# 设计文档：Web Order 系统

## 1. 概述

项目名称: Web Order 系统

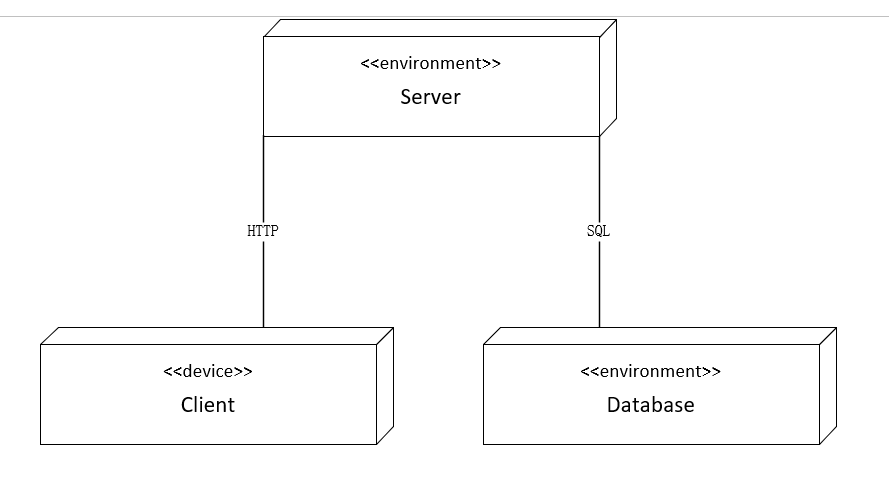
项目背景: 本项目旨在为用户提供在线乐器购买服务，系统将支持从乐器浏览、购物篮管理到提交订单、评价商品等功能。

项目目标: 开发一个稳定、高效、可扩展的乐器订单系统，满足用户从登录、浏览商品到订单提交的全流程需求。

## 2. 系统架构设计

系统将基于前后端分离的架构设计，前端使用Vue.js框架，后端采用Spring boot进行业务逻辑处理，数据库采用MySQL。系统还将部署在云平台（如AWS或Azure），支持高并发和扩展性。

### 2.1 系统架构图



- 前端：

- 用户界面通过 RESTful API 与后端交互，负责显示页面和数据渲染。

- 采用响应式设计，支持PC端和移动端访问。

- 后端：

- 处理业务逻辑，如用户登录、订单管理、商品查询等。

- 提供 RESTful API 接口供前端调用。

- 数据库：

- 存储用户、商品、订单、评价等数据。

- 采用关系型数据库，支持事务处理和高效查询。

- 服务器：

- 云服务器用于托管应用，保证高可用性和扩展性。

- 采用负载均衡和自动伸缩策略，确保性能和稳定性。

## 3. 数据库设计

### 3.1 数据库实体设计

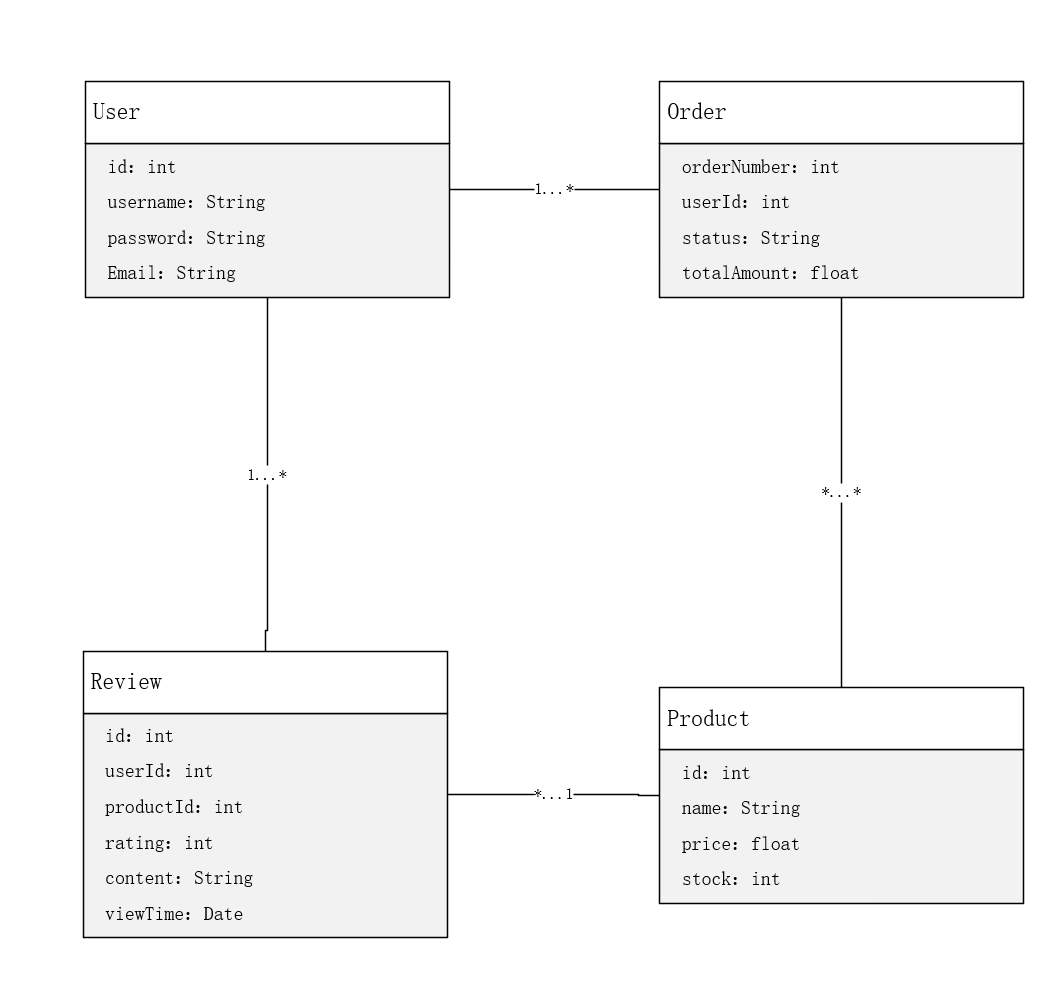
数据库将包含以下主要实体：

1. 用户表（User）：存储用户信息，包括用户名、密码、邮箱等。

2. 商品表（Product）：存储商品信息，包括乐器名称、价格、描述、库存等。

3. 订单表（Order）：存储订单信息，包括订单号、用户ID、订单状态、总金额等。

4. 评价表（Review）：存储用户对商品的评价，包括评分、评论内容、评价时间等。



### 3.2 数据库表设计

| 表名 | 字段 | 数据类型 | 描述 |

|-----------|-----------------------|----------|----------------------|

| User | user\_id, username, ... | INT, ... | 存储用户基本信息 |

| Product | product\_id, name, ... | INT, ... | 存储乐器相关信息 |

| Order | order\_id, user\_id, ... | INT, ... | 存储订单相关信息 |

| Review | review\_id, product\_id, ... | INT, ... | 存储用户的乐器评价 |

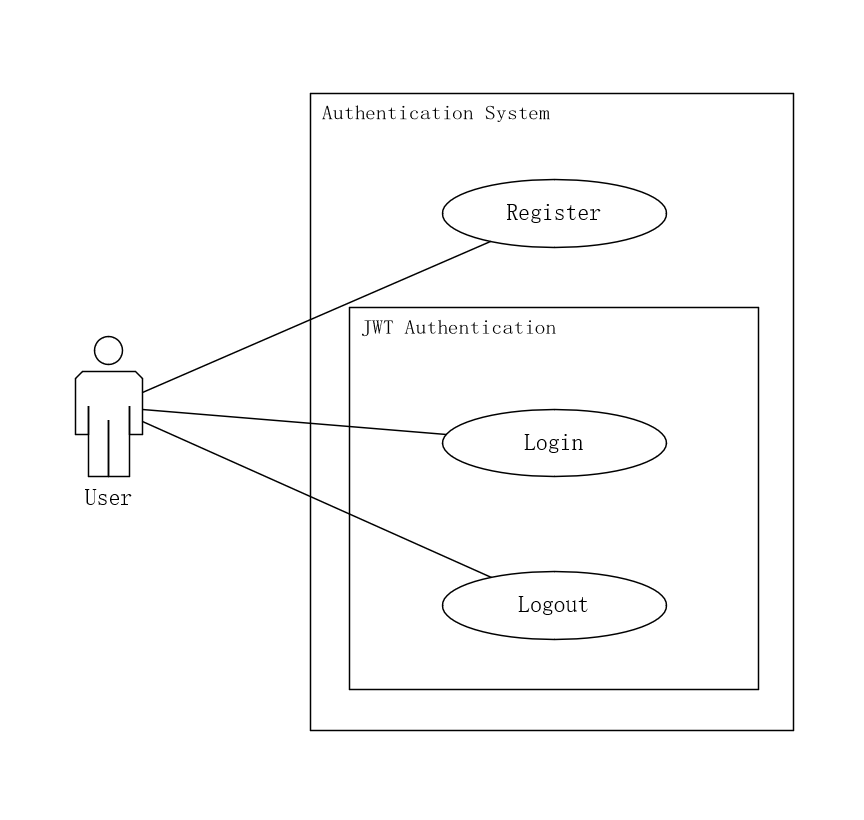
## 4. 模块设计

### 4.1 用户管理模块

主要功能：

- 用户注册、登录、登出。

- 使用JWT（JSON Web Token）进行身份认证，保护用户信息。



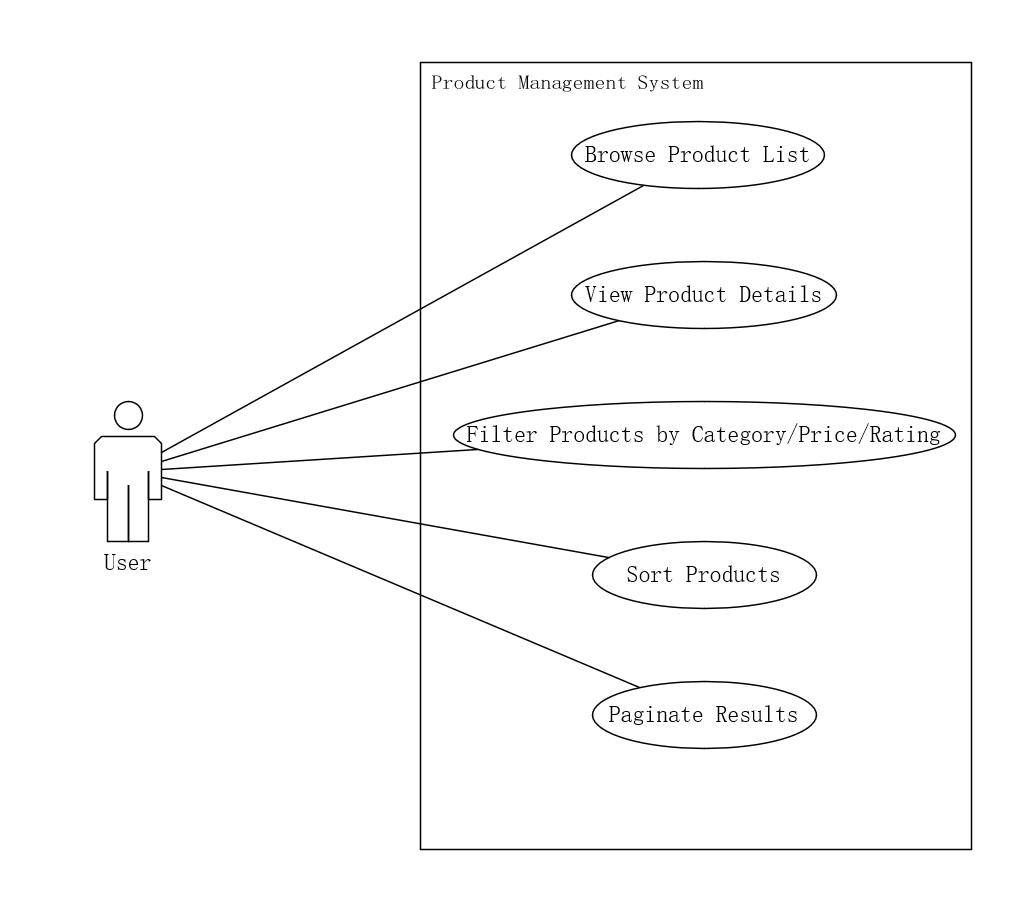
### 4.2 商品管理模块

主要功能：

- 浏览商品列表和详情。

- 支持通过不同类别、价格、评价进行筛选和排序。

- 提供分页功能，优化查询性能。



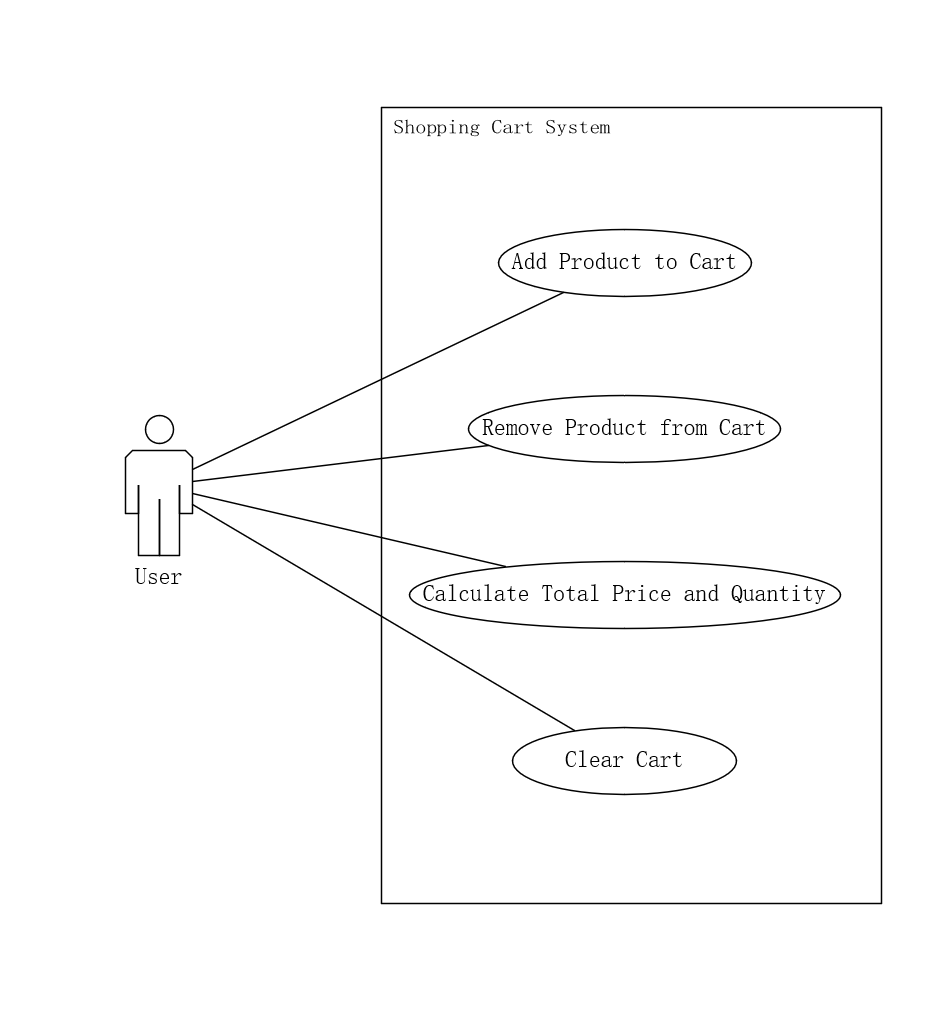
### 4.3 购物篮模块

主要功能：

- 用户可以添加、移除商品到购物篮。

- 实时计算购物篮中的商品总价和数量。

- 支持清空购物篮操作。



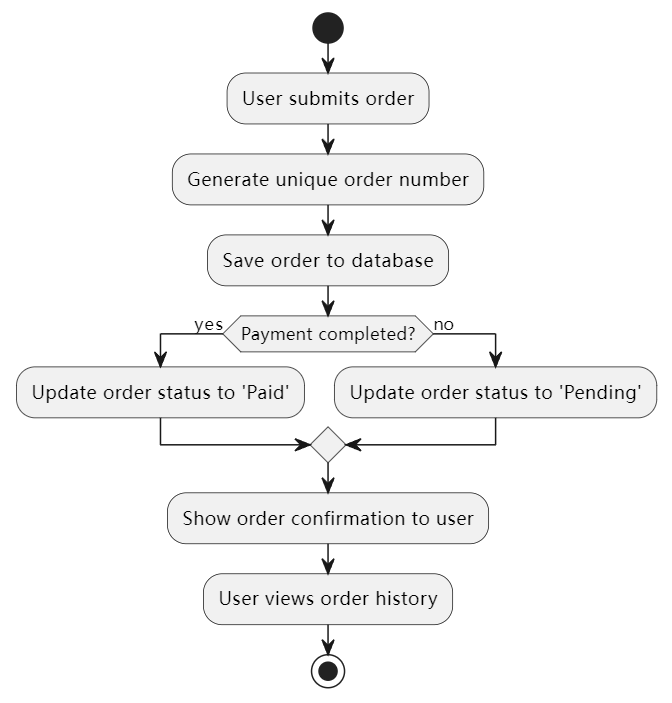
### 4.4 订单管理模块

主要功能：

- 用户可以提交订单并生成唯一订单号。

- 系统支持订单状态的更新（如已支付、待支付等）。

- 用户可以查看历史订单。

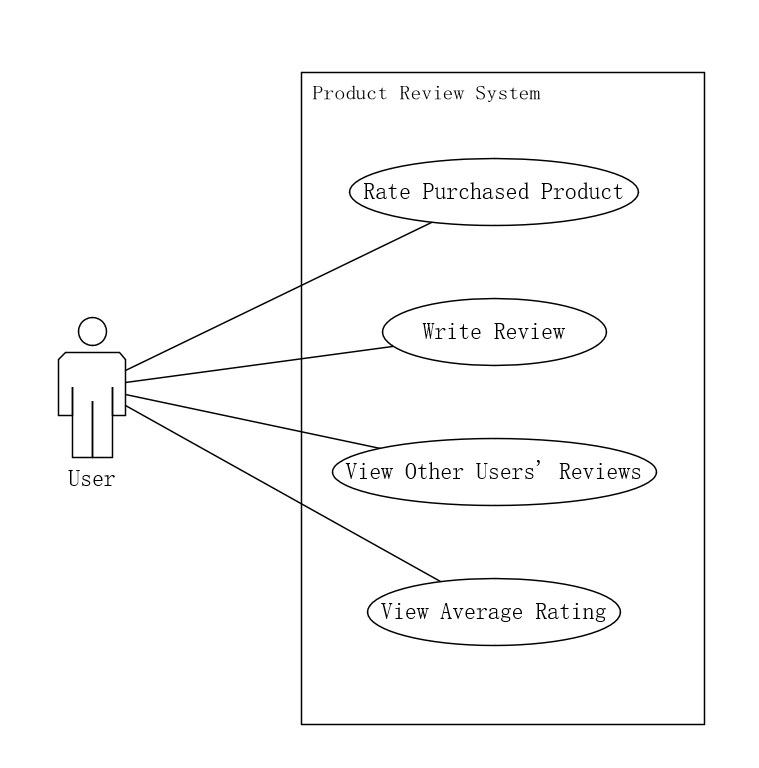


### 4.5 商品评价模块

主要功能：

- 用户可以评价已购买的商品并查看其他用户的评论。

- 支持评分和文字评论，系统展示平均评分。



## 5. 接口设计

### 5.1 RESTful API 设计

系统的前后端通过 RESTful API 进行通信，以下为主要 API 接口设计。

| 接口 | 请求方式 | URL | 描述 |

|---------------------|-----------|----------------------|--------------------------------|

| 用户注册 | POST | /api/register | 注册新用户 |

| 用户登录 | POST | /api/login | 用户登录，返回JWT |

| 商品列表 | GET | /api/products | 获取商品列表 |

| 商品详情 | GET | /api/products/{id} | 获取某个商品的详细信息 |

| 添加商品到购物篮 | POST | /api/cart | 将商品添加到购物篮 |

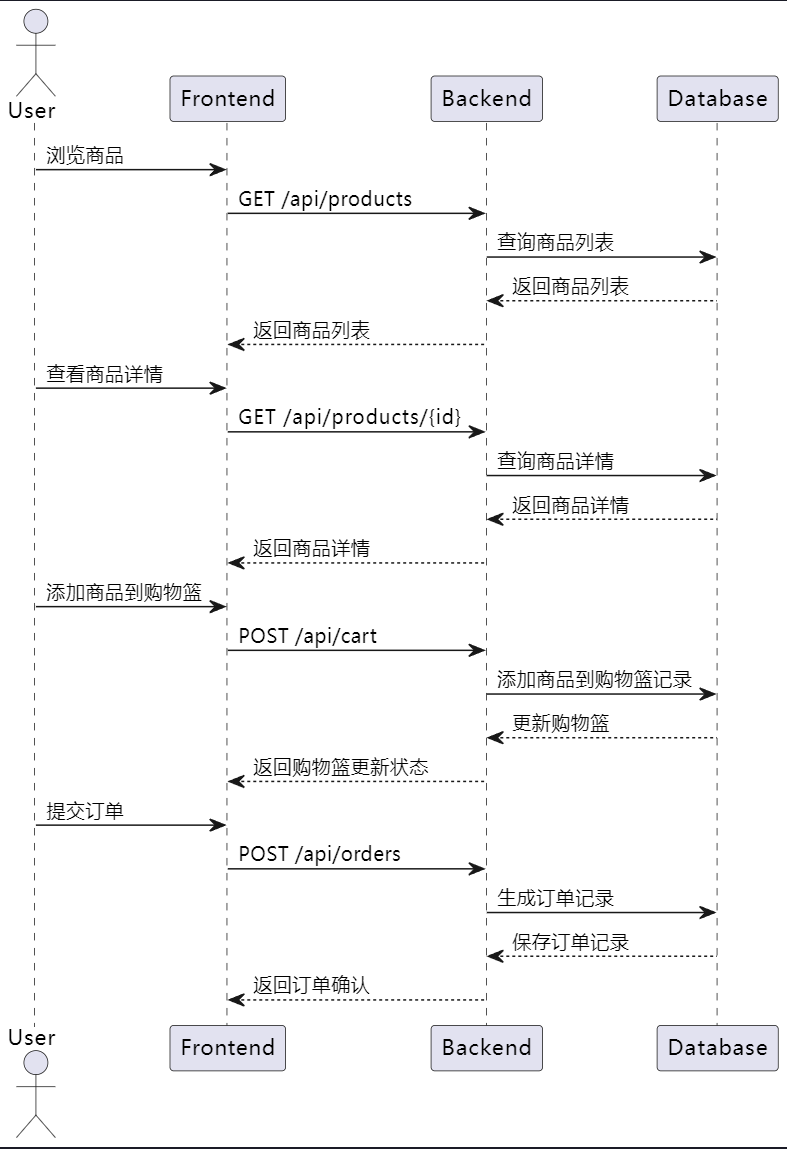
| 提交订单 | POST | /api/orders | 提交订单，生成订单记录 |

| 查看历史订单 | GET | /api/orders | 获取用户的历史订单 |

| 商品评价 | POST | /api/reviews | 提交商品评价 |

| 查看商品评价 | GET | /api/products/{id}/reviews | 获取某商品的评价 |

### 5.2 API 调用顺序



## 6. 流程设计

### 6.1 用户下单流程

1. 用户登录系统。

2. 浏览商品并选择需要购买的乐器。

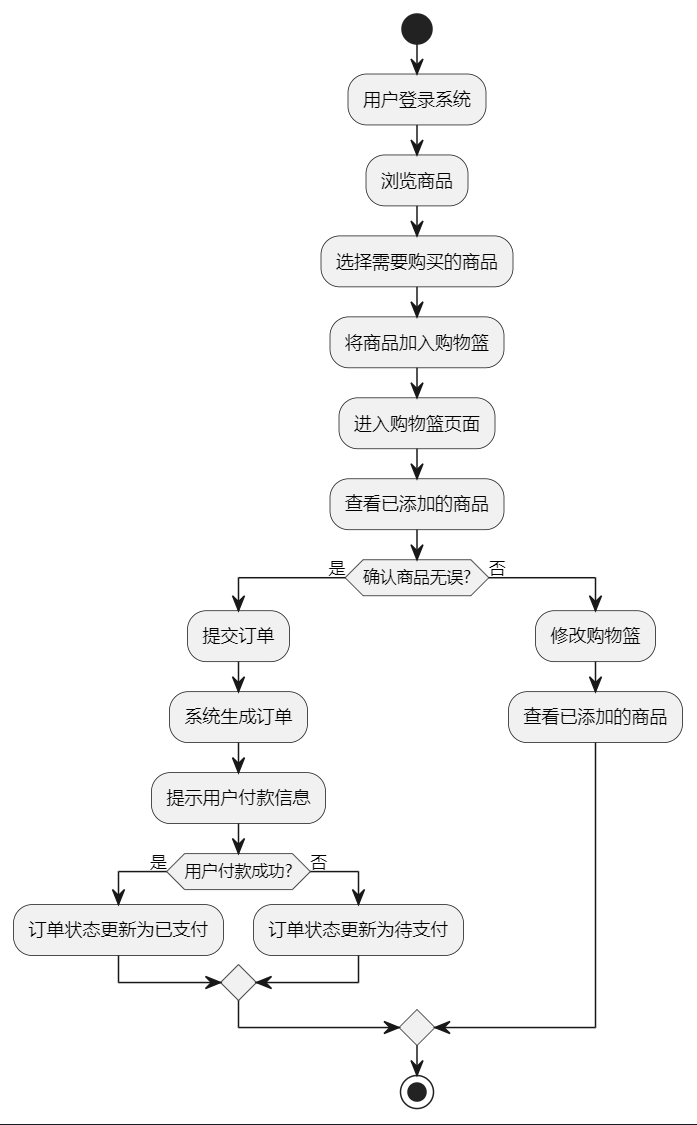
3. 将乐器加入购物篮。

4. 进入购物篮页面，查看已添加的商品。

5. 确认无误后，提交订单。

6. 系统生成订单，并提示用户付款信息。

7. 用户支付成功后，订单状态更新为已支付。

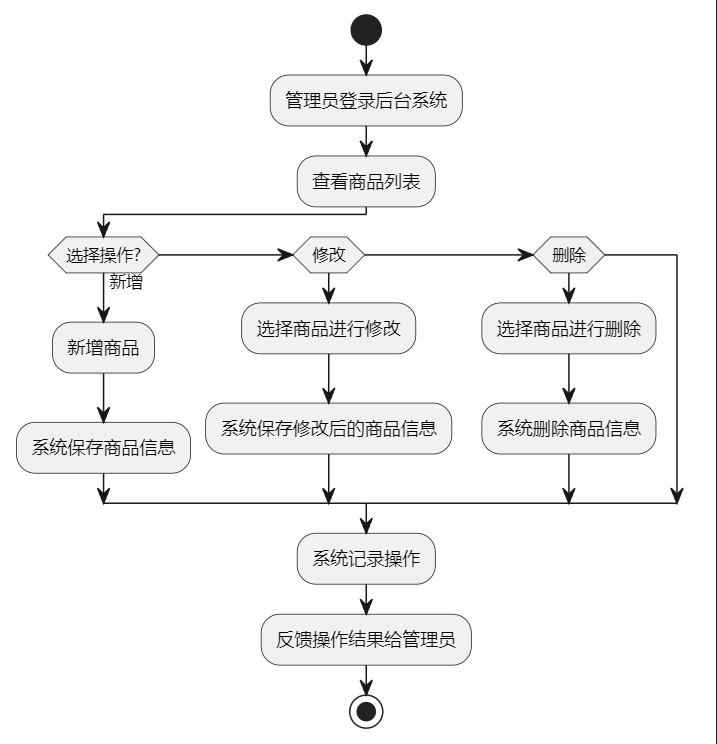


### 6.2 管理员商品管理流程

1. 管理员登录后台。

2. 管理员可以新增、修改或删除商品信息。

3. 系统将对用户的操作进行记录并反馈结果。



## 7. 安全设计

### 7.1 身份认证

- 使用JWT进行用户认证，用户在登录成功后会获得一个JWT令牌，随后每次请求都携带该令牌进行身份验证。

### 7.2 数据加密

- 用户密码在存储时进行加密（如使用bcrypt）。

- 敏感数据传输时使用SSL加密。

### 7.3 安全防护

- 防止常见安全攻击，如SQL注入、跨站脚本攻击（XSS）和跨站请求伪造（CSRF）。

## 8. 性能优化

### 8.1 数据库优化

- 采用数据库索引加速查询。

- 对频繁访问的数据进行缓存。

### 8.2 后端优化

- 实现API分页，避免一次性返回大量数据。

- 使用异步请求处理，提高服务器并发性能。

### 8.3 前端优化

- 通过按需加载减少初次加载时间。

- 使用浏览器缓存加速静态资源加载。

## 9. 测试计划

### 9.1 单元测试

- 对每个功能模块的核心逻辑进行单元测试，确保输入输出的正确性。

### 9.2 集成测试

- 测试前后端的交互是否正确，特别是REST API的响应和处理。

### 9.3 性能测试

- 使用负载测试工具（如JMeter）模拟高并发场景，确保系统在1000个并发用户下能平稳运行。

### 9.4 安全测试

- 进行漏洞扫描，确保系统安全无风险。

## 10. 部署设计

### 10.1 部署架构

- 采用云服务平台（如AWS或Azure）进行部署。

- 使用负载均衡器分配流量，确保高可用性。

- 实现自动化部署流程，使用CI/CD工具（如Jenkins或GitHub Actions）进行自动化构建与发布。

### 10.2 日志与监控

- 设置日志系统（如ELK Stack），记录系统运行状态与用户行为。

- 通过监控工具（如Prometheus、Grafana）进行实时监控，确保系统性能。