《驾校系统》开发文档

题 目：基于B/S驾校系统分析与设计

班 级：

开发者：许昌

时 间：2018.11.29

目录

[1 可行性分析报告 5](#_Toc532503581)

[1.1 引言 5](#_Toc532503582)

[1.2 问题定义 5](#_Toc532503583)

[1.3 背景 5](#_Toc532503584)

[1.4 可行性研究的前提 6](#_Toc532503585)

[1.4.1 要求 6](#_Toc532503586)

[1.4.2 目标 6](#_Toc532503587)

[1.4.3 条件、假定和限制 6](#_Toc532503588)

[1.4.4 进行可行性研究的方法 6](#_Toc532503589)

[1.4.5 评价尺度 8](#_Toc532503593)

[1.5 对系统的分析 9](#_Toc532503594)

[1.5.1 对现有系统分析 9](#_Toc532503595)

[1.5.2 系统大致处理流程 9](#_Toc532503596)

[1.5.3 系统流程图 9](#_Toc532503598)

[1.5.4 工作负荷 10](#_Toc532503600)

[1.5.5 费用开支 10](#_Toc532503601)

[1.5.6 人员 11](#_Toc532503602)

[1.5.7 设备 11](#_Toc532503603)

[1.5.8 局限性 11](#_Toc532503604)

[1.6 局限性 11](#_Toc532503605)

[1.6.1 对所建议系统的说明 11](#_Toc532503606)

[1.6.2 改进之处 11](#_Toc532503607)

[1.7 影响 11](#_Toc532503608)

[1.7.1 对设备的影响 11](#_Toc532503609)

[1.7.2 对软件的影响 12](#_Toc532503610)

[1.7.3 对用户单位机构的影响 12](#_Toc532503611)

[1.7.4 对系统运行过程的影响 12](#_Toc532503612)

[1.7.5 对开发的影响 12](#_Toc532503613)

[1.7.6 对地点和设施的影响 12](#_Toc532503614)

[1.7.7 对经费开支的影响 12](#_Toc532503615)

[1.8 局限性 12](#_Toc532503616)

[1.9 技术条件方面的可行性 13](#_Toc532503617)

[1.10 投资及效益分析 13](#_Toc532503618)

[1.10.1 支出 13](#_Toc532503619)

[1.10.1.1 基本建设投资 13](#_Toc532503620)

[1.10.1.2 其他一次性支出 13](#_Toc532503621)

[1.10.1.3 非一次性支出 13](#_Toc532503622)

[1.10.2 收益 14](#_Toc532503623)

[1.10.2.1 一次性收益 14](#_Toc532503624)

[1.10.2.2 非一次性收益 14](#_Toc532503625)

[1.10.2.3 不可定量的收益 14](#_Toc532503626)

[1.10.2.4 收益／投资比 14](#_Toc532503627)

[1.10.2.5 投资回收周期 14](#_Toc532503628)

[1.10.2.6 敏感性分析 14](#_Toc532503629)

[1.11 社会因素方面的可行性 15](#_Toc532503630)

[1.11.1 法律方面的可行性 15](#_Toc532503631)

[1.11.2 使用方面的可行性 15](#_Toc532503632)

[2 需求分析说明书 15](#_Toc532503633)

[2.1 引言 15](#_Toc532503634)

[2.2 定义 15](#_Toc532503635)

[2.3 用户的特点 16](#_Toc532503636)

[2.4 需求规定 16](#_Toc532503637)

[2.4.1 总体功能模块 16](#_Toc532503638)

[2.4.2 模块 17](#_Toc532503640)

[学员模块: 17](#_Toc532503642)

[教练模块: 18](#_Toc532503643)

[管理员模块: 18](#_Toc532503644)

[2.5 对性能的规定 18](#_Toc532503645)

[2.5.1 精度 18](#_Toc532503646)

[2.5.2 时间特性要求 18](#_Toc532503647)

[2.5.3 灵活性 19](#_Toc532503648)

[2.6 输入输出要求 19](#_Toc532503649)

[2.7 运行环境规定 19](#_Toc532503650)

[2.8 数据管理能力要求 19](#_Toc532503651)

[2.9 故障处理要求 19](#_Toc532503652)

[2.10 其他专门要求 19](#_Toc532503653)

[2.11 接口 20](#_Toc532503654)

[2.12 控制 20](#_Toc532503655)

[3.2.2 精化类图 23](#_Toc532503656)

[3.3 协作图 24](#_Toc532503657)

[3.6 **系统功能结构图** 27](#_Toc532503659)

[3.7 数据库设计 28](#_Toc532503661)

[3.8 概念结构设计 29](#_Toc532503664)

[3.9 逻辑结构设计 30](#_Toc532503666)

[3.10 数据表结构设计 31](#_Toc532503667)

[4.1.1 命名风格 34](#_Toc532503668)

[4.1.2 常量定义 37](#_Toc532503669)

[4.1.3 代码格式 38](#_Toc532503670)

[4.1.4 注释规约 41](#_Toc532503671)

[6 参考文献 44](#_Toc532503672)

1. 可行性分析报告
   1. 引言

**编写驾校刷题系统可行性分析报告目的是为了了解系统设计与实现在技术、资金以及法律方面的问题的可行性，去分析驾校系统是否有实现的可能。通过小组不断的讨论和交互，最终形成具有建设目标的书面条款。经确认后，将作为开发方设计开发的基本依据和需求方的软件验收标准，同时，通过报告，进一步了解系统的实现应该按照工程学的技术去设计实现，从而严格按照流程及时、准确地完成系统的开发，以满足系统需求。同时，该文档也作为概要设计及后续设计的基础。**

* 1. 问题定义

**想要实现驾校系统，应该需要相关的法律法规，需要专业的驾校考试的管理人员对于驾校视频的解读与试题的设计；团队人员的划分以及人员的不同技能等等。**

* 1. 背景

**随着社会经济发展与人民收入水平的提高，以及汽车价格日益亲民化，人们对汽车和驾驶的需求逐步提升，据公安部交管局统计，截止2015年底，全国机动车驾驶人数量超3.2亿，汽车驾驶人2.8亿人，全年新增汽车驾驶人3375万人；全国机动车保有量达2.79亿辆，其中汽车1.72亿辆，私家车达到1.24亿辆，汽车新注册辆和年增辆均达到最高水平。汽车驾驶员数量及汽车保有量的逐步提升，带来汽车生活的场景日益丰富化，催生出巨大的汽车生活服务需求，比如驾考需求、购车需求、汽车保养需求等等。而驾考需求作为最基本的也是占比最大的准入性需求，其市场前景巨大。如今世界上基于物联网的技术不断发展，而驾校系统的出现真正解决考驾照需求的18到70岁的初学者，能够在APP或者网站上了解驾照考试的流程和内容。**

* 1. 可行性研究的前提
     1. 要求
     + **功能：驾照考试需求，驾校**
     + **性能：经过软件测试，软件性能完好，满足软件开发协议**
     + **注册用户或者以游客方式登录，进行刷题**
     + **系统上线大概需要8周时间**
     + **内测时间大约一个星期**
     1. 目标
* **驾校系统是基于B/S模式以及手机移动端**
* **采用百度加速器优化系统，使系统在响应方面能快速；**
* **用户界面友好，用户体验良好；**
* **系统的响应时间不超过5秒，着重用户隐私不被泄露**
  + 1. 条件、假定和限制
* **系统测试版本上线时间为一个月**
* **系统运行测试时间为一个星期，维护周期为一个星期；**
  + **网站、系统需要线上备案，要求符合互联网安全协议；**
  + **经费、投资方面由华软快递中心老板邓培育提供；**
  + **申请相关法律和政策方面的条文，按照国家法律法规；**
  + **硬件、软件、运行环境和开发环境方面的条件和限制；**
    1. 进行可行性研究的方法

问卷调查（调查用户或潜在用户的使用频率、心得等）:

**驾照考试APP月活跃用户总体保持在3000万至4000万之间，渗透率基本保持在3%至4%之间，使用驾考相关APP用户日人均使用时长平均接近3小时。用户晚上8-10点为最活跃时段，单一用户日均停留2小时50分，其中，工作日和周末的活跃情况差异明显，在9时、21时达到活跃峰值，凌晨0点以后活跃度略低。**

心得：

**在汽车保有量和驾驶人员数量不断攀升的行业背景下，驾照考试行业仍然有较大的市场空间及持续性。而因新驾考规则的出台，使得这一数据又有了进一步的提高，也由此可看出，我国的驾考市场的巨大潜力。**

**对收集的资料和数据进行分析**

近五年私家车保有量及趋势：

图1.4.4.1

由驾考APP行业用户使用粘性看，除受到春节假期影响外，全年数据较为稳定；使用驾考APP用户月人均使用时长平均接近3小时

图1.4.4.2

图1.4.4.3

* + 1. 评价尺度

**驾照刷题系统合适的用户群体比较广泛，也能够目标群体的需求。按照国家相关法律法规及相关的互联网安全协议，用户界面友好，用户体验较好。**

* 1. 对系统的分析
     1. 对现有系统分析

**现有的驾校刷题系统（驾校一点通）能够做到系统性的刷题、究题、重做等功能，虽说我们现有系统的功能模块，从实际出发，能够有系统性对用户题型进行检查，分析。同时我们设计视频教学，资讯驾校新闻浏览，教练对学员的动态关注。**

* + 1. 系统大致处理流程



* + 1. 系统流程图



* + 1. 工作负荷

系统对于同时在线人数的负担在测试中，视频浏览模块的性能以及相应时间在可控的范围内。服务器环境等等；

* + 1. 费用开支
* **服务器：腾讯云主机（9.9每个月）数据库mysql（18每个月）**
* **操作系统：window10（5000）**
* **安卓手机：调试（2000）**
* **开发工具：eclipse、Intelij IDE、Tomcat8.5**
  + 1. 人员

**前端工程师、java工程师、运维师、系统测试、项目经理**

* + 1. 设备

|  |  |
| --- | --- |
| 操作系统： windows 10 | 最低配置： CPU 为 P4 2.8Ghz，内存为 8G； |
| IE6.0 以及以上版本浏览器 | 能支持 IE6.0 以及以上版本的操作系 统所需要的最低硬件配置 |
| 数据库： mysql 5.7 | 最低配置： CPU 为 P4 2.8Ghz，内存为 8G； |
| redis | 8核8G企业版 |
| 邮件服务器 | 8核8G企业版 |

* + 1. 局限性

**驾校系统避免不了需要存储试题库，而试题库中往往有上千条甚至上万条记录，拥有的表也会相对复杂，所以在对数据库进行操作的时候，需要数据库响应时间在人们能够接受的范围之内。系统服务器性能低，可允许客户访问数量有限制。**

* 1. 局限性
     1. 对所建议系统的说明

**针对不同的用户群体，不同的用户数量，我们现今无法对我们目前的驾校系统发现硬件性能上的需求。**

* + 1. 改进之处
  1. 影响
     1. 对设备的影响

**系统更新和维护，对设备的不兼容等**

* + 1. 对软件的影响

**系统及软件版本不符**

* + 1. 对用户单位机构的影响

**对驾校教练场造成数据丢失**

**对用户造成体验不好，影响心情**

* + 1. 对系统运行过程的影响
* **用户可能无法登录，系统功能崩溃；**
* **用户体验不好；**
* **源数据可能丢失；**
* **数据进入系统的过程缓慢；**
* **数据存储、恢复的处理程度及处理时间；**
* **系统失效的后果及恢复的处理。**
  + 1. 对开发的影响
* 用户需更新移动端；
* 数据库系统性能要求较高，数据量大；
* 用户数据泄露和系统被攻击等
  + 1. 对地点和设施的影响

**实验室环境浪费**

* + 1. 对经费开支的影响

**资金品谷不足，造成资源浪费；**

* 1. 局限性
* 现有技术水平的不定性；
* 软件测试性能不成熟；
* 用户数量及群体等。
  1. 技术条件方面的可行性
* **系统模块功能能够做出来；**
* **利用现有的技术，该系统的功能能够实现；**
* **开发人员的数量和质量的要求并说明这些要求能够满足；**
* **在规定的期限内，本系统的开发能完成。**
  1. 投资及效益分析
     1. 支出
        1. 基本建设投资
* **房屋和设施；**
* **开发环境**
* **数据通讯设备；**
* **云主机；**
* **安全与保密设备；**
* **数据库管理软件。**
  + - 1. 其他一次性支出
* **研究（需求的研究和设计的研究）；**
* **开发计划与测量基准的研究；**
* **数据库的建立；**
* **服务器环境搭建**
* **检查费用和技术管理性费用；**
* **培训费、旅差费以及开发安装人员所需要的一次性支出；**
  + - 1. 非一次性支出
* **设备的租金和维护费用；**
* **软件的租金和维护费用；**
* **数据通讯方面的租金和维护费用；**
* **人员的工资、奖金；**
* **房屋、空间的使用开支；**
* **保密安全方面的开支；**
* **其他经常性的支出等。**
  + 1. 收益
       1. 一次性收益
* **开支的缩减包括改进了的系统的运行所引起的开支缩减，如资源要求的减少，运行效率的改进，数据进入、存贮和恢复技术的改进，系统性能的可监控，软件的转换和优化，数据压缩技术的采用，处理的集中化／分布化等；**
* **价值的增升包括由于一个应用系统的使用价值的增升所引起的收益，如资源利用的改进，管理和运行效率的改进以及出错率的减少等；**
* **其他如从多余设备出售回收的收入等。**
  + - 1. 非一次性收益

**说明在整个系统生命期内由于运行所建议系统而导致的按月的、按年的能用人民币数目表示的收益，包括开支的减少和避免。**

* + - 1. 不可定量的收益

**逐项列出无法直接用人民币表示的收益，如服务的改进，由操作失误引起的风险的减少，信息掌握情况的改进，组织机构给外界形象的改善等。有些不可捉摸的收益只能大概估计或进行极值估计（按最好和最差情况估计）。**

* + - 1. 收益／投资比

**求出整个系统生命期的收益／投资比值。**

* + - 1. 投资回收周期

**求出收益的累计数开始超过支出的累计数的时间。**

* + - 1. 敏感性分析

**所谓敏感性分析是指一些关键性因素如系统生命期长度、系统的工作负荷量、工作负荷的类型与这些不同类型之间的合理搭配、处理速度要求、设备和软件的配置等变化时，对开支和收益的影响最灵敏的范围的估计。在敏感性分析的基础上做出的选择当然会比单一选择的结果要好一些。**

* 1. 社会因素方面的可行性
     1. 法律方面的可行性

**法律方面的可行性问题很多，如合同责任、侵犯专利权、侵犯版权等方面的陷井，软件人员通常是不熟悉的，有可能陷入，务必要注意研究。因此双方签订合法的条文等。**

* + 1. 使用方面的可行性

**按照用户手册使用**

1. 需求分析说明书
   1. 引言

**编写驾校系统需求报告分析目的是为了需求提供者和开发方明确对所建信息管理系统所达到的功能和目标。通过双方不断的讨论和交互，最终形成具有建设目标的书面条款。经双方确认后，将作为开发方设计开发的基本依据和需求方的软件验收标准，同时，通过该需求分析报告，开发方可以更加进一步了解客户的需求，从而严格按照流程及时、准确地完成系统的开发，以满足客户的需求。同时，该文档也作为概要设计及后续设计的基础。**

**本说明书的预期读者为：开发者 指导老师。**

* 1. 定义
  + **IntelliJ IDEA：是java语言开发的集成环境，IntelliJ在业界被公认为最好的java开发工具之一，尤其在智能代码助手、代码自动提示、重构、J2EE支持、Ant、JUnit、CVS整合、代码审查、 创新的GUI设计等方面的功能可以说是超常的。**
  + **Visual Studio：VS是一个基本完整的开发工具集，它包括了整个软件生命周期中所需要的大部分工具，如UML工具、代码管控工具、集成开发环境(IDE)等等。**
  + **MySQL：MySQL是最流行的关系型数据库管理系统，在WEB应用方面MySQL是最好的RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。**
  1. 用户的特点

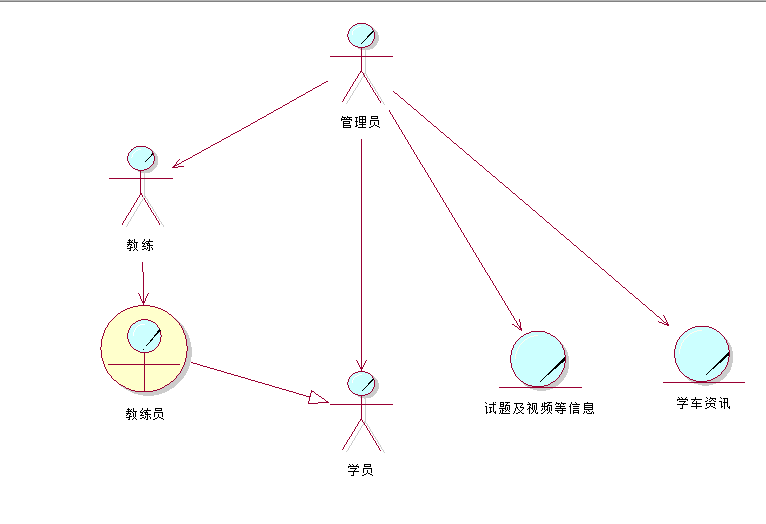
**本软件的用户有三种：**

**第一种是学员，即该系统的普通用户。他们将在驾车系统进行驾校考试学习，对于不同驾校的了解以及对于不同教练的认识，提供更多的渠道给他们选择驾校以及教练。他们不需要过多的计算机知识，只要能够使用系统就行。**

**第二种是教练，教练能够对管理学员，了解学员的动态。**

**第三种是系统管理员，需要有较高的专业知识，能够对数据库进行管理和维护。**

* 1. 需求规定
     1. 总体功能模块



**移动端：用户可以在移动端登陆驾校刷题系统**

**web端：用户可以在web端登陆驾校刷题系统**

**定期得维修：**

**为了防止漏洞的形成，系统管理员需要定期维修系统。**

**系统管理员需要定期更新和维护驾校刷题系统**

**统计：**

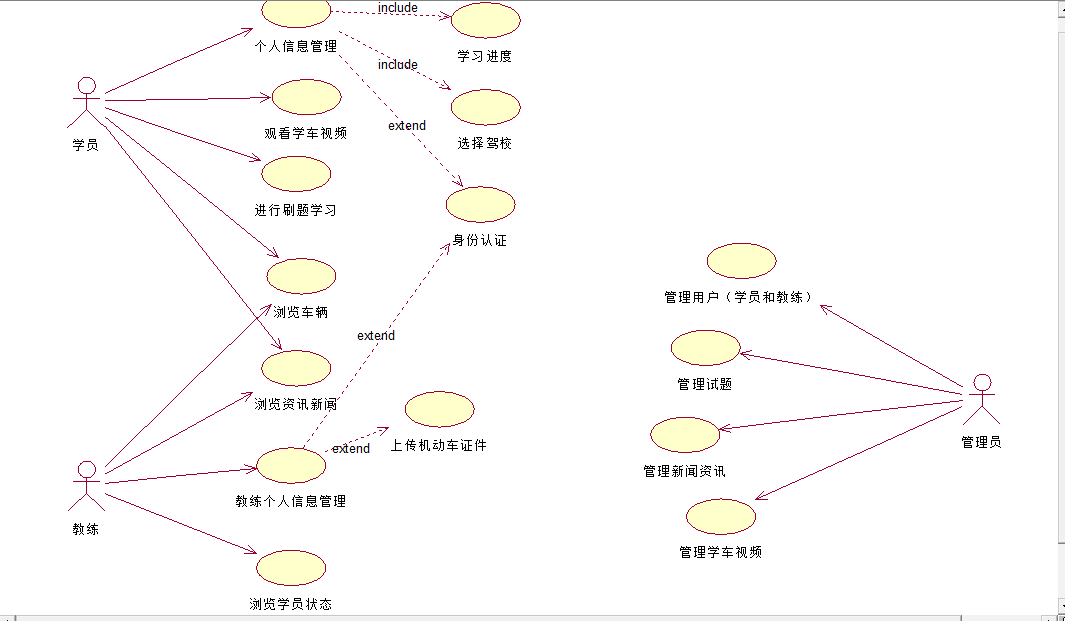
**计算教练的信息，方便用户选择教练**

**计算驾校的信息，方便用户选择驾校**

**计算车辆的信息，并放入购物车中，方便用户购买车辆。**

**计算刷题的信息**

* + 1. 模块



学员模块:

**游客：当用户暂时不想注册时，可以选择游客方式来浏览系统，游客方式只能刷题及查看个人信息。**

**注册：当用户想解锁更多内容时，必须先注册，因此系统提供用户注册功能，以及相关的用户信息修改、密码维护等功能。**

**登录：当用户报名驾校和选择教练以及购车时，必须登录成功之后才可以进行。**

**学车视频：用户可以看到很多优秀的学车视频教程。**

**刷题程序：用户可以在刷题程序中进行刷题以及比对答案。**

**个人管理：用户管理自己的个人信息、刷题的进度、购车记录、观看视频记录、所选择的驾校和教练等。**

**购车需求：用户可以挑选自己喜欢的车，并购买。**

**驾校报名：用户根据驾校的信息选择自己想要的驾校进行报名**

**选择教练：用户根据教练的信息选择自己想要的教练**

教练模块:

**查看学生状态：教练可以查看学生的状态**

**管理学生：教练对学生进行管理，如布置学生任务等**

**发布状态：教练发布自己当前状态，是空闲还是繁忙**

**个人管理：教练可以管理自己的个人信息**

管理员模块:

**管理试题：上传试题，然后分类管理，更新到网站前台，还可以进行试题增加，修改，删除等操作。**

**购车需求：上传车辆的信息，可以对车辆信息进行增加，修改，删除等操作，当接收到用户订单信息后，根据订单要求给用户发货，打印发货单。**

**发布学车视频：上传学车视频，进行学车视频教程的增加，修改，删除等操作。**

**管理用户1：用户基本信息的管理，如用户的个人信息、操作权限、教练信息等。**

**管理教练：教练基本信息的管理，如教练的个人信息、操作权限、学生信息等。**

**发布资讯：发布资讯，对资讯进行增加，修改，删除等操作。**

* 1. 对性能的规定
     1. 精度

**该系统的主要操作数据就是驾车的人员对于视频和刷题量的精确。**

**用户进行视频等多功能的时应保证注册成功，所有在相应域中包含查询关键字的记录都应该能查到。**

* + 1. 时间特性要求

**要想提供令用户满意的服务，在完善强大功能的背后，系统的反应时间也是关键的因素，在这里我们要求对于搜索的响应时间不能超过8秒，对于在线设计的响应时间不能超过20秒。**

**用户对商品进行评论时，信息更新处理时间不能超过10秒。**

**其它所有交互功能反应速度：不超过3秒；**

* + 1. 灵活性

**该系统的数据库要求可以在需要的时候添加新的关系，或者属性，而不会降低系统的性能。当数据库发生变化时，我们对应的系统版本需要升级，但是现有的界面与接口不会发生太大的变化，只是添加相应的功能按钮。**

* 1. 输入输出要求

**为了让用户能流畅使用，系统必须要有高的运作速度，用户进行答题，浏览时，系统必须能快速及时响应操作，迅速处理各项数据、信息，显示出用户所需要的信息，所以要求系统有很高的信息量速度和大的主存容量；由于要存贮大量的数据和信息，也要有足够大的磁盘容量；另外，系统必须要有可靠的安全措施，对数据输入均进行数据有效性检查，以保证用户信息的存储安全。**

* 1. 运行环境规定
  2. 数据管理能力要求

**运行本软件系统所需的各种基础数据（如上传的模型）以及前期的其他数据的规模约为1000M，数据的平均增长约为50M/月，系统用于日志等记录的数据增长约为20M/月。具体增长速度由用户和供应商的使用频率及所发生业务的数据量决定。**

* 1. 故障处理要求

**该系统的核心部分在数据库和文件系统，要求数据库具有即时备份功能，而且文件系统的数据也要及时备份。正常使用时不应出错，对于用户的输入错误应给出适当的改正提示。当数据量的更新达到一定的限度，或者间隔时长达到一定要求，数据库可以自主选择备份，以尽可能降低在出现断电、误操作等行为造成的损失。**

**软件在运行过程中产生的数据错误，将由系统自动记入错误日志，非网络传输引起的错误将由系统管理员或软件开发者解决。**

**软件在运行过程中产生的其他错误，将根据情况由软件开发者或软件开发者协助系统管理员解决。**

* 1. 其他专门要求

**该系统的安全要求级别很高，对于数据库的访问及操作需要足够的权限。用户不具备对其基本信息以外的数据进行修改更新等操作的权限，管理员可以对数据库进行增添、修改。对于过期或需要注销的用户，在满足预定义条件后，由数据库系统进行自动删除。对于用户花费情况的统计，只能由数据库自动进行修改，或者数据库管理人员具有修改权限（需要主管部门的同意）。**

**同时还要防黑客的攻击，当数据库系统被破解或者登录时，它可以完成自动备份；当检查到有“恶意”操作的时候，我们在服务器上的数据应该具有自动drop功能，以减少信息的泄漏和降低经济损失。系统管理员可以把备份的数据重新放至到服务器上。**

* 1. 接口

**本系统移动终端采用图形用户接口，方便用户使用，此外采用树视图技术，满足用户对家居建材模型进分类查看的需求。**

**本系统基于B/S模式开发，从而使服务器数据源与客户端分离，保证数据库系统的物理独立性。**

* 1. 控制

**本系统的系统架构，以及权限机制可以保证系统的安全性。服务器的系统是可以自动运行的，同时数据库管理人员有权限进行管理。在远程服务器端，数据库管理系统自动对相关预设定信息进行更新，例如用户花费信息、家居建材模型等的更新，是不需要进行任何手动修改的，除非在获得相关部门的同意后，数据库管理人员可以拥有权限进行相关修改。**

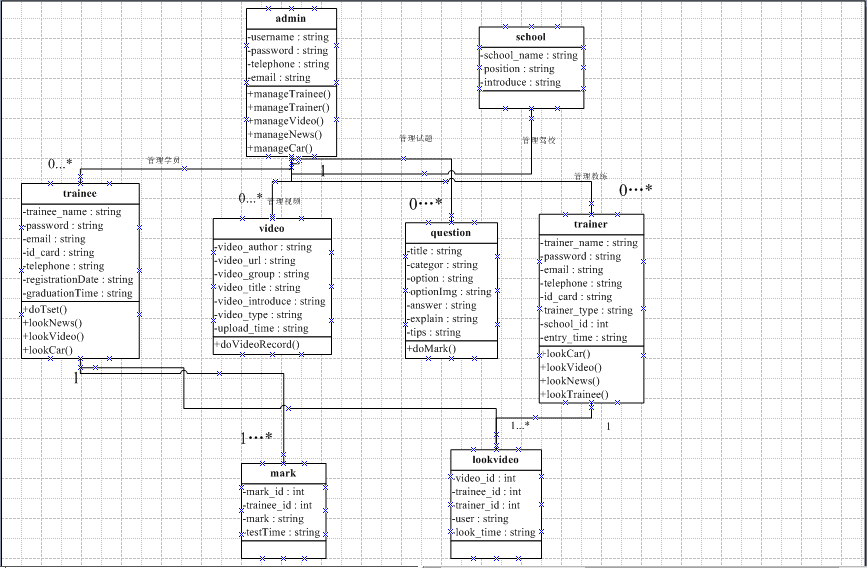
1. 系统设计
   1. 用例规约

|  |  |
| --- | --- |
| **用例名称** | **个人信息管理模块** |
| **用例描述** | **学员、教练注册登录进入个人信息界面，进行身份证实名认证，修改完善个人信息，查看学习计划进度等**  **教练注册登录进入个人信息界面，进行身份证实名认证、教练上传机动车车证，修改完善个人信息，浏览学员状态等** |
| **用例图** |  |
| **主要流程** | **学员、教练登录进入到个人主页，身份证认证**  **机动车认证**  **修改，完善个人信息**  **退出** |
| **替代流程** | **身份证照片，机动车车证上传，大小错误等提示错误信息** |

|  |  |
| --- | --- |
| **用例名称** | **刷题学习模块** |
| **用例描述** | **学员、游客进行科目一或科目四的模拟考试** |
| **用例图** |  |
| **主要流程** | **学员、游客进入试题界面，选择考试类型**  **系统随机分配考试试题**  **考试时间结束或者提交试卷，生成成绩等**  **退出或再次进行答题** |
| **替代流程** | **判断是否登录（学员或游客），游客等待考试结束直接生成成绩及错题信息，学员生成成绩单及提供学习计划等** |

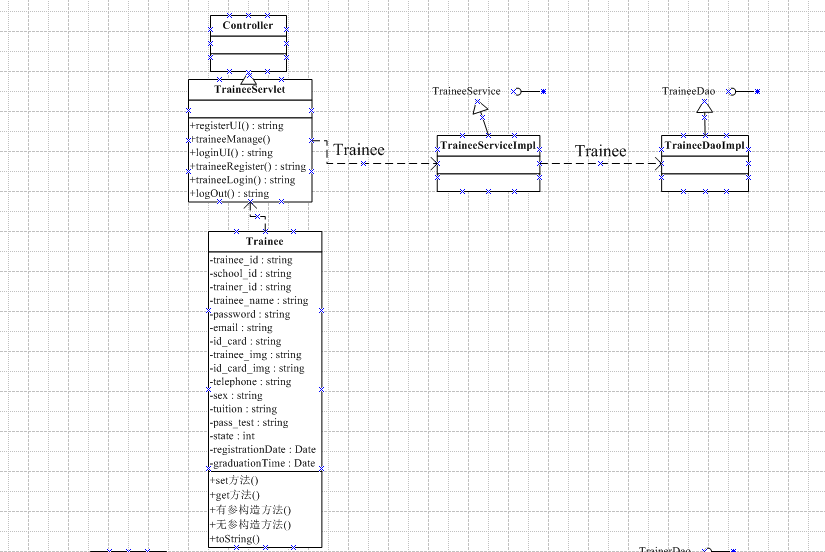
|  |  |
| --- | --- |
| **用例名称** | **观看视频模块** |
| **用例描述** | **学员、游客进行科目二或科目三的视频学习** |
| **用例图** |  |
| **主要流程** | **学员、游客进入视频界面，选择观看视频类型**  **学员视频观看完毕，生成视频记录等**  **退出或再次进行观看** |
| **替代流程** | **判断是否登录（学员或游客），学员生成视频记录及提供学习计划等** |

* 1. 类图
     1. 初步类图

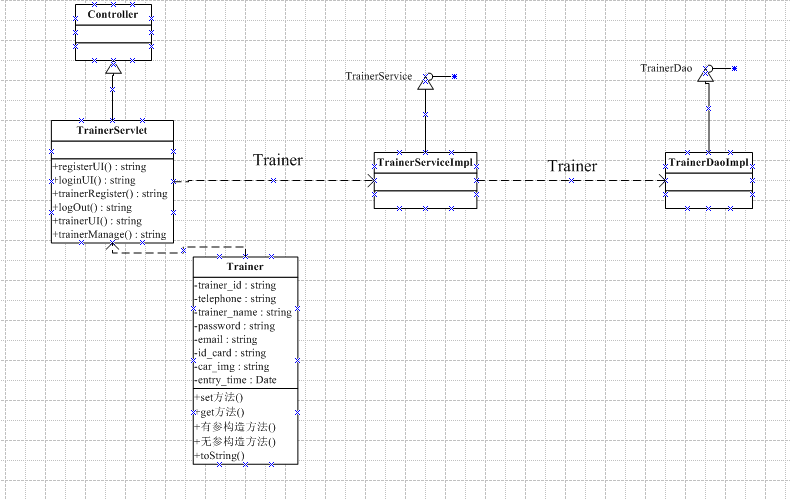


* + 1. 精化类图

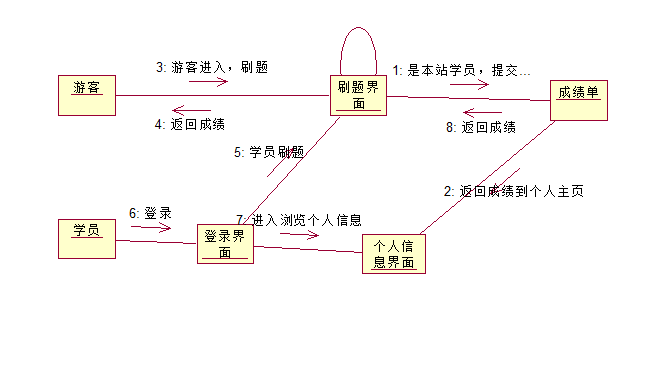
学员



教练

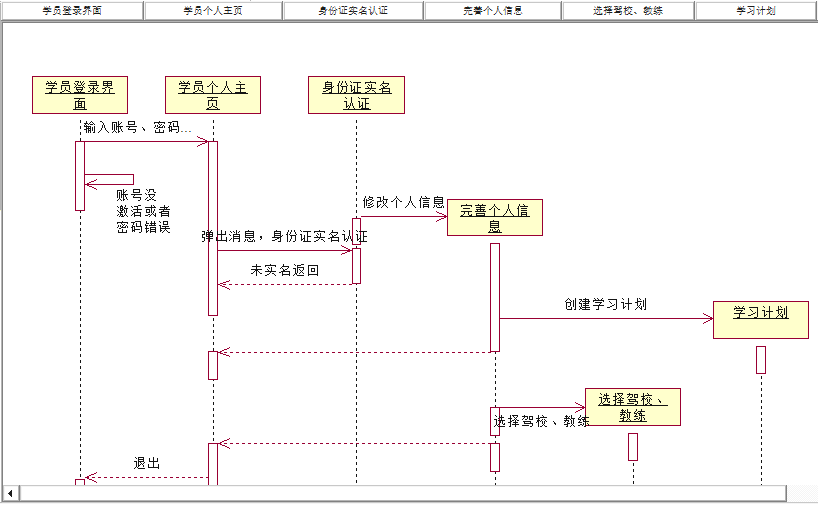


* 1. 协作图

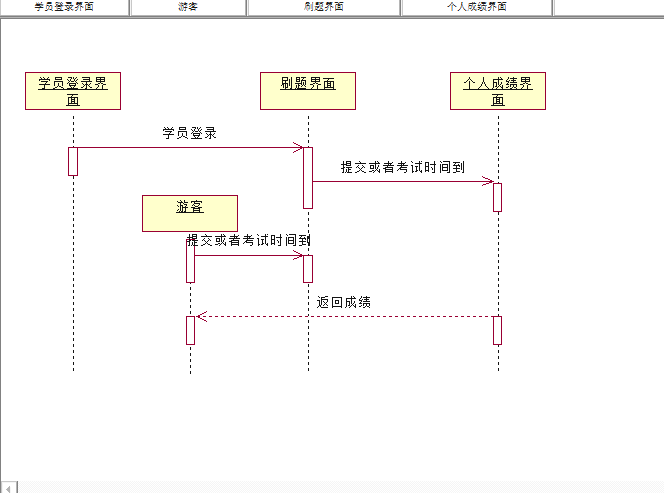


* 1. 顺序图

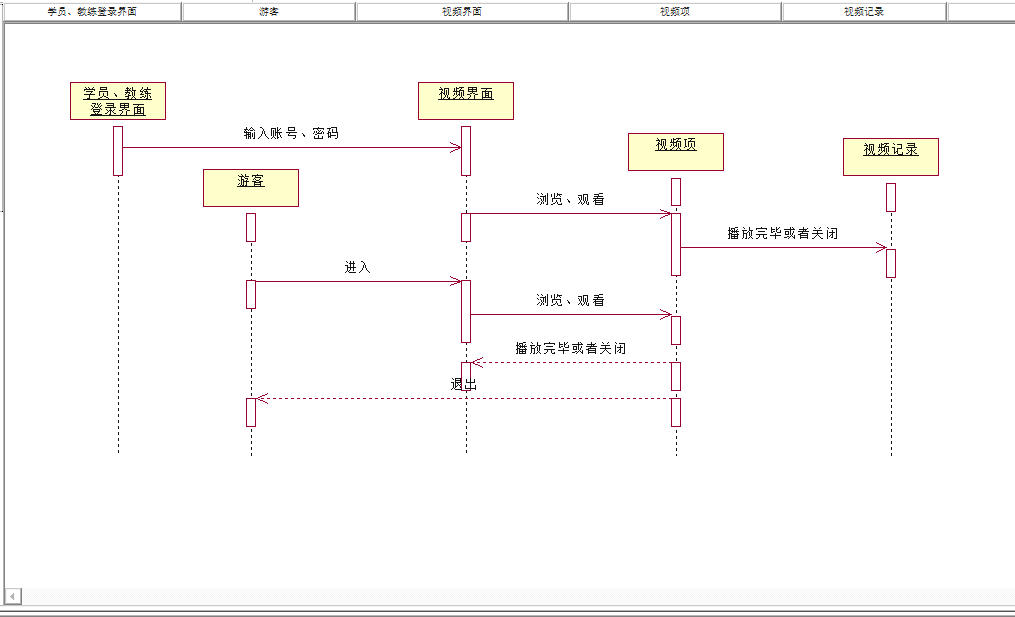
学员个人信息管理



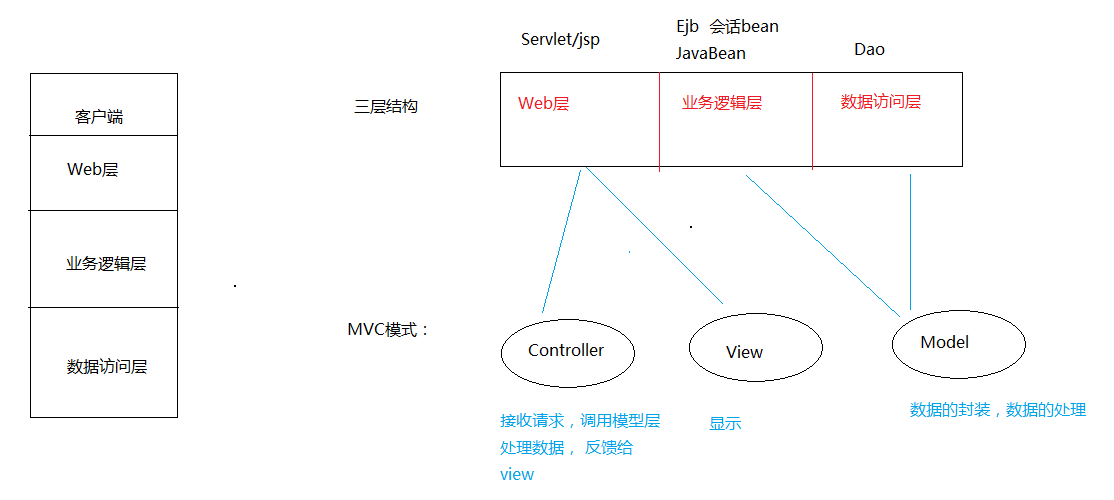
刷题学习



观看学车视频



* 1. 系统整体架构



* 1. **系统功能结构图**



* 1. 数据库设计

在本系统中，数据库系统应该解决如下需求：

* **保存用户信息，包括学员、教练资料等；**
* **保存试题信息、视频信息、资讯信息、车信息**
* **保存用户记录信息，包括成绩表，视频与资讯浏览信息，车订单等。**

根据以上系统需求，可以列出一下数据项和数据结构：

* **管理员信息表：包括用户名、密码等；**
* **学员信息表：包括用户名、密码、电子邮箱、身份证、电话号码、所属驾校、教练等**
* **教练信息表：包括用户名、密码、电子邮箱、身份证、电话号码、所属驾校、机动车证等**
* **驾校信息表：包括驾校名、所属地区、介绍**
* **试题表：包括题目类型、题目、答案、解释等**
* **成绩表：包括学员id、成绩、考试时间**
* **视频信息表：包括视频作者、视频标题、视频类型、上传时间等**
* **个人观看视频记录表：包括用户、视频id、观看时间等**
  1. 概念结构设计



* 1. 逻辑结构设计

**从前面ER图中得到下面关系模式：**

**管理员表（管理员id、用户名、密码、邮箱、电话）**

**学员信息表（学员id、用户名、密码、电子邮箱、身份证、电话号码、所属驾校、教练、头像、身份证图片）**

**教练信息表（教练id、用户名、密码、电子邮箱、身份证、电话号码、所属驾校、教练、头像、身份证图片、机动车证）**

**驾校信息表（驾校id、驾校名、所属地区、介绍）**

**试题表（试题id、题目、答案、解释、题目类型、选项）**

**成绩表（成绩id、学员id、成绩、考试时间）**

**视频信息表（视频id、视频作者、视频标题、视频类型、上传时间）**

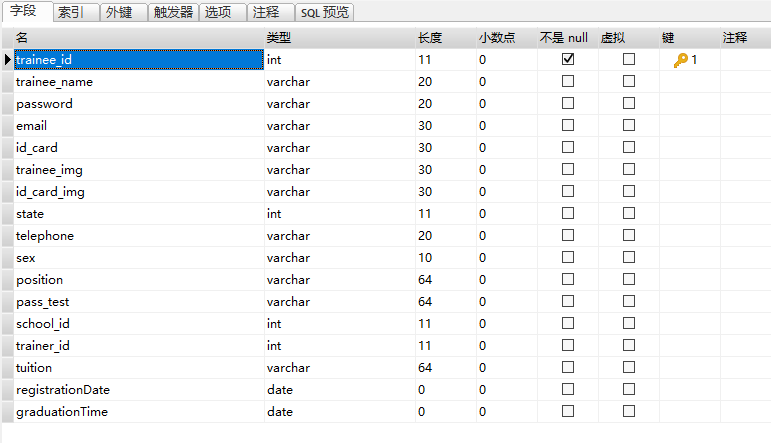
**个人观看视频记录表（视频记录id、用户、视频id、观看时间）**

* 1. 数据表结构设计

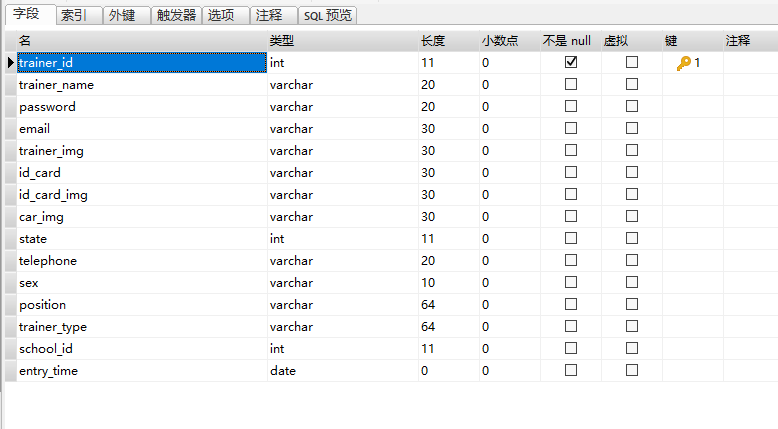
管理员表：



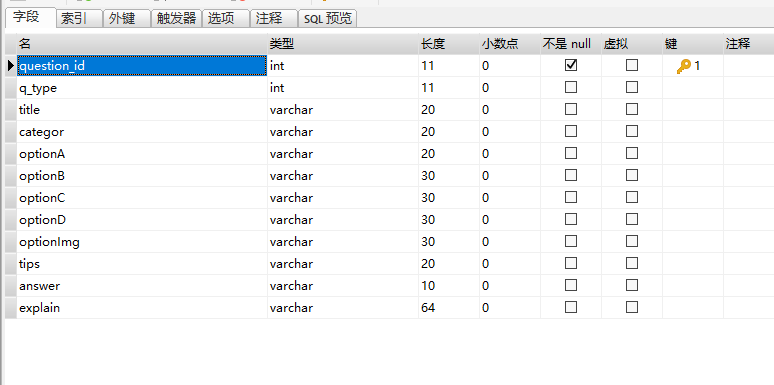
学员表：



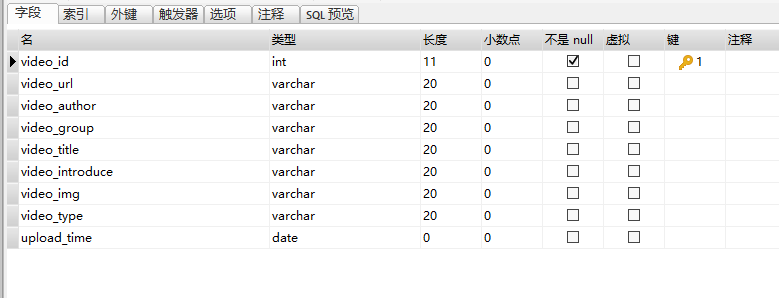
教练表：



试题表：



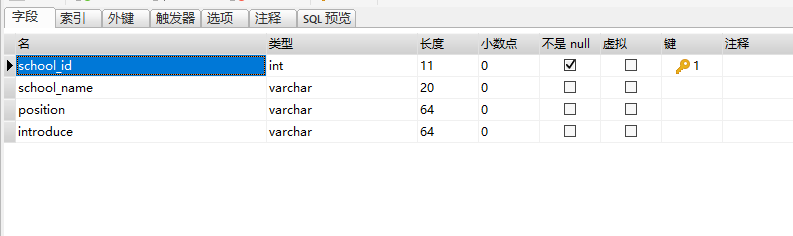
视频表：



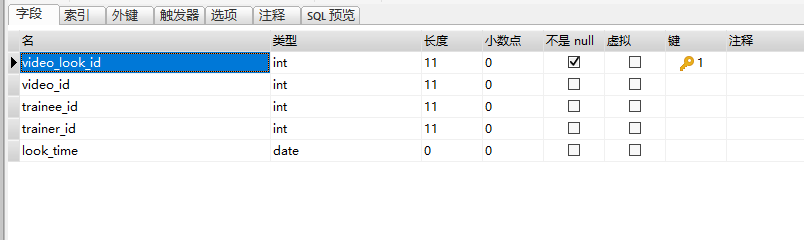
成绩表：



驾校表：



浏览记录表：



1. 编码规范
   1. 编程规约
      1. 命名风格

**1. 【强制】 代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。  
反例： \_name / \_\_name / $Object / name\_ / name$ / Object$  
2. 【强制】 代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。  
说明： 正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式  
也要避免采用。  
正例： alibaba / taobao / youku / hangzhou 等国际通用的名称， 可视同英文。  
反例： DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3  
3. 【强制】类名使用 UpperCamelCase 风格，必须遵从驼峰形式，但以下情形例外： DO / BO /  
DTO / VO / AO  
正例： MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion  
反例： macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion  
4. 【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从  
驼峰形式。  
正例： localValue / getHttpMessage() / inputUserId  
5. 【强制】常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。  
正例： MAX\_STOCK\_COUNT  
反例： MAX\_COUNT  
6. 【强制】抽象类命名使用 Abstract 或 Base 开头； 异常类命名使用 Exception 结尾； 测试类  
命名以它要测试的类的名称开始，以 Test 结尾。  
7. 【强制】中括号是数组类型的一部分，数组定义如下： String[] args;  
反例： 使用 String args[]的方式来定义。  
8. 【强制】 POJO 类中布尔类型的变量，都不要加 is，否则部分框架解析会引起序列化错误。  
反例： 定义为基本数据类型 Boolean isDeleted； 的属性，它的方法也是 isDeleted()， RPC框架在反向解析的时候， “以为”对应的属性名称是 deleted，导致属性获取不到，进而抛出异  
常。  
9. 【强制】包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用  
单数形式，但是类名如果有复数含义，类名可以使用复数形式。  
正例： 应用工具类包名为 com.alibaba.open.util、类名为 MessageUtils（此规则参考  
spring 的框架结构）  
10. 【强制】杜绝完全不规范的缩写， 避免望文不知义。  
反例： AbstractClass“缩写” 命名成 AbsClass； condition“缩写” 命名成 condi，此类  
随意缩写严重降低了代码的可阅读性。  
11. 【推荐】如果使用到了设计模式，建议在类名中体现出具体模式。  
说明： 将设计模式体现在名字中，有利于阅读者快速理解架构设计思想。  
正例： public class OrderFactory;  
public class LoginProxy;  
public class ResourceObserver;  
12. 【推荐】接口类中的方法和属性不要加任何修饰符号（public 也不要加） ，保持代码的简洁  
性，并加上有效的 Javadoc 注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是  
与接口方法相关，并且是整个应用的基础常量。  
正例： 接口方法签名： void f();  
接口基础常量表示： String COMPANY = "alibaba";  
反例： 接口方法定义： public abstract void f();  
说明： JDK8 中接口允许有默认实现，那么这个 default 方法，是对所有实现类都有价值的默  
认实现。  
13. 接口和实现类的命名有两套规则：  
1） 【强制】对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部  
的实现类用 Impl 的后缀与接口区别。  
正例： CacheServiceImpl 实现 CacheService 接口。  
2） 【推荐】 如果是形容能力的接口名称，取对应的形容词做接口名（通常是–able 的形式） 。  
正例： AbstractTranslator 实现 Translatable。  
14. 【参考】枚举类名建议带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。  
说明： 枚举其实就是特殊的常量类，且构造方法被默认强制是私有。  
正例： 枚举名字： DealStatusEnum， 成员名称： SUCCESS / UNKOWN\_REASON。  
15. 【参考】各层命名规约：  
A) Service/DAO 层方法命名规约  
1） 获取单个对象的方法用 get 做前缀**

**2） 获取多个对象的方法用 list 做前缀。  
3） 获取统计值的方法用 count 做前缀。  
4） 插入的方法用 save（推荐） 或 insert 做前缀。  
5） 删除的方法用 remove（推荐） 或 delete 做前缀。  
6） 修改的方法用 update 做前缀。  
B) 领域模型命名规约  
1） 数据对象： xxxDO， xxx 即为数据表名。  
2） 数据传输对象： xxxDTO， xxx 为业务领域相关的名称。  
3） 展示对象： xxxVO， xxx 一般为网页名称。  
4） POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。**

* + 1. 常量定义

**1. 【强制】不允许任何魔法值（即未经定义的常量） 直接出现在代码中。  
反例： String key = "Id#taobao\_" + tradeId;  
cache.put(key, value);  
2. 【强制】 long 或者 Long 初始赋值时，必须使用大写的 L，不能是小写的 l，小写容易跟数字  
1 混淆，造成误解。  
说明： Long a = 2l; 写的是数字的 21，还是 Long 型的 2?  
3. 【推荐】不要使用一个常量类维护所有常量，应该按常量功能进行归类，分开维护。如：缓存  
相关的常量放在类： CacheConsts 下； 系统配置相关的常量放在类： ConfigConsts 下。  
说明： 大而全的常量类，非得使用查找功能才能定位到修改的常量，不利于理解和维护。  
4. 【推荐】常量的复用层次有五层：跨应用共享常量、应用内共享常量、子工程内共享常量、包  
内共享常量、类内共享常量。  
1） 跨应用共享常量：放置在二方库中，通常是 client.jar 中的 constant 目录下。  
2） 应用内共享常量：放置在一方库的 modules 中的 constant 目录下。  
反例： 易懂变量也要统一定义成应用内共享常量，两位攻城师在两个类中分别定义了表示  
“是”的变量：  
类 A 中： public static final String YES = "yes";  
类 B 中： public static final String YES = "y";  
A.YES.equals(B.YES)，预期是 true，但实际返回为 false，导致线上问题。  
3） 子工程内部共享常量：即在当前子工程的 constant 目录下。  
4） 包内共享常量：即在当前包下单独的 constant 目录下。  
5） 类内共享常量：直接在类内部 private static final 定义。**

**5. 【推荐】如果变量值仅在一个范围内变化，且带有名称之外的延伸属性， 定义为枚举类。下面  
正例中的数字就是延伸信息，表示星期几。  
正例： public Enum { MONDAY(1), TUESDAY(2), WEDNESDAY(3), THURSDAY(4), FRIDAY(5), SATURDAY(6),  
SUNDAY(7);}**

* + 1. 代码格式

**1. 【强制】大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行； 如果  
是非空代码块则：  
1） 左大括号前不换行。  
2） 左大括号后换行。  
3） 右大括号前换行。  
4） 右大括号后还有 else 等代码则不换行； 表示终止的右大括号后必须换行。  
2. 【强制】 左小括号和字符之间不出现空格； 同样，右小括号和字符之间也不出现空格。详见  
第 5 条下方正例提示。  
反例： if (空格 a == b 空格)  
3. 【强制】 if/for/while/switch/do 等保留字与括号之间都必须加空格。  
4. 【强制】任何二目、 三目运算符的左右两边都需要加一个空格。  
说明： 运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号等。  
5. 【强制】 缩进采用 4 个空格，禁止使用 tab 字符。  
说明： 如果使用 tab 缩进，必须设置 1 个 tab 为 4 个空格。 IDEA 设置 tab 为 4 个空格时，  
请勿勾选 Use tab character；而在 eclipse 中，必须勾选 insert spaces for tabs。  
正例： （涉及 1-5 点）  
public static void main(String[] args) {  
// 缩进 4 个空格  
String say = "hello";  
// 运算符的左右必须有一个空格  
int flag = 0;  
// 关键词 if 与括号之间必须有一个空格，括号内的 f 与左括号， 0 与右括号不需要空格  
if (flag == 0) {  
System.out.println(say);  
}  
// 左大括号前加空格且不换行；左大括号后换行  
if (flag == 1) {  
System.out.println("world");  
// 右大括号前换行，右大括号后有 else，不用换行  
} else {  
System.out.println("ok");**

**6. 【强制】单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：  
1） 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。  
2） 运算符与下文一起换行。  
3） 方法调用的点符号与下文一起换行。  
4） 在多个参数超长， 在逗号后换行。  
5） 在括号前不要换行，见反例。  
正例：  
StringBuffer sb = new StringBuffer();  
//超过 120 个字符的情况下，换行缩进 4 个空格，并且方法前的点符号一起换行  
sb.append("zi").append("xin")...  
.append("huang")...  
.append("huang")...  
.append("huang");  
反例：  
StringBuffer sb = new StringBuffer();  
//超过 120 个字符的情况下，不要在括号前换行  
sb.append("zi").append("xin")...append  
("huang");  
//参数很多的方法调用可能超过 120 个字符， 不要在逗号前换行  
method(args1, args2, args3, ...  
, argsX);  
7. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。  
正例： 下例中实参的"a",后边必须要有一个空格。  
method("a", "b", "c");  
8. 【强制】 IDE 的 text file encoding 设置为 UTF-8; IDE 中文件的换行符使用 Unix 格式，  
不要使用 windows 格式。  
9. 【推荐】没有必要增加若干空格来使某一行的字符与上一行对应位置的字符对齐。  
正例：  
int a = 3;  
long b = 4L;  
float c = 5F;  
StringBuffer sb = new StringBuffer();  
说明： 增加 sb 这个变量，如果需要对齐，则给 a、 b、 c 都要增加几个空格，在变量比较多的  
情况下，是一种累赘的事情。  
10. 【推荐】方法体内的执行语句组、变量的定义语句组、不同的业务逻辑之间或者不同的语义  
之间插入一个空行。相同业务逻辑和语义之间不需要插入空行。  
说明： 没有必要插入多个空行进行隔开。**

* + 1. 注释规约

**1. 【强制】类、类属性、类方法的注释必须使用 Javadoc 规范，使用/\*\*内容\*/格式，不得使用  
//xxx 方式。  
说明： 在 IDE 编辑窗口中， Javadoc 方式会提示相关注释，生成 Javadoc 可以正确输出相应注  
释； 在 IDE 中，工程调用方法时，不进入方法即可悬浮提示方法、参数、返回值的意义，提高  
阅读效率。  
2. 【强制】所有的抽象方法（包括接口中的方法） 必须要用 Javadoc 注释、除了返回值、参数、  
异常说明外，还必须指出该方法做什么事情，实现什么功能。  
说明： 对子类的实现要求，或者调用注意事项，请一并说明。  
3. 【强制】所有的类都必须添加创建者和创建日期。  
4. 【强制】方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释  
使用/\* \*/注释，注意与代码对齐。  
5. 【强制】所有的枚举类型字段必须要有注释，说明每个数据项的用途。  
6. 【推荐】与其“半吊子”英文来注释，不如用中文注释把问题说清楚。专有名词与关键字保持  
英文原文即可。  
反例： “TCP 连接超时”解释成“传输控制协议连接超时”，理解反而费脑筋。  
7. 【推荐】代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑  
等的修改。  
说明： 代码与注释更新不同步，就像路网与导航软件更新不同步一样，如果导航软件严重滞后，  
就失去了导航的意义。  
8. 【参考】 合理处理注释掉的代码。 在上方详细说明，而不是简单的注释掉。 如果无用，则删除。  
说明： 代码被注释掉有两种可能性： 1） 后续会恢复此段代码逻辑。 2） 永久不用。前者如果没  
有备注信息，难以知晓注释动机。后者建议直接删掉（代码仓库保存了历史代码） 。  
9. 【参考】对于注释的要求：第一、能够准确反应设计思想和代码逻辑； 第二、能够描述业务含  
义，使别的程序员能够迅速了解到代码背后的信息。完全没有注释的大段代码对于阅读者形同  
天书，注释是给自己看的，即使隔很长时间，也能清晰理解当时的思路； 注释也是给继任者看  
的，使其能够快速接替自己的工作。  
10. 【参考】好的命名、代码结构是自解释的，注释力求精简准确、表达到位。避免出现注释的  
一个极端：过多过滥的注释，代码的逻辑一旦修改，修改注释是相当大的负担。  
反例：  
// put elephant into fridge  
put(elephant, fridge);  
方法名 put，加上两个有意义的变量名 elephant 和 fridge，已经说明了这是在干什么，语  
义清晰的代码不需要额外的注释。  
11. 【参考】特殊注释标记，请注明标记人与标记时间。注意及时处理这些标记，通过标记扫描，  
经常清理此类标记。线上故障有时候就是来源于这些标记处的代码。  
1） 待办事宜（TODO） :（ 标记人，标记时间， [预计处理时间]）  
表示需要实现，但目前还未实现的功能。这实际上是一个 Javadoc 的标签，目前的 Javadoc  
还没有实现，但已经被广泛使用。只能应用于类，接口和方法（因为它是一个 Javadoc 标签） 。  
2） 错误，不能工作（FIXME） :（标记人，标记时间， [预计处理时间]）  
在注释中用 FIXME 标记某代码是错误的，而且不能工作，需要及时纠正的情况。**

* 1. 其它

**1. 【强制】在使用正则表达式时，利用好其预编译功能，可以有效加快正则匹配速度。  
说明： 不要在方法体内定义： Pattern pattern = Pattern.compile(规则);  
2. 【强制】 velocity 调用 POJO 类的属性时，建议直接使用属性名取值即可， 模板引擎会自动按  
规范调用 POJO 的 getXxx()，如果是 boolean 基本数据类型变量（boolean 命名不需要加 is  
前缀） ，会自动调用 isXxx()方法。  
说明： 注意如果是 Boolean 包装类对象，优先调用 getXxx()的方法。  
3. 【强制】后台输送给页面的变量必须加$!{var}——中间的感叹号。  
说明： 如果 var=null 或者不存在，那么${var}会直接显示在页面上。  
4. 【强制】注意 Math.random() 这个方法返回是 double 类型，注意取值的范围 0≤x<1（能够  
取到零值，注意除零异常） ， 如果想获取整数类型的随机数，不要将 x 放大 10 的若干倍然后  
取整，直接使用 Random 对象的 nextInt 或者 nextLong 方法。  
5. 【强制】获取当前毫秒数 System.currentTimeMillis(); 而不是 new Date().getTime();  
说明： 如果想获取更加精确的纳秒级时间值， 使用 System.nanoTime()的方式。在 JDK8 中，  
针对统计时间等场景，推荐使用 Instant 类。  
6. 【推荐】 不要在视图模板中加入任何复杂的逻辑。  
说明： 根据 MVC 理论，视图的职责是展示，不要抢模型和控制器的活。  
7. 【推荐】 任何数据结构的构造或初始化，都应指定大小，避免数据结构无限增长吃光内存。  
8. 【推荐】对于“明确停止使用的代码和配置”，如方法、变量、类、配置文件、动态配置属性  
等要坚决从程序中清理出去，避免造成过多垃圾**

1. 项目总结与展望

驾校系统的实践，懂得项目对于我们开发人员来说，需要文档，管理，编码；也懂得需要一定的软件工程的方法，问题的定义-可行性分析-需求分析-编码-测试-维护，这几个步骤缺一不可相互制约，缺少一个项目或者哪一个环节出现失误、错漏都对于工程不能进行下去，工程化的开发思想需要学习的知识点很多，单独一个人都完成不了，所以需要一个团队。作为驾校系统的组长，需要协调组员，对于项目的分期计划，项目的部署，搭建等等工作很多，不过学到的东西也挺多的。

1. 参考文献

[1] 韩万江，姜立新，《软件工程（第3版）》，2018

[2] 本项目的《可行性分析》

[3] 李刚《JavaEE》