湖南电子科技职业学院
HUNAN VOCATIONAL COLLEGE OF ELECTRONIC AND TECHNOLOGY

产品设计	方案设计	工艺设计
	√	

信息工程学院

毕 业 设 计

题目：基于 Web 的二手车销售系统设计方案

学生姓名	潘文臻
学生学号	010425171774
班级名称	G32208 班
专业名称	计算机网络技术
指导教师	龙佳

2025 年 05 月

毕业设计真实性承诺及指导教师声明

本人郑重声明：所提交的毕业设计是本人在指导教师的指导下，独立进行研究工作所取得的成果，内容真实可靠，不存在抄袭、造假等学术不端行为。除毕业设计中已经注明引用的内容外，本设计不含其他个人或集体已经发表或撰写过的研究成果。对本毕业设计的研究做出重要贡献的个人和集体，均已在本设计中以明确方式标明。如被发现设计中存在抄袭、造假等学术不端行为，本人愿承担相应的法律责任和一切后果。

学生（签名）：潘文臻　　日　期：2025.5.11

指导教师关于学生毕业设计真实性审核的声明

本人郑重声明：已经对学生毕业设计所涉及的内容进行严格审核，确定其成果均由学生在本人指导下取得，对他人成果的引用已经明确注明，不存在抄袭等学术不端行为。

指导教师（签名）：[illegible]　　日　期：2025.5.12

（注：本页学生和指导教师须亲笔签名。）

目录

一、系统需求分析

（一）需求分析

在二手车销售系统中，线上交易是核心功能，商品更新速度快，用户购车决策时间也因人而异，有的可能长时间犹豫不决，有的则可能瞬间拍板。该系统主要面向两类用户：普通使用者和管理者。用户端功能包括注册、登录、浏览车辆列表、添加至购物车、下单以及查看订单；管理端则负责登录、管理用户信息、商品信息、分类以及订单。从用户和管理者的角度出发，经过综合分析，确定了二手车销售系统的以下模块需求。

图 1.1 系统整体用例图

（二）功能模块分析

1.用户注册

用户注册功能涵盖录入用户名、密码、姓名、电话、邮箱及详细地址等信息，并完

成注册流程。

图 1.2 用户注册用例图

用户注册用例描述如表 1.1 所示：

表 1.1 用户注册用例描述

用例编号	TANJING_001
用例名称	用户注册
用例描述	用户进行账号注册
参与者	用户
前置条件	拥有相应权限
后置条件	用户成功
基本路径	1. 进入系统主页，单击跳转注册页面 2. 弹出用户注册对话框 3. 按照提示的建议键入私人资料，单击注册 4. 系统保存用户信息

2.用户登录

用户登录功能：通过输入正确的用户名和密码，进行登录验证。

图 1.3 用户登录用例图

用户登录用例描述如表 1.2 所示：

表 1.2 用户登录用例描述

用例编号	TANJING_002
用例名称	用户登录
用例描述	用户进行账号登录
参与者	用户
前置条件	拥有相应权限
后置条件	用户成功
基本路径	1. 进入系统主页，点击登录按钮 2. 弹出用户登录对话框 3. 按照提示的建议键入资料，单击要登陆的登陆键 4. 进入系统主页

3.用户浏览商城列表

用户可通过输入关键字搜索，按不同分类浏览品牌车辆，查看热门车型及新上架车辆。

图 1.4 用户浏览商品列表用例图

用户浏览商品列表用例描述如表 1.3 所示：

表 1.3 用户浏览商品列表用例描述

用例编号	TANJING_003
用例名称	用户浏览商品列表
用例描述	用户进行商品浏览
参与者	用户
前置条件	拥有相应权限
后置条件	用户成功

基本路径	1. 用户进入系统主页 2. 输入想搜索的二手车品牌，点击搜索 3. 点击分类，选择品牌以及型号 4. 下滑系统主页，分为热门车和新上架 5. 点击加载更多，加载更多车辆直至全部加载完成

4.用户加入购物车

购物车功能支持用户将车辆添加至购物车，并可对购物车内的商品进行删除或修改操作：

图 1.5 用户加入购物车用例图

用户加入购物车用例描述如表 1.4 所示：

表 1.4 用户加入购物车用例描述

用例编号	TANJING_004
用例名称	用户加入购物车
用例描述	用户加入购物车以及删除购物车商品
参与者	用户
前置条件	拥有相应权限
后置条件	用户成功
基本路径	1. 进入系统主页 2. 点击进入商品主页 3. 点击加入购物车 4. 点击我的购物车，进入我的购物车页面 5. 点击删除，删除购物车内商品

5.用户下单

用户下单功能实现：输入用户相关信息：姓名，电话等，确认提交订单。

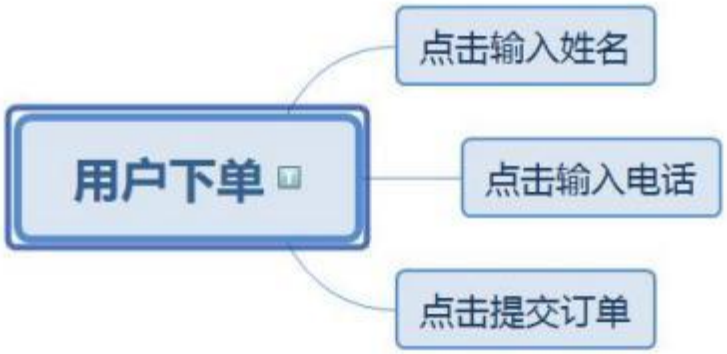

图 1.6 用户下单用例图

用户下单用例描述如表 1.5 所示：

表 1.5 用户下单用例描述

用例编号	TANJING_005
用例名称	用户下单
用例描述	用户下单心仪二手车
参与者	用户
前置条件	拥有相应权限
后置条件	用户成功
基本路径	1. 进入系统主页 2. 点击我的购物车进入购物车 3. 输入姓名和电话 4. 提交订单

6.用户查看订单

用户下单成功后，可查看订单详情，并进行订单取消操作。

图 1.7 用户查看订单用例图

用户查看订单用例描述如表 1.6 所示：

表 1.6 用户查看订单用例描述

用例编号	TANJING_006
用例名称	用户查看订单
用例描述	用户查看预订二手车选择是否取消预订
参与者	用户
前置条件	拥有相应权限
后置条件	用户成功
基本路径	1. 进入系统主页 2. 点击我的订单 3. 点击查看详情查看车辆详情 4. 点击取消订单弹出是否取消订单对话框

7.管理员登录

管理员登录功能实现：进行管理员登录验证和管理员账号密码重置功能。

图 1.8 管理员登录用例图

管理员登录用例描述如表 1.7 所示：

表 1.7 管理员登录用例描述

用例编号	TANJING_007
用例名称	管理员登录
用例描述	管理员登录二手车后台管理系统
参与者	管理员
前置条件	拥有相应权限
后置条件	管理员成功登录
基本路径	1. 进入二手车后台管理系统登陆网页 2. 输入指定账号和指定密码 3. 点击重置，重置指定账号和指定密码 4. 点击登录进入后台管理页面

8.管理员管理用户

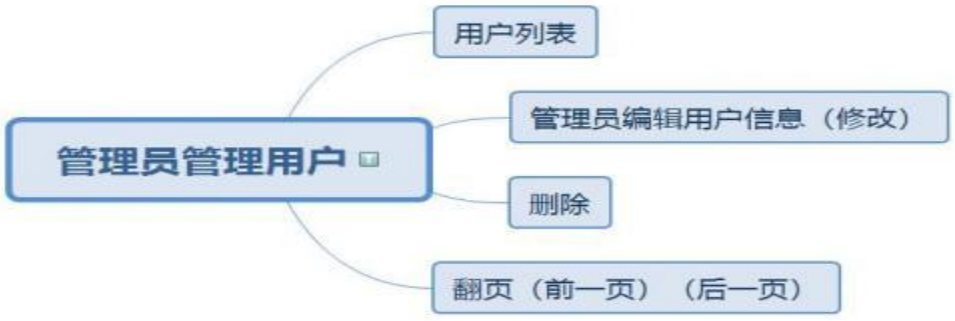

图 1.9 管理员管理用户用例图

管理员管理用户用例描述如表 1.8 所示：

表 1.8 管理员管理用户用例描述

用例编号	TANJING_008
用例名称	管理员管理用户
用例描述	管理员进入用户管理页面，删除和修改用户
参与者	管理员
前置条件	拥有相应权限
后置条件	管理员成功

基本路径	1. 进入用户管理页面 2. 查看所有用户信息 3. 点击编辑，修改用户信息 4. 点击删除，删除用户信息 5. 点击前一页后一页进行翻页操作

9.管理员管理分类

管理员管理分类功能实现：管理员管理进行二级分类，可以对二级分类进行增，删，改，查等操作。

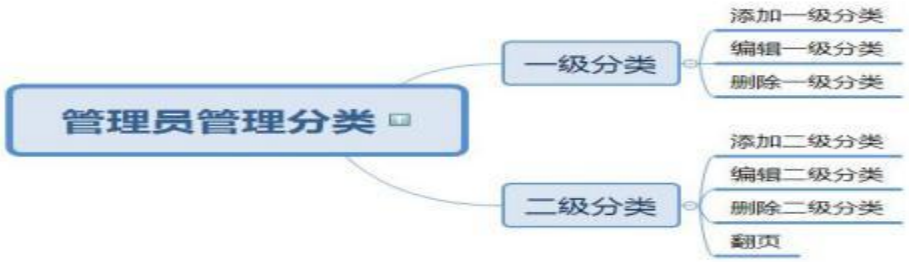

图 1.10 管理员管理分类用例图

管理员管理分类用例描述如表 1.9 所示：

表 1.9 管理员管理分类用例描述

用例编号	TANJING_009
用例名称	管理员管理分类
用例描述	管理员进入分类管理页面管理分类增删改查
参与者	管理员
前置条件	拥有相应权限
后置条件	管理员成功

基本路径	1. 进入分类管理页面 2. 进入一级分类 3. 点击添加，添加一级分类 4. 点击编辑，修改一级分类信息 5. 点击删除，删除一级分类信息 6. 进入二级分类 7. 点击添加，添加二级分类 8.点击编辑，修改二级分类信息 9.点击删除，删除二级分类信息 10. 点击前一页后一页进行翻页

10.管理员管理订单

管理员管理订单功能实现：所有订单列表显示，订单详情查看。

图 1.11 管理员管理订单用例图

管理员管理订单用例描述如表 1.10 所示：

表 1.10 管理员管理订单用例描述

用例编号	TANJING_010
用例名称	管理员管理订单
用例描述	管理员可以通过订单管理网页来看到订单。
参与者	管理员
前置条件	拥有相应权限

后置条件	管理员成功
基本路径	1. 进入订单管理页面 2. 查看所有订单信息 3. 点击订单详情下拉框，查看订单详情 4. 点击前一页后一页进行翻页操作

11.管理员管理商品

管理员管理商品实现功能：进行商品的添加，编辑，删除以及翻页操作。

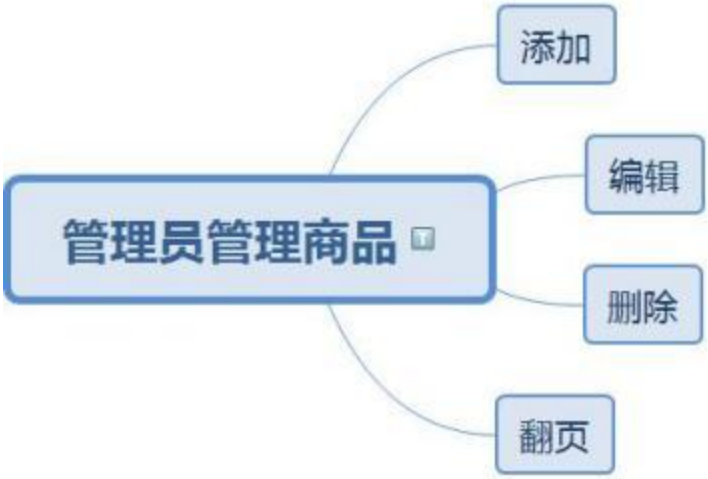

图 1.12 管理员管理商品用例图

管理员管理商品用例描述如表 1.11 所示：

表 1.11 管理员管理商品用例描述

用例编号	TANJING_011
用例名称	管理员管理商品
用例描述	管理员进入商品管理页面查看商品
参与者	管理员
前置条件	拥有相应权限
后置条件	管理员成功

基本路径	1. 进入商品管理页面 2. 查看所有商品信息 3. 点击添加，新增二手车 4. 点击编辑，编辑二手车信息 5. 点击删除，删除二手车信息 6. 页面下滑到最底部，点击数字或者下一页，进行翻页操作

二、系统设计

（一）系统总体设计

图 2.1 系统设计总揽

（二）数据库设计

1.数据库概念结构设计

根据二手车销售系统的需求，梳理出系统的关键实体及其属性如下：

（1）用户实体：用户 ID、用户名、用户身份证号码、用户密码、用户联系方式。

（2）管理员实体：管理员 ID、管理员编号、管理员密码、管理员联系方式。

（3）二手车实体：二手车 ID、二手车编号、二手车名称、二手车价格、车况描述。

（4）分类实体：分类 ID、分类编号、分类名称。

（5）订单实体：订单 ID、订单总价、用户身份证号码、用户联系方式、下单时间、订单详情、订单状态。

2.数据库表结构设计

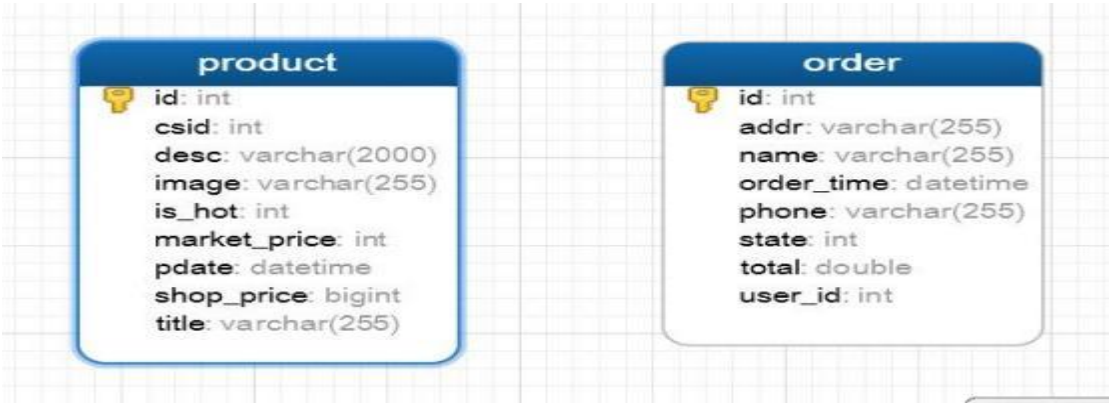

图 2.2 二手车表与订单表

图 2.3 用户表与订单详情表

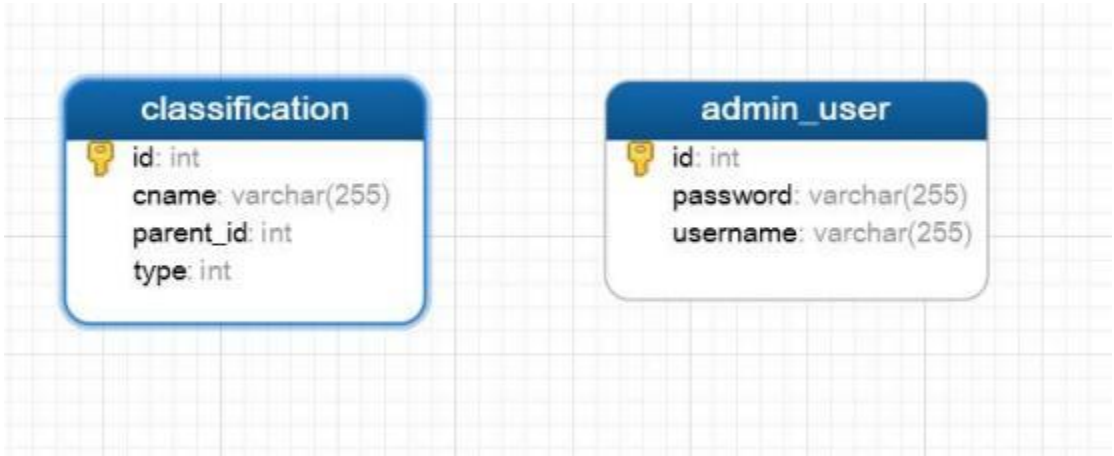

图 2.4 分类表与管理员表

三、功能界面展示及代码实现

（一）登录模块的显示

用户登录：用户进入系统后，不会直接显示登录入口，而是先展示二手车列表。当用户点击“加入购物车”按钮或右上角的登录按钮时，会弹出下拉框，用户可以选择登录或注册。已有账号的用户可直接登录，新用户需先完成注册再登录。

管理员登录：管理员登录界面仅提供登录入口，没有注册选项，只有指定账号可以登录。

- 

图 3.1 用户登录界面

图 3.2 用户注册界面

图 3.3 管理员登录界面

主要实现代码：

```
/**
 * 注册
 */
@RequestMapping("/register.do")
public void register(String username,
                     String password,
                     String name,
                     String phone,
                     String email,
                     String addr,
                     HttpServletResponse response) throws IOException {
    User user = new User();
    user.setUsername(username);
    user.setPhone(phone);
    user.setPassword(password);
    user.setName(name);
    user.setEmail(email);
    user.setAddr(addr);
    userService.create(user);
    // 注册完成后重定向到登录页面
    response.sendRedirect("/carshop/user/toLogin.html");
}

/**
 * 登出
 */
@RequestMapping("/logout.do")
public void logout(HttpServletRequest request, HttpServletResponse response) th
    request.getSession().removeAttribute("user");
    response.sendRedirect("/carshop/index.html");
}
/**
 * 登录
 *
 * @param username
 * @param password
 */
@RequestMapping("/login.do")
public void login(String username,
                  String password,
                  HttpServletRequest request,
                  HttpServletResponse response) throws IOException {
    User user = userService.checkLogin(username, password);
    if (user != null) {
        //登录成功 重定向到首页
        request.getSession().setAttribute("user", user);
        response.sendRedirect("/carshop/index.html");
    } else {
        throw new LoginException("登录失败！ 用户名或者密码错误");
    }
```

```
 */
@RequestMapping("/toLogin.html")
public String toLogin() {
    return "admin/login";
}

/**
 * 登录请求
 *
 * @param username
 * @param password
 */
//@ResponseBody
@RequestMapping(method = RequestMethod.POST, value = "/login.do")
public void login(String username, String password, HttpServletRequest request, Htt
    AdminUser adminUser = adminUserService.checkLogin(request, username, password);
    response.sendRedirect("/carshop/admin/toIndex.html");
}

/**
 * 退出登录
 * @param request
 * @param response
 * @throws IOException
 */
@RequestMapping("/logout.do")
public void logout(HttpServletRequest request, HttpServletResponse response) throws
    request.getSession().removeAttribute("login_user");
    response.sendRedirect("toLogin.html");
}
```

（二）系统主页面设计

系统主页面具备二手车分类、搜索以及热门车辆展示等功能。对于部分顾客而言，可能尚未确定心仪的车型，因此可以选择从头浏览至尾，逐一查看车辆信息；而对于那些心中已有明确品牌或具体车型的顾客，则可以通过分类功能快速筛选，或者直接使用搜索功能。系统主页面的布局和功能设计旨在满足不同用户的需求，提供便捷的浏览体验：

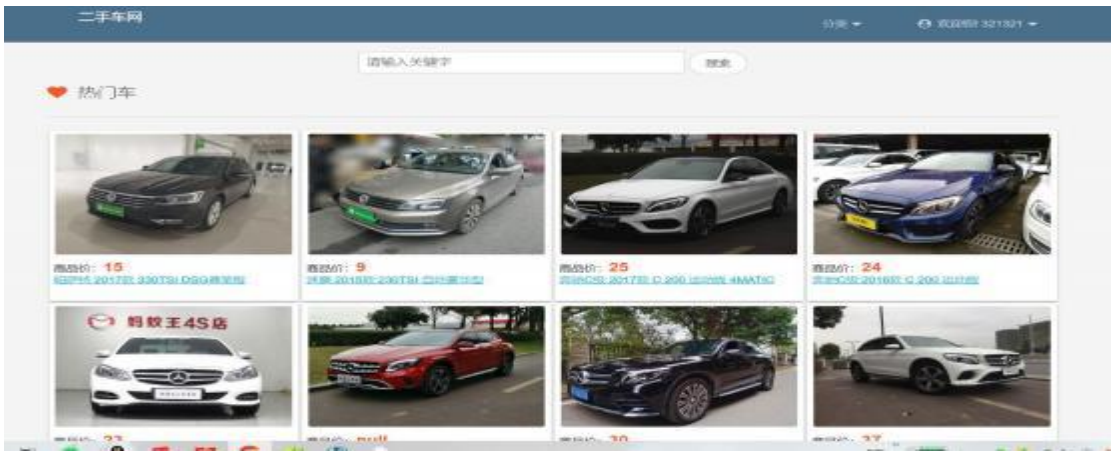

图 3.4 商品总览页面

主要实现代码：

```
mport javax.servlet.http.HttpServletRequest;
mport java.text.SimpleDateFormat;
mport java.util.ArrayList;
mport java.util.Date;
mport java.util.List;
mport java.util.Map;

Controller
RequestMapping("/product")
ublic class ProductController {
    @Autowired
    private ProductService productService;
    @Autowired
    private ClassificationService classificationService;
    @Autowired
    private ShopCartService shopCartService;
    @Autowired
    private MessageService messageService;

    /**
     * 关键字搜索
     *
     * @return
     */
    @RequestMapping("/serchPro.html")
    public String serchPro(String serchname,HttpServletRequest request) {
        request.getSession().setAttribute("serchname",serchname);
        return "carshop/product/product";
    }

    @ResponseBody
    @RequestMapping("/serchPro.do")
    public ResultBean<List<Product>> getSerchPro(HttpServletRequest request) {
        String serchname = (String)
        request.getSession().getAttribute("serchname");
        List<Product> products = new ArrayList<>();
        List<Product> list  = productService.findAll();
        for (Product product:list) {
            if(product.getTitle().contains(serchname)){
                products.add(product);
            }
        }
        return new ResultBean<>(products);
    }



    /**
     * 获取商品信息
     *
     * @param id
     * @return
     */
    @RequestMapping("/get.do")
    public ResultBean<Product> getProduct(int id) {
        Product product = productService.findById(id);
        return new ResultBean<>(product);
    }
```

```
     * 查找热门商品
     *
     * @return
     */
    @ResponseBody
    @RequestMapping("/hot.do")
    public ResultBean<List<Product>> getHotProduct() {
        List<Product> products = productService.findHotProduct();
        return new ResultBean<>(products);
    }

    /**
     * 查找最新商品
     *
     * @param pageNo
     * @param pageSize
     * @return
     */
    @ResponseBody
    @RequestMapping("/new.do")
    public ResultBean<List<Product>> getNewProduct(int pageNo, int pageSize) {
        Pageable pageable = new PageRequest(pageNo, pageSize);
        List<Product> products = productService.findNewProduct(pageable);
        return new ResultBean<>(products);
    }

    /**
     * 打开分类查看商品页面
     *
     * @return
     */
    @RequestMapping("/category.html")
    public String toCatePage(int cid, Map<String, Object> map) {
        Classification classification = classificationService.findById(cid);
        map.put("category", classification);
        return "carshop/product/category";
    }

    @RequestMapping("/toCart.html")
    public String toCart(){
        return "carshop/product/cart";
    }

    /**
     * 按一级分类查找商品
     *
     * @param cid
     * @param pageNo
     * @param pageSize
     * @return
     */
    @ResponseBody
    @RequestMapping("/category.do")
    public ResultBean<List<Product>>
```

（三）购物车模块设计

购物车模块主要实现车辆的添加、删除以及筛选功能。当顾客遇到心仪的二手车时，可将其加入购物车，便于后续集中对比和选择。在准备购车时，顾客可以在购物车内进行筛选，确定最终购买的车辆后提交订单，同时可删除其他不需要的车辆。这种设计既方便了顾客的决策过程，又提高了购物效率：

图 3.5 购物车页面

主要实现代码：

```
*/
public interface ShopCartService {

    String NAME_PREFIX = "shop_cart_";

    /**
     * 加购物车
     * @param
     */
    void addCart(int productId, HttpServletRequest request) throws Exception;

    void addCart1(int productId, HttpServletRequest request) throws Exception;

    /**
     * 移除
     * @param productId
     * @param request
     */
    void remove(int productId, HttpServletRequest request) throws Exception;
    void remove1(int productId, HttpServletRequest request) throws Exception;
    /**
     * 查看购物车
     * @param request
     * @return
     */
    List<OrderItem> listCart(HttpServletRequest request) throws Exception;

    void updateOrderItem(OrderItem orderItem);

    OrderItem selectOrderItem(int id);
```

（四）订单模块设计

订单模块支持查看订单详情和取消订单的功能。在“我的订单”页面，用户可以查看当前正在交易的二手车信息，包括车辆的详细情况。如果用户对某辆车不再感兴趣，可以随时取消订单。这种灵活的订单管理方式，能够满足用户在购车过程中的各种需求，提升用户体验：

图 3.6 订单页面

主要相关代码：

```
/**
 * 查询用户订单列表
 *
 * @param request
 * @return
 */
@RequestMapping("/list.do")
@ResponseBody
public ResultBean<List<Order>> listData(HttpServletRequest request) {
    List<Order> orders = orderService.findUserOrder(request);
    return new ResultBean<>(orders);
}

/**
 * 查询订单详情
 *
 * @param orderId
 * @return
 */
@RequestMapping("/getDetail.do")
@ResponseBody
public ResultBean<List<OrderItem>> getDetail(int orderId) {
    List<OrderItem> orderItems = orderService.findItems(orderId);
    return new ResultBean<>(orderItems);
}
```

（五）系统管理员管理用户界面

管理员在用户管理模块中的主要职责是对用户信息进行修改与删除操作。由于购买二手车的用户可能在完成交易后不再进行后续购买，因此管理员需要定期清理这类不再活跃的用户数据，以保持系统的整洁和高效。此外，当用户忘记密码时，管理员可以通过后台直接为其重置密码，确保用户能够顺利恢复正常使用。管理员管理用户模块的界面设计如下所示：

图 3.7 用户管理页面

主要实现代码：

```
/**
 * 打开用户列表页面
 * @return
 */
@RequestMapping("/toList.html")
public String toList() {
    return "admin/user/list";
}

/**
 * 打开编辑页面
 * @param id
 * @param map
 * @return
 */
@RequestMapping("/toEdit.html")
public String toEdit(int id, Map<String, Object> map) {
    User user = userService.findById(id);
    map.put("user", user);
    return "admin/user/edit";
}
```

```
 * @return
 */
@ResponseBody
@RequestMapping("/list.do")
public ResultBean<List<User>> findAllUser
    (int pageindex,
        @RequestParam(value = "pageSize", defaultValue = "15") int pageSize)
    Pageable pageable = new PageRequest(pageindex, pageSize, null);
    List<User> users = userService.findAll(pageable).getContent();
    return new ResultBean<>(users);
}

@ResponseBody
@RequestMapping("/getTotal.do")
public ResultBean<Integer> geTotal() {
    Pageable pageable = new PageRequest(1, 15, null);
    int total = (int) userService.findAll(pageable).getTotalElements();
    return new ResultBean<>(total);
}

@ResponseBody
@RequestMapping("/del.do")
public ResultBean<Boolean> del(int id) {
    userService.delById(id);
    return new ResultBean<>(true);
}

@ResponseBody
@RequestMapping(method = RequestMethod.POST, value = "/update.do")
public ResultBean<Boolean> update(int id,String username,
                                  String password,String name,
                                  String phone,String email,
                                  String addr) {
    // 更新前先查询
    User user = userService.findById(id);
    user.setId(id);
    user.setName(name);
    user.setUsername(username);
    user.setPassword(password);
    user.setAddr(addr);
    user.setEmail(email);
    user.setPhone(phone);
    userService.update(user);
    return new ResultBean<>(true);
}
```

（六）系统管理员管理分类模块

管理员在分类管理模块中主要负责一级分类与二级分类的增加、删除和修改操作。为了确保系统能够及时反映市场动态和车辆类型的变化，管理员可以灵活调整分类体系。例如，当出现新的车辆类型时，管理员可以新增相应的分类；如果某个分类长时间没有对应的二手车，管理员可以将其移除。相关页面如下所示：

图 3.8 一级分类页面

图 3.9 二级分类页面

主要实现代码：

```
 * 打开添加分类页面
 *
 * @param type
 * @return
 */
@RequestMapping("/toAdd.html")
public String toAdd(int type) {
    if (type == 1) {// 一级分类页面
        return "admin/category/add";
    } else if (type == 2) {// 二级分类页面
        return "admin/categorysec/add";
    } else {
        return "";
    }
}

/**
 * 打开编辑页面
 *
 * @param id
 * @param type
 * @param map
 * @return
 */
@RequestMapping("/toEdit.html")
public String toEdit(int id, int type, Map<String, Object> map) {
    Classification classification = classificationService.findById(id);
    map.put("cate", classification);
    if (type == 1) {// 一级分类页面
        return "admin/category/edit";
    } else if (type == 2) {// 二级分类页面
        Classification classification1 =
            classificationService.findById(classification.getParentId());
        map.put("cate", classification1);
        map.put("catese",classification);
        return "admin/categorysec/edit";
    } else {
        return "";
    }
}

@ResponseBody
@RequestMapping(method = RequestMethod.POST, value = "/add.do")
public ResultBean<Boolean> add(String cname, int parentId, int type) {
    Classification classification = new Classification();
    classification.setCname(cname);
    classification.setParentId(parentId);
    classification.setType(type);
    classificationService.create(classification);
    return new ResultBean<>(true);
}

@ResponseBody
@RequestMapping(method = RequestMethod.POST, value = "/update.do")
public ResultBean<Boolean> update(int id, String cname, int parentId, int t
    Classification classification = classificationService.findById(id);
    classification.setCname(cname);
    classification.setParentId(parentId);
    classification.setType(type);
    classificationService.update(classification);

@RequestMapping("/toList.html")
public String toList(int type) {
    if (type == 1) {// 一级分类页面
        return "admin/category/list";
    } else if (type == 2) {// 二级分类页面
        return "admin/categorysec/list";
    } else {
        return "";
    }
}
```

（七）系统管理员管理订单模块

管理员在订单管理模块中主要负责查询订单信息。由于订单情况复杂多变，顾客的行为也难以完全预测，例如可能会出现下单后又迅速取消的情况。订单管理页面能够帮助管理员更好地监控订单动态，及时处理异常情况。相关页面如下所示：

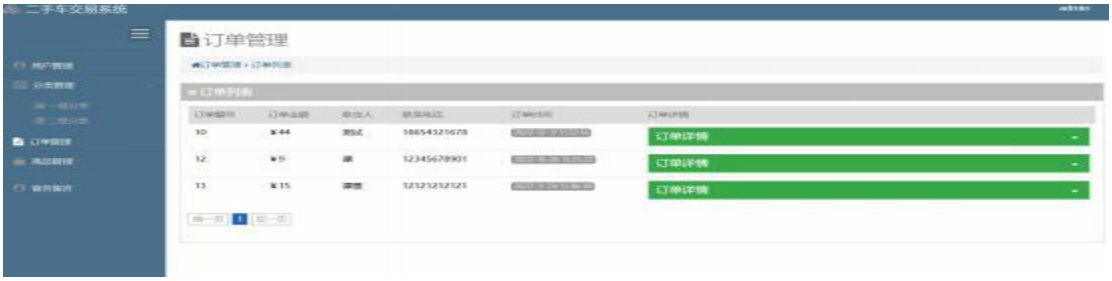

图 3.10 订单管理页面

主要实现代码：

```
/**
 * 打开订单列表页面
 * @return
 */
@RequestMapping("/toList.html")
public String toList() {
    return "admin/order/list";
}

/**
 * 获取所有订单的总数
 * @return
 */
@ResponseBody
@RequestMapping("/getTotal.do")
public ResultBean<Integer> getTotal() {
    Pageable pageable = new PageRequest(1, 15, null);
    int total = (int) orderService.findAll(pageable).getTotalElements();
    return new ResultBean<>(total);
}

/**
 * 获取所有订单
 * @param pageindex
 * @param pageSize
 * @return
 */
@ResponseBody
@RequestMapping("/list.do")
```

```
/**
 * 获取所有订单
 * @param pageindex
 * @param pageSize
 * @return
 */
@ResponseBody
@RequestMapping("/list.do")
public ResultBean<List<Order>>
    listData(int pageindex,
    @RequestParam(value = "pageSize", defaultValue = "15") int pageSize) {
    Pageable pageable = new PageRequest(pageindex, pageSize, null);
    List<Order> list = orderService.findAll(pageable).getContent();
    return new ResultBean<>(list);
}

/**
 * 获取订单项
 * @param orderId
 * @return
 */
@ResponseBody
@RequestMapping("/getDetail.do")
public ResultBean<List<OrderItem>> getDetail(int orderId) {
    List<OrderItem> list = orderService.findItems(orderId);
    return new ResultBean<>(list);
}

/**
 * 发货
 * @param id
 * @return
 */
@ResponseBody
@RequestMapping("/send.do")
public ResultBean<Boolean> send(int id) {
    orderService.updateStatus(id,3);
    return new ResultBean<>(true);
}
```

（八）系统管理员管理商品模块

管理员在商品管理模块中主要负责商品的新增、修改和删除操作。在二手车销售业务中，管理员需要灵活调整商品信息，以适应市场需求。例如，管理员可以将新回收的二手车添加到系统中，也可以将滞销的车辆移除，甚至在信息录入错误时进行修改。这种灵活的管理方式有助于保持系统的高效运行。相关页面如下所示：

图 3.11 商品管理页面

主要实现代码：

```
                String desc,
                int csid,
                HttpServletRequest request,
                HttpServletResponse response) throws Exception {
    Product product = new Product();
    product.setTitle(title);
    product.setMarketPrice(marketPrice);
    product.setShopPrice(shopPrice);
    product.setDesc(desc);
    product.setIsHot(isHot);
    product.setCsid(csid);
    product.setPdate(new Date());
    String imgUrl = FileUtil.saveFile(image);
    product.setImage(imgUrl);
    int id = productService.create(product);
    if (id <= 0) {
        request.setAttribute("message", "添加失败！");
        request.getRequestDispatcher
            ("toAdd.html").forward(request, response);
    } else {
        request.getRequestDispatcher
            ("toEdit.html?id=" + id).forward(request, response);
    }
}
```

```
@Autowired
private ProductService productService;
@Autowired
private ClassificationService classificationService;

@RequestMapping("/toList.html")
public String toList() {
    return "admin/product/list";
}

@RequestMapping("/toAdd.html")
public String toAdd() {
    return "admin/product/add";
}

@RequestMapping("/toEdit.html")
public String toEdit(int id, Map<String, Object> map) {
    Product product = productService.findById(id);
    Classification classification =
    classificationService.findById(product.getCsid());
    product.setCategorySec(classification);
    map.put("product", product);
    return "admin/product/edit";
}

@ResponseBody
@RequestMapping("/list.do")
public ResultBean<List<Product>>
    listProduct(int pageindex,
    @RequestParam(value = "pageSize", defaultValue = "15") int pageSiz
    Pageable pageable = new PageRequest(pageindex, pageSize, null);
    List<Product> list = productService.findAll(pageable).getContent()
    return new ResultBean<>(list);
}
```

```
@RequestMapping(method = RequestMethod.POST, value = "/update.do")
public void update(int id,
                   String title,
                   Double marketPrice,
                   Double shopPrice,
                   String desc,
                   int csid,
                   int isHot,
                   MultipartFile image,
                   HttpServletRequest request,
                   HttpServletResponse response) throws Exception {
    Product product = productService.findById(id);
    product.setTitle(title);
    product.setMarketPrice(marketPrice);
    product.setShopPrice(shopPrice);
    product.setDesc(desc);
    product.setIsHot(isHot);
    product.setCsid(csid);
    product.setPdate(new Date());
    String imgUrl = FileUtil.saveFile(image);
    if (StringUtils.isNotBlank(imgUrl)) {
        product.setImage(imgUrl);
    }
    boolean flag = false;
    try {
        productService.update(product);
        flag = true;
    } catch (Exception e) {
        throw new Exception(e);
    }
    if (!flag) {
        request.setAttribute("message", "更新失败！");
    }
    response.sendRedirect("toList.html");
}
```

四、系统测试

（一）测试方案

为确保系统的稳定性和可用性，必须进行全面的功能测试，以排查可能影响用户体验的漏洞。以下是详细的测试方案：

（1）登录模块测试：

功能测试：验证正确输入用户名和密码时能否成功登录，以及输入错误凭证时的提示信息是否准确。

安全性测试：检查登录模块是否能够抵御 SQL 注入、XSS 攻击等常见的安全威胁。

兼容性测试：测试登录功能在不同浏览器（如 Chrome、Firefox、Safari）和设备（如 PC、平板、手机）上的表现。

（2）用户搜索功能测试：

功能性测试：验证用户输入关键词后能否返回相关结果，包括不同类型的搜索词和搜索结果的排序。

边界值测试：测试搜索框对空输入、超长输入以及特殊字符的处理能力。

性能测试：模拟高并发场景，测试搜索功能的响应时间和系统稳定性。

安全性测试：确保搜索功能不会泄露用户隐私或成为安全漏洞的入口。

（3）购物车功能测试：

功能测试：验证用户能否顺利将商品加入购物车，以及删除和修改购物车中的商品。

性能测试：测试购物车在高并发情况下的响应速度和稳定性。

用户体验测试：评估购物车界面的交互设计是否友好，操作是否便捷。

（4）分类功能测试：

功能测试：验证分类的增删改功能是否正常，以及分类信息是否能够正确显示。

用户体验测试：评估分类导航的易用性和直观性。

（5）订单功能测试：

功能测试：验证订单的创建、查询和取消功能是否正常。

性能测试：测试订单处理的响应时间和系统稳定性。

安全性测试：确保订单信息的安全性和隐私保护。

（6）管理员管理用户测试：

功能测试：验证管理员能否顺利修改和删除用户信息。

安全性测试：确保管理员操作的权限控制和数据安全性。

（7）管理员管理分类测试：

功能测试：验证管理员对分类的增删改操作是否正常。

用户体验测试：评估分类管理界面的操作便捷性和直观性。

（8）管理员管理商品测试：

功能测试：验证商品的新增、修改和删除功能是否正常。

性能测试：测试商品管理在高并发情况下的响应速度和稳定性。

用户体验测试：评估商品管理界面的交互设计是否友好。

（二）测试用例

1.登录的测试用例

功能名称	用户登录		
测试目的	确保用户能够正常登录。		
预置条件	系统存在账号 321321，密码 321321 系统不存在账号 000000 密码 011128		
用例编号	操作步骤	输入数据	期望结果
USER_001	点击登录	账号：321321 密码：321321	成功登录进入系统主页
USER_002	点击登录	账号：000000 密码：011128	系统提示用户名或密码错误

2. 搜索功能的测试用例

功能名称	搜索		
测试目的	确保搜索功能能正常搜索。		
预置条件	系统存在商品奥迪 系统不存商品兰博基尼		
用例编号	操作步骤	输入数据	期望结果
SEARCH_001	点击搜索	奥迪	搜索出相关商品
SEARCH_002	点击搜索	兰博基尼	搜索结果为空

3. 购物车功能的测试用例

功能名称	购物车		
测试目的	确保购物车功能能够正常操作。		
预置条件	系统有可以添加到购物车中的商品 购物车存在能够删除的商品		
用例编号	操作步骤	输入数据	期望结果
TROLLEY_001	点击加入购物车		购物车内出现商品
TROLLEY_002	点击删除		原本购物车内的商品消失

4. 分类功能的测试用例

功能名称	分类		
测试目的	确保分类功能可以正常操作。		
预置条件	系统函数网页存在足够的二手车		
用例编号	操作步骤	输入数据	期望结果
CLASSIFY_001	点击一级分类		筛选出一级分类商品
CLASSIFY_002	点击二级分类		筛选出二级分类商品

5. 订单功能的测试用例

功能名称	订单		
测试目的	确保订单功能可以正常操作。		
预置条件	购物车存在可以预定的商品		
用例编号	操作步骤	输入数据	期望结果
ORDER_001	在购物车点击提交订单	数据订单	订单列表出现新订单
ORDER_002	在订单列表点击取消订单		订单列表取消指定单

6. 管理用户功能的测试用例

功能名称	管理用户		
测试目的	确保管理员管理用户的功能可以正常操作。		
预置条件	有登记的使用者存在于该系统中		
用例编号	操作步骤	输入数据	期望结果
ADMIN_USER_001	点击编辑	用户 id：id10100 用户名称：user01 用户身份证代号： 430312197809082190 用户密码：123456 用户联系方式： 18576439087	能正常编辑用户信息
ADMIN_USER_002	点击删除		能删除用户信息

7. 管理分类功能的测试用例

功能名称	管理分类		
测试目的	确保管理员管理的分类能正常操作。		
预置条件	管理员页面有分类按钮		
用例编号	操作步骤	输入数据	期望结果
ADMIN_SORT_001	点击一级分类添加		能添加一级分类
ADMIN_SORT_002	点击一级分类删除		能删除一级分类
ADMIN_SORT_003	点击一级分类编辑		能编辑一级分类
ADMIN_SORT_004	点击二级分类添加		能添加二级分类
ADMIN_SORT_005	点击二级分类删除		能删除二级分类
ADMIN_SORT_006	点击二级分类编辑		能编辑二级分类

8. 管理商品功能的测试用例

功能名称	管理商品		
测试目的	确保管理员管理的商品能正常操作		
预置条件	管理商品页面存在		
用例编号	操作步骤	输入数据	期望结果
GOODS_001	点击添加		能添加商品
GOODS_002	点击删除		能删除商品
GOODS_003	点击编辑		能编辑商品信息

（三）测试结论

经过测试，发现部分功能有些小问题，不影响正常使用，经过调试之后这些问题也已经基本解决，后续测试结果全部通过，经过此次测试，我发现我在开发时有些地方做的不够好，后续经过调试，让系统的运行和操作变的更稳定。

参考资料

[1]袁莹静,陈婷,陈龙,周芷仪,谢鹏辉.基于 Web 的二手车交易系统的设计与实现[J].软件,2023,41(04):195-199.

[2]左志宇.基于 Android 插件化的二手车销售系统研究与实现[D].辽宁大学,2022.

[3]李浩明.二手车交易平台的设计与实现[J].现代信息科技,2022,6(23):21-24.

[4]王滨.汽车销售系统数据库设计与实现[J].网络安全技术与应用,2024,(07):50-52.

[5]丁海洋,王昊翔,姚全珠.基于 MVVM 框架的汽车销售管理系统设计与实现[J].电子制作,2024,(22):55-56+8.

[6]王辰夏.瀚众鑫辰汽车销售管理系统设计与实现[D].大连理工大学,2021.

[7]杨维娜,姜军霞.二手车交易系统中数据挖掘技术应用研究[J].现代信息科技,2022,6(16):142-144.

[8]陈冰.数据挖掘技术在二手车交易系统中的应用[J].计算机技术与发展,2023,30(05):180-184.

[9]崔臣,宋甲旭.基于 SpringBoot 的校园二手交易系统研究[J].无线互联科技,2023,20(18):31-34.

[10]王志亮,纪松波.基于 SpringBoot 的 Web 前端与数据库的接口设计[J].工业控制计算机,2023,36(03):51-53.