车牌信息相符。		
测试环境与系统配置： 软件环境：Micrsoft windowXP Professional + Micrsoft IE6.0 硬件环境：P4 1.7GHz CPU + 512MB 内存 网络环境：6 人共享 1M 带宽 详见《测试计划》		
测试输入数据	无	
测试次数：应至少测试 3 个不同的车牌号，并随机进行。		
预期结果： 页面正确显示了车牌号、泊车位号、开始泊车时间、泊车时长、当前费用、路径显示、结算状态、缴费几项。 车辆详细信息页面上的信息与输入要查询的车牌信息相符。		
测试过程： -- 登录系统，输入车牌号。		
测试结果： 跳转到车辆详细信息页面，页面正确显示了该车辆的车牌号、泊车位号、开始泊车时间、泊车时长、当前费用、路径显示、结算状态、缴费几项。		
测试结论：页面信息显示正常。		
实现限制：无		
备注：无		

3.6 用例六：最优路径页面

这个测试用例的测试编号是 TestCase-FUNC-06，测试内容是测试最优路径页面中车位到达每个指定点的路径有正确的显示。表 F-8 是这个测试用例的具体设计。

表 F-8：TestCase-FUNC-06 测试用例

测试项目名称：停车场管理系统－客户端		
测试用例编号：TestCase-FUNC-06	**测试人员：**	**测试时间：**

测试项目标题：最优路径页面的功能测试	蒋芷昕	2019/11/1
测试内容： -- 当存在可通行路径时，验证页面是否正确显示了所有指定入口到达车位的可通行路径。 -- 当不存在可通行路径时，验证页面是否正确给出错误提示。 -- 验证显示的路径是否为最优路径。		
测试环境与系统配置： 详见《测试计划》		
测试输入数据	无	
测试次数： 应至少测试 3 个不同的车位，并随机进行。		
预期结果： 能够正确显示所有最优路径，否则能够给出错误提示。		
测试过程： -- 登录系统，输入车牌号，点击“查找”。 -- 在车辆详细页面点击“路径显示”。		
测试结果： 最优路径页面正确显示所有指定位置到达该车位的可通行路径，且均为最优路径。		
测试结论：最优路径显示正确，能给出相应提示。		
实现限制：无		
备注：无		

3.7 用例七：支付订单页面

这个测试用例的测试编号是 TestCase-FUNC-07，测试内容是测试支付订单页面有正确的显示。表 F-9 是这个测试用例的具体设计。

表 F-9：TestCase-FUNC-07 测试用例

测试项目名称： 网上招聘系统－客户端		
测试用例编号：TestCase-FUNC-07	**测试人员：** 蒋芷昕	**测试时间：** 2019/11/1
测试项目标题：支付订单页面测试		
测试内容： -- 验证页面是否正确显示了车牌号、车位号、起始泊车时间、泊车时长、费用信息。 -- 验证页面下方是否正确显示了支付方式选择框、“微信支付”选项、“支付宝“支付选项、“立即支付”按钮。 -- 验证页面的选项是否能正确勾选。		
测试环境与系统配置： 详见《测试计划》		

测试输入数据	-- 勾选支付方式“微信支付”，点击“立即支付”。 -- 勾选支付方式“支付宝支付”，点击“立即支付”。
测试次数：每个测试过程做 2 次。	
预期结果： -- 页面正确显示了车牌号、车位号、起始泊车时间、泊车时长、费用信息。 -- 页面下方正确显示了支付方式选择框、“微信支付”选项、“支付宝“支付选项、“立即支付”按钮。 -- 页面的选项能正确勾选。	
测试过程： 见测试输入数据。	
测试结果： -- 页面正确显示了车牌号、车位号、起始泊车时间、泊车时长、费用信息。 -- 页面下方正确显示了支付方式选择框、“微信支付”选项、“支付宝“支付选项、“立即支付”按钮。 -- 页面的选项能正确勾选，点击“立即支付”能够正常跳转到对应界面。	
测试结论： 订单详细信息页面功能正常。	
实现限制：无	
备注：无	

3.8 用例八：完整的支付测试

这个测试用例的测试编号是 TestCase-FUNC-08，测试内容是测试用户在选择线上支付时在正常（非正常）支付状态下是否可以成功支付，同时所有的页面都按照需求有正确的显示。表 F-10 是这个测试用例的具体设计。

表 F-10：TestCase-FUNC-08 测试用例

测试项目名称：停车场管理系统－客户端		
测试用例编号：TestCase-FUNC-08	**测试人员：** 蒋芷昕	**测试时间：** 2019/11/1
测试项目标题：正常支付流程的功能测试		
测试内容： -- 验证用户点击“立即支付”后，是否能相应跳转到微信程序中的支付界面，或支付宝应用中的支付界面。 -- 验证确认付款并成功后，是否能给出支付成功的提示信息，并跳转到“支付成功”页面，再跳转到主页面。 -- 验证中止付款后，是否能够给出支付失败的提示信息，并跳转到“支付失败”页面，再跳转到“支付订单”页面。 -- 验证支付成功回到主页面时，输入框中的上条记录是否已清空。		
测试环境与系统配置： 详见《测试计划》		

测试输入数据	无
测试次数：每个测试过程做 2 次。	
预期结果： -- 用户点击“立即支付”后，能相应跳转到微信程序中的支付界面，或支付宝应用中的支付界面。 -- 确认付款并成功后，能给出支付成功的提示信息，并跳转到“支付成功”页面，再跳转到主页面。 -- 中止付款后，够给出支付失败的提示信息，并跳转到“支付失败”页面，再跳转到“支付订单”页面。 -- 支付成功回到主页面时，输入框中的上条记录已清空。	
测试过程： 见测试内容	
测试结果： -- 用户点击“立即支付”后，能相应跳转到微信程序中的支付界面，或支付宝应用中的支付界面。 -- 确认付款并成功后，能给出支付成功的提示信息，并跳转到“支付成功”页面，再跳转到主页面。 -- 中止付款后，够给出支付失败的提示信息，并跳转到“支付失败”页面，再跳转到“支付订单”页面。 -- 支付成功回到主页面时，输入框中的上条记录已清空。	
测试结论：页面跳转正常，能够给出相应提示信息。	
实现限制：无	
备注：无	

3.9 用例九：系统安全性测试

这个测试用例的测试编号是 TestCase-Perf-1，测试内容是测试用户进行非正常访问时系统的异常处理。表 F-11 是这个测试用例的具体设计。

表 F-11：TestCase-Perf-1 测试用例

测试项目名称：停车场管理系统－客户端		
测试用例编号：TestCase-Perf-1	**测试人员：** 冯郎	**测试时间：** 2019/11/1
测试项目标题：非正常页面访问的测试		
测试内容： 直接访问后续页面而不通过首页。		
测试环境与系统配置： 详见《测试计划》		

测试输入数据	登录系统
测试次数：每个测试过程做 2 次。	
预期结果： 如果未登录直接访问后续页面，那么会无法进入，有错误提示框出现。或者重定向到首页。	
测试过程： 登录系统。	
测试结果：如果未登录直接访问后续页面，那么会被拒绝访问，被重定向到登录界面；如果已经登录访问后续页面，则正常访问。	
测试结论：对于非正常页面的访问，能做到拒绝访问；对于正常访问，可以允许访问。	
实现限制：无	
备注：无	

3.10 用例十：系统的并发性测试

这个测试用例的测试编号是 TestCase-Perf-2，测试内容是测试 20 个用户同时访问系统时,系统的性能情况。表 F-12 是这个测试用例的具体设计。

表 F-12：TestCase-Perf-2 测试用例

测试项目名称：停车场管理系统－客户端		
测试用例编号：TestCase-Perf-2	**测试人员：** 冯朗	**测试时间：** 2019/11/1
测试项目标题：并发访问的性能测试		
测试内容： 20 个用户同时访问系统时,系统的性情况。		
测试环境与系统配置： 详见《测试计划》		
测试输入数据	1、生成单用户正常访问脚本 2、对脚本参数化 3、在脚本中增加事务、集合点，以每次点击“下一步”或“提交”按钮为界限 脚本见“TC-P-01.c”(详见测试开发文档)	
测试次数：每个测试过程做 2 次。		
预期结果： 有错误提示，或者无。		

测试过程： -- 使用 loadrunner 的 visual user generator 录制基本的用户脚本 -- 将用户在录制脚本时填写并提交的一些数据参数化，另外在提交数据的函数前面设置集合点 -- 设置运行环境，独立运行修改后的脚本，根据产生的错误修改直至脚本正确 -- 打开 loadrunner 的 controller 新建一个运行场景。在运行场景中新建一个虚拟用户组，设置其中的虚拟用户数目为 20。所有虚拟用户的运行脚本为刚刚录制并修改的脚本虚拟用户的 load generator 为本机 -- 启动 ip 欺骗，之后将 load generator 的状态由 down 改变为 ready -- 设置场景的 schedual 为同时启动所有用户，其它使用默认设置 -- 设置结果保存路径 -- 设置集合点，选择当 20 用户全部到达集合点时释放虚拟用户，时间间隔为 1 分钟 --设置 runtime settings，均采用默认设置 --运行场景-脚本见“性能测试脚本.c” --打开 loadrunner 的 analysis 分析场景的运行结果
测试结果： 多用户并发操作时候，都能正常执行功能，响应足够迅速。
测试结论：多用户并发操作的时候，响应速度差强人意。
实现限制：无
备注：无