城市共享停车管理系统

体系结构设计

1.0

2018-12-23

田宇

2016212011

软件工程导论

2018 秋季学期

版本历史

日期	描述	作者	评论
2018/12/23	1.0	田宇	第一版，对学生运动成绩管理系统的体系结构做了设计和说明。

文档审批

签名	姓名	标题	日期

目　录

第一章 项目相关信息

为了解决停车难、车为空忙不均等的情况；为了推动单位大院、居民区等开放空车位。我们设计一个城市共享停车管理系统，使市民享受到便捷的停车服务并且提高停车场利用率实现更高的创收。

1.1.目的

1) 详细阐述城市共享停车管理系统的功能、性能要求和服务设计
2) 为司机，停车场管理者等多个角色分别制定相应的需求，并分析各个角色功能间的联系，力求完整一致。
3) 对城市共享停车管理系统的实现作使命描述，帮助客户判断所规定的系统是否符合要求，如何修改才能符合要求。
4) 对空车位以及司机分配问题进行科学管理，同时减轻停车场管理者的工作量，从根本上提高车位利用率和工作人员的工作效率。

1.2.范围

本系统为城市共享停车管理系统，经过对系统需求的研究分析，该系统应具备以下功能：①车位的录入；②车位状态的获取；③司机查询目的地车位情况；④空车位的预定；⑤目的地导航；⑥停车时间计算；⑦停车费计算；⑧出示二维码结束计费

针对不同类型用户，本系统分为三个功能块：①司机客户端；②停车场管理员端；③系统管理员端。

1.3.定义

名称	定义
城市共享停车管理系统/系统	本文档负责的软件系统
司机	注册本系统且使用本系统的司机
停车场管理员	拥有并注册使用本系统的停车场管理员
系统管理员	在使用本系统的管理此系统的公司中的专业人员，拥有最大权限
计费规则	由系统默认的或由停车管理员制定的停车费收费标准
司机个人信息	司机的性别，用户名，手机号码，车牌号，经常使用的停车场等，
停车场管理员信息	停车场地理位置，停车场名字，管理员用户名，停车场营收情况等

1.4.综述

本文档共分为五个章节，至此第一章节引言部分结束。第二章节是对系统需求的总体描述，从产品愿景、产品功能、用户特征、一般约束、假设和依赖几个方面对系统做了描述；第三章节是对系统的功能性需求分析，也是本文档主体部分，包括周境分析、功能需求以及用例分析；第四章节是对系统的非功能性分析，从质量要求、工程要求和其他要求出发；最后为需求变更分析。

第二章 体系结构需求

2.1.关键指标

1. 司机客户端

   可以根据输入的目的地寻找最近的空停车位并且可以选择开始导航或者预约停车位；如果预约停车位之后便开始计时，计算停车费用。

2. 停车场管理员端

   可以注册停车场，输入停车位数量，查看当前停车位状态，查询每月收入等。

3. 系统管理员端

   管理司机，停车场管理员，系统后台等信息的操作。

2.2.体系结构用例

2.2.1.用户角色

1) 司机

i. 年龄跨度广，所以需要新手引导以及功能提示

ii. 大量司机行为存在随机性，并且在某些时候比如周末，停车需求暴增，需要后台服务器能够处理大量并发的服务，维持相应的稳定性。

2) 停车场管理员

i. 停车场管理员可能覆盖的年龄比较广，因此需要新手引导以及功能提示。

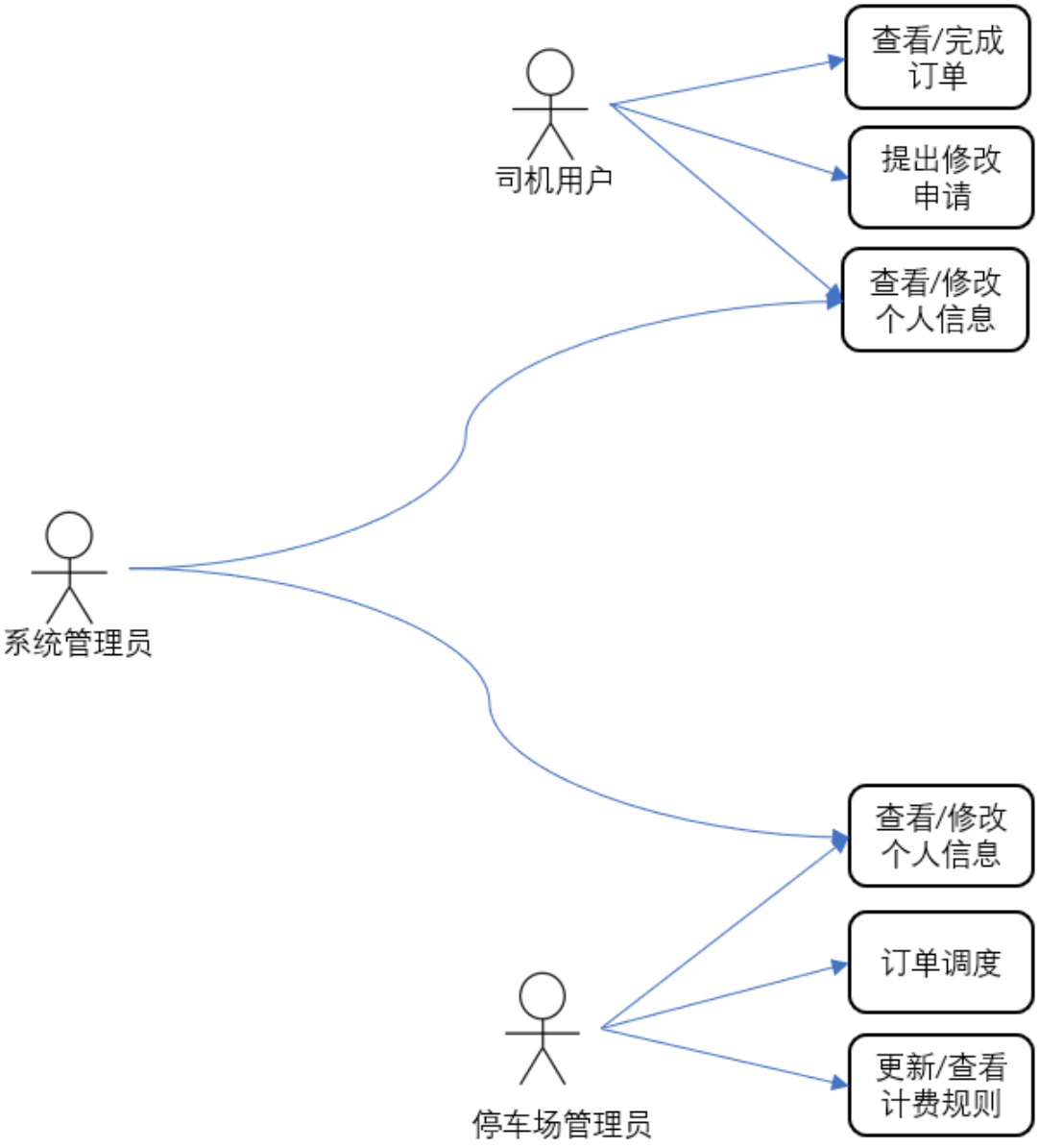
ii. 考虑到停车场基本都具有相应的岗亭，因此考虑停车场管理员端使用电脑端而不是不稳定的手机端，并且应具备上手易，操作简单的特点，所以考虑为管理员配备扫码枪以及车牌识别服务。

iii. 在面对未使用系统的司机时应该具有“扫描车牌即可停\取车“的功能，将车牌识别放在入口，进出停车场时即可进行记录计费，并应该提供多种支付方式。

3) 系统管理员

i. 需要拥有最高系统权限，并且可以调用丰富的接口进行错误处理，可以查看任何信息（除了用户名密码以及其他隐私信息）。

2.2.2.用户功能用例图



2.2.3.用户功能用例分析

A. 司机客户端

可以根据输入的目的地寻找最近的空停车位并且可以选择开始导航或者预约停车位；如果预约停车位之后便开始计时，计算停车费用。

B. 停车场管理员端

可以注册停车场，输入停车位数量，查看当前停车位状态，查询每月收入等。

C. 系统管理员端

管理司机，停车场管理员，系统后台等信息的操作。

2.3.各相关方对体系结构的要求

1.用户:

1) 司机：可以根据输入的目的地寻找最近的空停车位并且可以选择开始导航或者预约停车位；如果预约停车位之后便开始计时，计算停车费用。

2) 停车场管理员：可以注册停车场，输入停车位数量，查看当前停车位状态，查询每月收入等。

3) 系统管理员：修改用户的权限和信息，可以对用户的账号进行管理，可增加或删除用户。

2.开发技术人员:

系统有明确的开发需求，客户不增添和本系统无关的任意功能，不添加不可能实现的任意功能，不在职责之外添加任务，保证规定时间之内任务能够完成。

3.客户:

项目需要保证在规定的合同约定的时间内能定时定量地完成进度，保证进度不拖拉，同时在财政预算的方面没有多余的预算。

4.项目经理:

保证任务能够及时并合理地进行分解，分发给每一个开发团队独立工作。保证每个团队都能按质按量按时地完成任务。

2.4.约束条件

1) 常用的信息管理系统采用 B/S 开发模式，系统应采用网站开发的基本技术。
2) Web 服务器部署于 Linux 系统之上的系统一般运行稳定性高。
3) 服务器系统硬件的配置能支持服务器高效稳定的运行。
4) 为了系统将来的可扩展性，系统在硬件的使用上需尽可能减少针对性。整个 系统也应尽量减少各模块间的调用，尽量做到松耦合。
5) 数据文件、系统配置文件应当安全可靠的存储。
6) 系统的数据需要具有足够的可靠性，才会保证系统正常运行。

2.5.非功能需求

2.5.1.性能

响应时间：系统响应时间应小于 3 秒钟。

吞吐量：单位时间（按 1 秒钟计）内吞吐量应不低于 10000

并发用户数：系统支持并发用户数最少为 30000。

资源利用率：最大限度提高资源利用率为 4。

2.5.2.可靠性

1）采用面向对象的系统开发方法;

2）选用合适的开发工具;

3）采用结构化的程序设计方法;

4）程序设计风格化;

5）经过严格的测试;

6）设置必要的错误处理和错误陷阱;

7）选择典型的单位试运行。

2.5.3.易用性

系统应当具有针对教师的详细的用户使用手册，方便教师学习使用。

系统应当具有针对学生的概括性的用户使用手册，方便学生学习使用。

系统应当具有在线帮助界面等部分，方便用户学习操作。

系统应当足够简洁明了，使大部分用户能够自己学会使用本系统。

2.5.4.密安性

系统应当能够保证用户信息不泄露，系统配置文件和数据库存储文件应该进行加密处理。

2.5.5.可维护性

系统应该能够容易诊断出存在的缺陷和失效原因，容易识别出待修改部分的可能性或能力。开发人员应当记录开发过程日志，以便备份追踪。

系统应当能够保证开发过程的代码、设计和文档容易修改，代码应当结构清晰且有较详细的注释，设计文档详细明确。

2.5.6.可移植性

系统应当能够保证在 Windows、Linux 等多平台上容易安装和部署。

系统应当能够和其他软件共存于一个平台上，存在冲突的软件不超过。

系统应当能够容易地被卸载，也容易被更高版本的系统替换。

2.6.风险

2.6.1.内部风险

1) 团队内部分工定位不明确，信息沟通不透明
2) 需求变更：包括客户提出新需求、原有需求发生变更、原有需求不够清晰
3) 系统变更：包括平台核心功能升级、已上线系统的升级、新模块上线
4) 技术变更：包括核心技术的更换，或者新技术、新工具的引入
5) 制定计划时有部分任务或事项被遗漏
6) 相关任务或事项没有被具体细化，或相关技术方案尚未被验证通过，或存在尚未明确的任务
7) 资源不足——人员变动，或者工作量超过实际人员的负荷能力，或对某项工作难度估计不足
8) 相关任务的进度推迟，或质量无法满足阶段性要求，需要返工或调整计划，或测试不完整、不充分

9) 系统性能无法满足需求

10) 无原则的对客户要求作出承诺

11) ……

2.6.2.外部风险

1) 与项目有关的客户方组织架构发生变动，例如换了项目的主管部门，或者项目经理

2) 客户对系统或需求重新定位

3) 相关计划或方案与客户沟通不充分，或尚未通过评审

4) 与客户或用户沟通不及时，或信息不完整，或客户内部对相关工作理解不统一

5) 需要客户配合的相关工作缺少相关的指导或规范，做法不统一

6) 客户系统环境或用户使用环境复杂，不统一

7) ……

第三章 解决方案

3.1.相关的体系结构模式

分层架构模式中的组件被组织成水平层，每个层执行应用中的一个具体角色(比如，表示逻辑或业务逻辑)。尽管分层架构模式不会具体要求模式中必须存在的层的数目和类型，但大多数分层架构由四个标准层：表示(presentation)，业务(business)，持久(persistence)和数据库(database)。

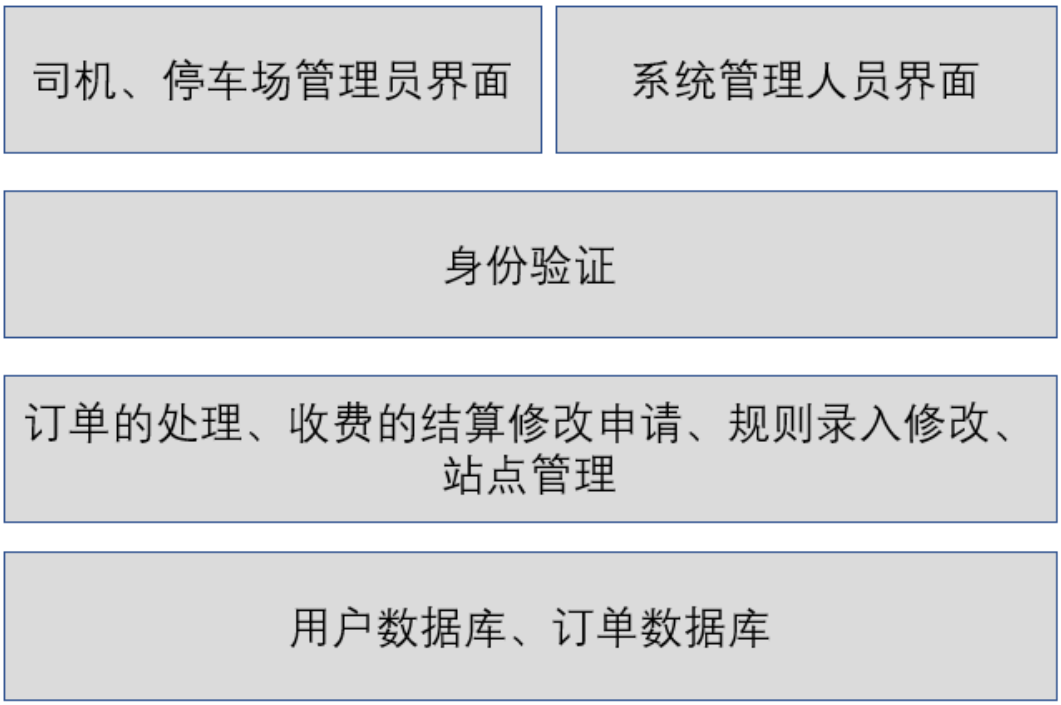
层次架构模式的每层有一个具体角色和职责。比如，一个表示层用于处理所有用户接口和浏览通信逻辑，而一个业务层会用于执行具体的和要求(request)相关的业务规则。架构的每个层会形成满足具体业务要求的一个抽象。举个例子，表示层不必知道或不必关心如何得到用户的数据，而只需要以特定的格式展示屏幕上的信息。同样地，业务层不必关心如何格式化用户数据来展示在屏幕上，甚至用户数据从哪里来；只需从持久层获得数据，用业务逻辑处理数据（比如，计算值或聚集数据），最后把信息传到表示层。

分层架构模式最重要的特性之一就是组件的关注分离(separation of concerns)。一个层中的组件只处理和该层相关的逻辑。比如，表示层中的组件只处理表示逻辑，而业务层中的组件只处理业务逻辑。这种组件分类使构建有效角色和职责模型变得容易，也有益于开发，测试，掌控和维护，这是因为这个架构有着定义明确的组件接口和限制的组件视野。

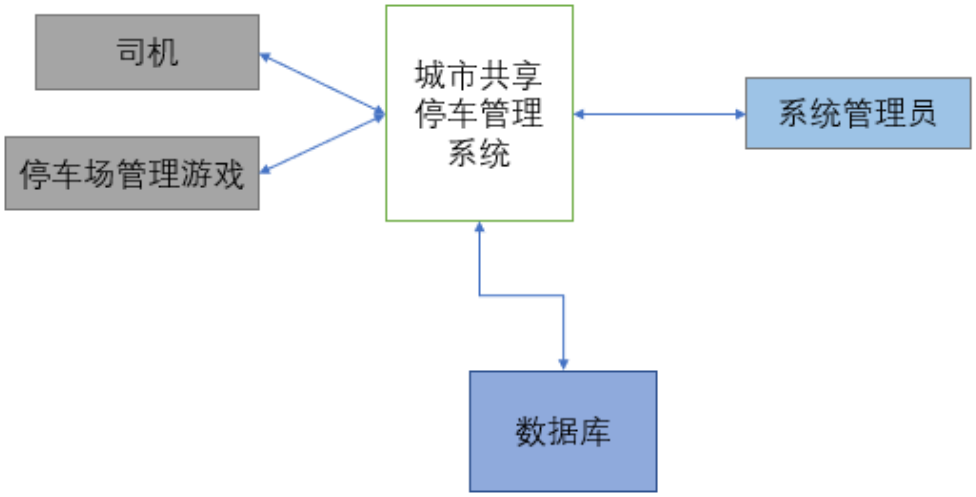
3.1.1.模块级体系结构


城市共享停车管理系统
司机管理
司机个人信息管理
登录个人界面
查看个人信息
修改个人信息
目的地查询
查看目的地
目的地附近停车场预约
目的地导航
计费错误申请
提出申请并提交凭证
停车场管理员管理
停车场管理员个人信息管理
登录个人界面
查看个人信息
修改个人信息
修改停车场信息
成绩输入
计费规则管理
查看当前计费规则
输入新的规则
计费错误管理
发现错误获得修改权限
修正费用
回退/收取费用
系统管理员
用户管理
查看、修改各个用户的个人信息及权限

3.1.2. 分层级体系结构



3.1.3. 概念级体系结构



3.2.体系结构概述

3.2.1.B/S 三层体系结构简介

B/S 结构（Browser/Server，浏览器/服务器模式），是 WEB 兴起后的一种网络结构模式，WEB 浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。客户机上只要安装一个浏览器（Browser 英 [’braʊzə]美 [’braʊzɚ]），如 Netscape Navigator 或 Internet Explorer，服务器安装 SQL Server、Oracle、MYSQL 等数据库。浏览器通过 Web Server 同数据库进行数据交互。

3.2.2.体系结构设计

1) 城市共享停车管理基于 J2ee 的 B/S 结构，通过 WEB 的方式提供人机交互的界面，便于系统远程维护及升级，便于用户随时随地通过网络登录系统平台。
2) 采用 VPN 网络系统支撑平台运行，平台为应用系统提供包括：用户访问控制、信息加密、身份认证等安全方面的服务，全面保证系统安全。
3) 系统具有高可靠性，保证联网用户的在线率及成绩或评分规则准确无误的上传。
4) 系统具有高稳定性，保证服务器在处理大量信息时不死机，尤其是在期末等访问量大的情况下。
5) 系统并发可支持该校的学生访问，满足各用户的要求。

6) 系统保证可扩展性。系统可继续开发为为企业提供运动管理、成员信息管理，要求整个系统能在不间断使用的情况下完成系统的升级。我们的产品在设计中，主服务器及网络设备采用模块化结构，硬件平台可以积木式拼装。平台产品提供良好的业务类型扩展性和业务规模扩展性，保证系统能快速方便地引入新的硬件和软件系统，可以随服务内容和业务量的增加动态部署计算机以提高系统处理能力。

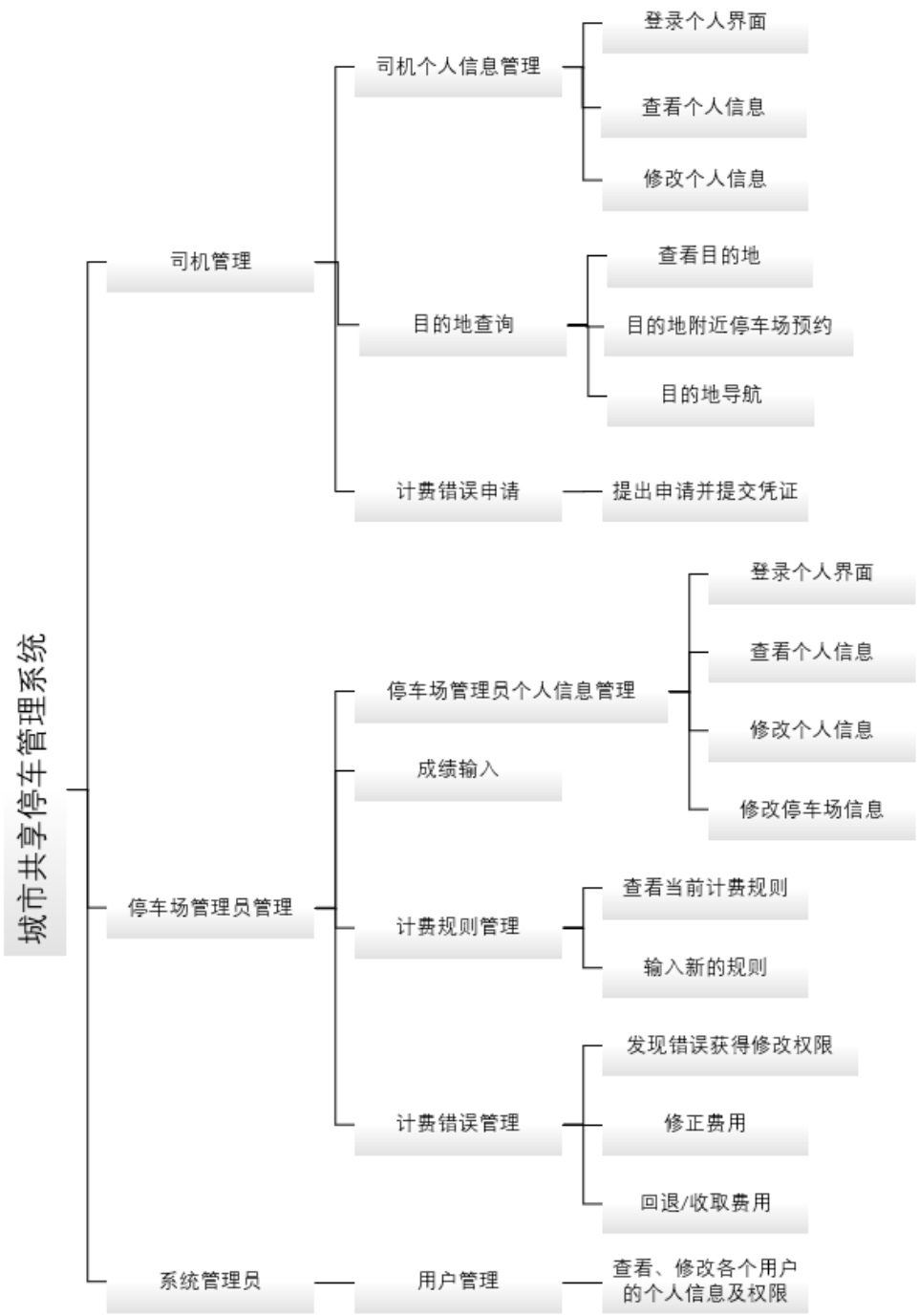
7) 系统支持开放性与标准化原则。采用开放技术标准，便于与基于不同开发技术实现的各种内外部系统互联互通，另一方面，在产品供应商和技术服务商的选择上也提供了更大的余地。

8) 系统实时运行过程中对数据进行备份，保证数据的安全性和有效性，同时实现系统运行时联网数据导入导出，不影响系统的实时运行。

9) 系统支持用户实时接入。联网用户的接入不影响系统的实时运行。

3.3.结构化视图

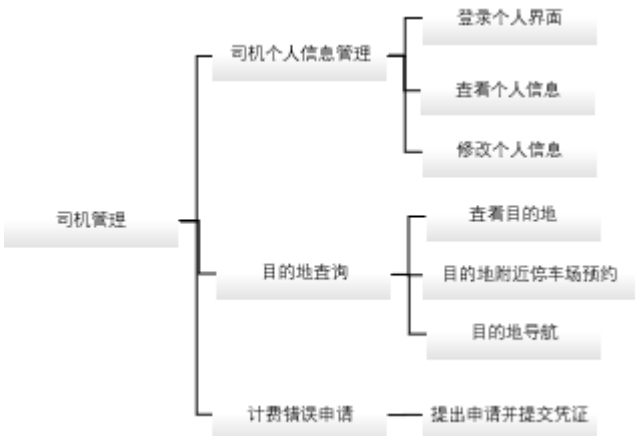
3.3.1.功能结构图



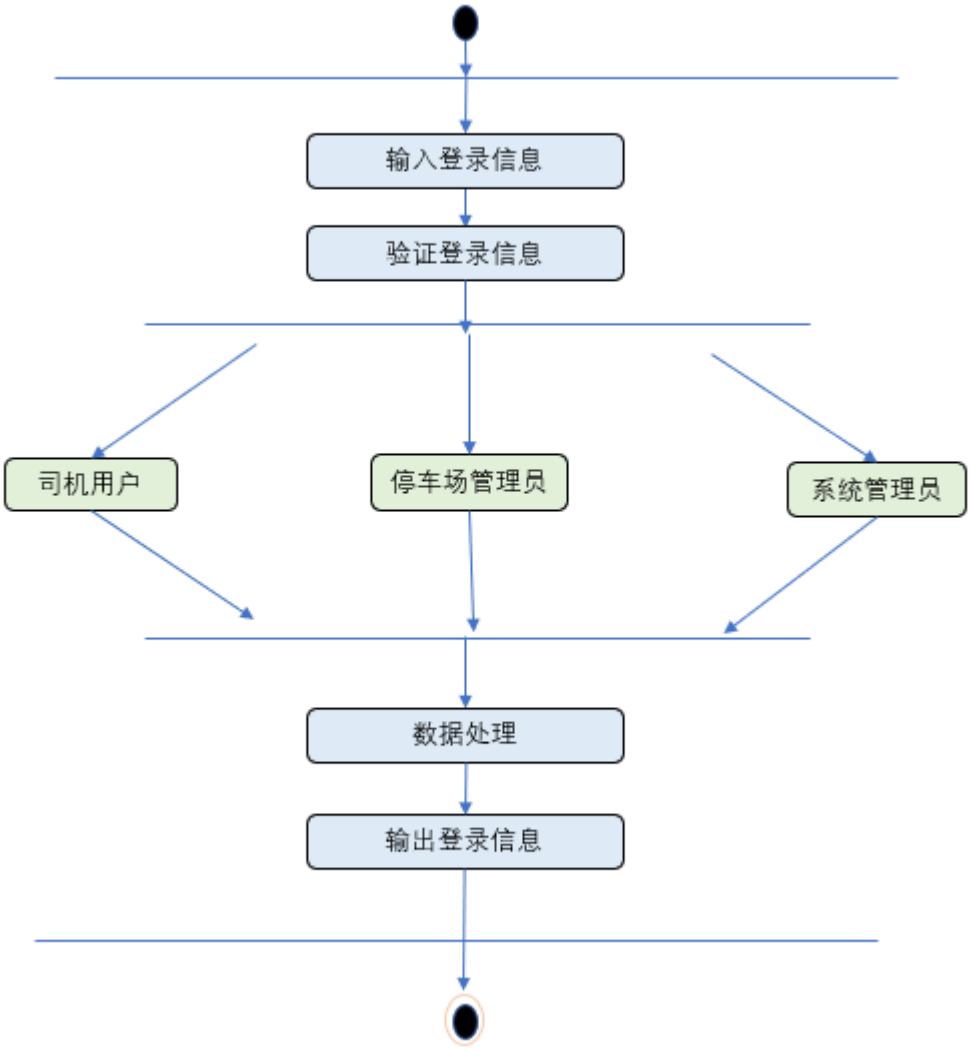
3.3.2. 系统管理员功能结构图

3.3.3. 停车场管理员功能结构图

3.3.4. 司机功能结构图



3.4. 行为视图



第四章 系统的质量分析和评价

4.1. 场景分析

4.1.1. 用例场景

1) 当发生数据异常时，系统要通知到所有在线用户，并在屏幕上用红色的字体显示出来。该场景代表了用户期望的可靠性。
2) 学生期望可以看到自己的总体育成绩以及各部分复杂成绩。该场景代表了用户期望系统易于使用，即易用性。
3) 当用户登录系统时需要进行身份验证，而不是任何人都可以登录。该场景代表了用户所期望的安全性。
4) 系统管理员期望授予一类用户以权限，例如授予所有的学生以查询成绩的权限等。该场景代表了用户期望的易用性。
5) 当用户将图形界面放大时，屏幕要在 1 秒内重新显示出来。该场景代表了系统性能要求。

4.1.2. 增长性场景

预期未来系统修改时可能发生的场景。

1) 通过扩充现有数据库表的规模，把检索时间降低到平均 1s 以内。
2) 当系统不再局限于 web，而能用于手机 app 时，该系统能照样运行良好，检索速度快等

4.1.3.探索性场景

推动系统封装和降低工作压力的场景。

1) 系统能够从 Windows 平台更换到 Linux 平台。

2) 改进系统的可使用性，使其从 98%提升到 99.9999%。

3) 正常情况下，当一半服务器宕机时，不影响整个系统的可使用性。

4.2.原型分析

4.2.1.学生界面

4.2.2. 体育教师界面

4.2.3. 体育教务长界面

4.3.风险

在项目进行过程中可能发生的事件，这些事件将会对项目按预期时间，资源和预算完成产生重大影响。

1) 规模风险

需求是否相当稳定并得到了充分的了解。

项目规模是固定不变还是在不断扩展。

项目开发的时间范围是否太短、不够灵活。

2) 技术风险

异常处理是否得当。

构件可能要在若干次发布后才能变得稳定，以致无需重大变更即可复用。

需求中的事务量是否合理。

数据量是否合理？当前可用的框架是否能够保存这些数据。

对于与其他系统（包括企业以外的系统）的接口是否存在外部依赖性？是否存在必需的接口或必须创建它们 。

是否存在极不灵活的可用性和安全性需求（例如“系统必须永远不出现故障”）。

系统的用户是否对正在开发的系统类型没有经验。

应用程序的大小或复杂性，或者技术的新颖性是否导致了风险的增加。

是否存在对国家语言支持的需求。

第五章 更改管理过程

几乎可以肯定，体系结构在开发过程中一定会发生变化，也许是出自客户的遗漏，也可能是在开发过程中被激发出来的，此时做好变更日志管理就显得非常重要。

关于变更管理可以参照下图示意：