软件架构文档(SAD)

目录

HYPERLINK \l "\_Toc198285991"软件架构文档(SAD) PAGEREF \_Toc198285991 \h1

HYPERLINK \l "\_Toc198285992"1引言 PAGEREF \_Toc198285992 \h2

HYPERLINK \l "\_Toc198285993"1.1标识 PAGEREF \_Toc198285993 \h2

HYPERLINK \l "\_Toc198285994"1.2系统概述 PAGEREF \_Toc198285994 \h2

HYPERLINK \l "\_Toc198285995"1.3文档概述 PAGEREF \_Toc198285995 \h2

HYPERLINK \l "\_Toc198285996"1.4基线 PAGEREF \_Toc198285996 \h2

HYPERLINK \l "\_Toc198285997"2引用文件 PAGEREF \_Toc198285997 \h2

HYPERLINK \l "\_Toc198285998"3 CSCI级设计决策 PAGEREF \_Toc198285998 \h3

HYPERLINK \l "\_Toc198285999"4 CSCI体系结构设计 PAGEREF \_Toc198285999 \h4

HYPERLINK \l "\_Toc198286000"4.1体系结构 PAGEREF \_Toc198286000 \h4

HYPERLINK \l "\_Toc198286001"4.1.1程序(模块)划分 PAGEREF \_Toc198286001 \h4

HYPERLINK \l "\_Toc198286002"4.1.2程序(模块)层次结构关系 PAGEREF \_Toc198286002 \h4

HYPERLINK \l "\_Toc198286003"4.2全局数据结构说明 PAGEREF \_Toc198286003 \h5

HYPERLINK \l "\_Toc198286004"4.2.1常量 PAGEREF \_Toc198286004 \h5

HYPERLINK \l "\_Toc198286005"4.2.2变量 PAGEREF \_Toc198286005 \h5

HYPERLINK \l "\_Toc198286006"4.2.3数据结构 PAGEREF \_Toc198286006 \h5

HYPERLINK \l "\_Toc198286007"4.3 CSCI部件 PAGEREF \_Toc198286007 \h5

HYPERLINK \l "\_Toc198286008"4.4执行概念 PAGEREF \_Toc198286008 \h6

HYPERLINK \l "\_Toc198286009"4.5接口设计 PAGEREF \_Toc198286009 \h6

HYPERLINK \l "\_Toc198286010"5 CSCI详细设计 PAGEREF \_Toc198286010 \h7

HYPERLINK \l "\_Toc198286011"6需求的可追踪性 PAGEREF \_Toc198286011 \h7

HYPERLINK \l "\_Toc198286012"7注解 PAGEREF \_Toc198286012 \h8

HYPERLINK \l "\_Toc198286013"附录 PAGEREF \_Toc198286013 \h8

# 1引言

## 1.1标识

系统名称：网上书店系统（Alpha）

文档：软件架构文档（SAD）

版本号：V1.0.0

缩略词： SAD（软件架构文档）

## 1.2系统概述

系统基于 Flask + MySQL 后端与 HTML/CSS/JavaScript 前端，实现图书展示、购物车、在线支付、电子书阅读、订单管理和后台运维等功能，满足高并发、数据安全及可扩展性要求。系统部署于 Linux 环境，支持云端弹性伸缩与 CDN 加速。

## 1.3文档概述

本条应概述本文档的用途与内容，并描述与其使用有关的保密性或私密性要求。本说明书按照《软件(结构)设计说明》(SDD)的格式组织，覆盖全局架构设计、模块划分、数据结构、接口和详细设计等内容。章节结构包括：引言（标识、系统概述、文档用途、基线说明）、引用文件、CSCI级设计决策、CSCI体系结构设计、CSCI详细设计、需求可追踪性、注解及附录。文档概述明确本文档目的在于为开发、测试和后期维护提供架构基线，并指明架构决策的依据和理由。文档内容适用于项目团队内部交流，包含必要的保密性说明（如适用）并标明版本基线、审批状态等信息。

## 1.4基线

本架构说明所依据的设计基线为需求规格说明书 (SRS) V1.0和项目需求变更会议决议。当前版本为初稿基线，后续版本将根据设计评审和实现情况更新。所有后续变更将通过配置管理流程跟踪记录，更新基线号和版本。

# 2引用文件

本章列出本文档引用的主要文件及标准，包括但不限于：

* 《软件需求规格说明书 (SRS) V1.0》，2025年3月。
* ISO/IEC/IEEE 42010:2011 《系统和软件工程——架构描述》标准。
* 国标《13 - 软件(结构)设计说明(SDD)》
* Flask官方文档和相关技术资料（例如Flask用户指南）。
* MySQL参考文档（Oracle官方手册）。
* 项目管理和配置管理规范文档。
* 其他关联文档，如用户界面设计说明、接口协议文档等。

# 3 CSCI级设计决策

本章节列出系统级（CSCI级）的主要设计决策，并说明采用该决策的理由（架构决策原理）。CSCI（Computer Software Configuration Item）在本说明中即指“线上书店系统”整体。每项决策从用户需求和项目目标出发，从外部行为角度描述系统所做的关键选择。

* **架构风格**：采用典型的**Web客户端-服务器架构**。前端与后端通过HTTP/HTTPS协议交互，前端实现基于浏览器的用户界面，后端提供RESTful API服务。采用这种分层架构有利于分工清晰和系统可扩展。
* **使用Flask框架**：选用Flask作为后端开发框架。Flask是一个轻量的Python Web框架，具有灵活性高、易于扩展和丰富社区资源等特点。它支持快速开发和插件扩展，适合构建中小型Web应用。
* **MVC 设计模式**：在后端逻辑上采用MVC（模型-视图-控制器），将数据模型、业务逻辑和界面分离。例如：创建Book等模型类（Model）、视图函数(View)处理HTTP请求、控制层路由协调两者。这样便于维护和测试。
* **RESTful接口**：系统对外提供REST风格的API接口，包括GET/POST/PUT/DELETE等方法。例如，通/books接口获取书籍数据，通过/cart管理购物车。采用RESTful设计使得前后端解耦，未来可拓展移动客户端或第三方调用。
* **持久化存储**：选择MySQL作为关系型数据库管理系统。MySQL是开放源代码的RDBMS，性能优良且易于使用，广泛用于Web应用。其事务支持和查询优化能够满足本系统对数据一致性和并发访问的要求。
* **数据缓存与索引**：对于读操作频繁的书籍信息，可使用数据库索引来提高响应性能。具体缓存策略由性能评测决定。
* **安全机制**：决策包括：通过HTTPS加密传输保护数据安全；用户登录采用基于令牌（Token）或会话（Session）认证；敏感信息（如用户密码）在后端以哈希方式存储；使用防跨站请求伪造（CSRF）令牌保护表单提交；限制接口访问权限（用户和管理员角色隔离）。这些措施满足关键的安全性和隐私性需求。
* **性能与可用性**：系统需要支持并发用户访问，响应延迟应符合用户体验要求（例如页面加载<2秒）。为此，可在架构层面采用异步处理、数据库连接池等技术，并设置监控报警保证系统可用性。
* **可维护性与可扩展性**：模块化设计，每个功能模块独立，如用户模块、商品模块、订单模块等，降低耦合度，方便后期维护和升级。采用ORM框架管理数据模型，提高开发效率。同时，代码托管使用版本控制，遵循编码规范，方便团队协作。

以上设计决策覆盖系统输入/输出、行为响应、数据库策略、安全策略和可维护性等方面，是满足需求和质量目标的关键选择。各决策的详细背景和权衡（如备选方案和性能比较）在本说明后续章节和相应设计记录中加以说明。

# 4 CSCI体系结构设计

本章应分条描述CSCI体系结构设计。如果设计的部分或全部依赖于系统状态或方式，则应指出这种依赖性。如果设计信息在多条中出现，则可只描述一次，而在其他条引用。应给出或引用为理解这些设计所需的设计约定。

## 4.1体系结构

### 4.1.1程序(模块)划分

系统软件配置项内部划分为多个功能模块，每个模块负责特定功能并封装相应业务逻辑和数据。主要模块包括：

* **用户模块(User Module)**：负责用户注册、登录、身份验证、个人信息管理等。
* **商品模块(Book Module)**：管理图书数据，包括图书信息的增删改查、分类查询和搜索功能。
* **购物车模块(Cart Module)**：维护用户的购物车状态，支持添加、删除商品和修改数量。
* **订单模块(Order Module)**：处理用户下单和订单管理，生成订单记录、订单状态跟踪、订单历史查询等。
* **支付模块(Payment Module)**：与第三方支付网关接口，处理在线支付事务，保证交易安全性。
* **库存管理/管理员模块(Admin Module)**：供商家或管理员使用的后台，包括商品库存管理、订单审核、数据统计等功能。
* **公共组件(Common Services)**：包括数据库访问层(DAO/ORM)、缓存服务、安全认证服务、日志和异常处理模块等。

以上每个模块在实现上由若干类/函数组成。例如“商品模块”包含Book实体类、BookService业务类、BookController视图函数等；各模块对应的源文件按功能组织并使用一致的命名和编码约定。

### 4.1.2程序(模块)层次结构关系

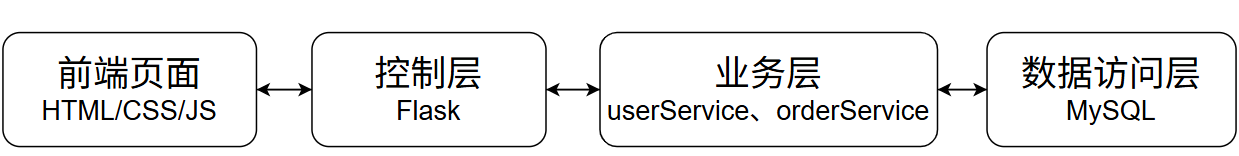
系统遵循分层调用关系：

* **表示层(Front-End)**：浏览器中的HTML/CSS/JS界面；与后端通过REST API交互，不直接访问数据库。
* **控制层(Controller)**：Flask路由和视图函数，接收前端请求，调用业务层逻辑。
* **业务层(Services)**：包含各模块的业务逻辑类（例如UserService、OrderService等），封装具体功能的实现。
* **数据访问层(DAO/ORM)**：封装数据库操作，通过ORM映射或SQL语句读写MySQL数据库中的表。

各模块之间通过接口交互。常见层次关系示例如下：

* **客户发起请求** → 前端路由到**控制层** (Flask视图) → **业务层** (处理业务逻辑) → **数据访问层** (与数据库交互)。
* 控制层还负责调用**认证服务**、**缓存服务**等公共组件。
* 不同模块之间部分逻辑相互依赖，如购物车模块可能需要调用商品模块验证商品库存，订单模块调用购物车确认商品列表。

系统的模块结构如图所示（可使用类图或组件图说明模块之间的调用/层次关系）。



## 4.2全局数据结构说明

系统使用的全局数据结构包括数据库表和配置常量等。

### 4.2.1常量

* 数据库连接配置：主机名、端口、用户名、密码等（配置在配置文件中）。
* 系统常量：如购物车最大数量、订单失效时间等。

### 4.2.2变量

* 全局变量：在应用初始化时加载的配置参数（例如缓存大小、分页大小等）。
* 运行时全局状态：如登录会话超时时间、当前活跃用户数等，可通过监控动态更新。

### 4.2.3数据结构

主要数据结构体定义如下：

* **用户表(User)**：字段包括用户ID、用户名、邮箱、密码哈希、权限角色等。
* **图书表(Book)**：字段包括图书ID、标题、作者、价格、库存数量、分类等。
* **购物车表(Cart)**：每条记录关联用户ID和图书ID以及数量。
* **订单表(Order)**：字段包括订单ID、用户ID、订单状态、总金额、时间戳等；以及订单项表关联订单ID和图书ID。
* **支付记录表(Payment)**：存储支付交易信息，如支付ID、订单ID、支付金额、支付时间、第三方交易号等。

每个表字段及约束（主键、外键、索引）在数据库设计说明中详细描述，此处仅概述关键结构。

## 4.3 CSCI部件

本CSCI包含以上各功能模块，每个模块可视为一个软件配置项。各配置项静态关系（如组合关系）和分配情况如下：

* 所有模块共同构成系统整体，彼此通过软件接口交互。各模块的用途与分配的需求、设计决策在模块说明中列出。
* 模块间关系示例：用户模块与购物车模块配合完成认证与购物功能；商品模块为订单和购物车模块提供数据。
* 每个模块声明开发状态：本系统为新开发项目，所有模块均为新开发软件配置项；未来若有重用组件（如现成的OAuth认证库），则列为重用软件配置项并注明来源。
* 硬件资源：本系统运行在标准Web服务器和数据库服务器上，所需资源包括处理器、内存和存储等；其中MySQL数据库根据预估数据量配置磁盘及内存。资源分配根据性能测试进行调优（参见4.4）。
* 软件配置项对应的库文件和部署位置将记录于配置管理工具。

## 4.4执行概念

本节描述系统运行时组件交互的动态行为。典型场景：用户在浏览器搜索图书时，前端发送查询请求到后端商品模块；后端调用数据库检索数据，将结果以JSON返回前端，前端渲染显示图书列表。下单场景：用户提交订单请求，后端订单模块验证库存、扣减库存、生成订单并返回确认；随后支付模块与第三方接口交互完成支付，并更新订单状态。

执行概念可用时序图或交互图表示组件之间的控制流和数据流。例如：

* **用户登录**：前端提交用户名/密码 → 用户模块验证 → 成功后生成会话令牌 → 返回前端。
* **下单流程**：前端调用购物车模块准备订单 → 前端提交订单请求到订单模块 → 订单模块调用数据库写入订单和订单项、调用支付模块引导支付 → 支付完成回调更新订单状态。

在动态过程中，要考虑并发控制（如并发订单防止超卖）、异常处理（支付失败回滚）、事务处理（保证一致性）等。流程中的关键步骤、交互顺序、并发锁定和超时机制均在详细设计中具体描述。

## 4.5接口设计

本节描述系统内部模块间及与外部实体的接口。采用RESTful接口标准，主要接口如下：

* **用户界面 (UI)**：浏览器客户端与后端服务的接口，通过HTTP(S)进行通信。所有前端请求通过AJAX或表单提交到后端相应的API端点。
* **内部服务接口**：模块内部调用接口，如订单模块调用商品模块的库存查询接口。接口采用函数调用或HTTP请求视部署方式而定。
* **数据库接口**：系统使用ORM或SQL直接与MySQL数据库交互，实现数据的读写操作。
* **第三方接口**：支付模块使用外部支付网关API（HTTPS/REST）进行交易；如短信通知使用第三方短信API。
* **外部系统**：例如提供图书元数据的外部服务（可选）或ERP系统对接接口。

每个接口应定义访问方式、请求格式（URL、方法、参数）、返回格式（JSON结构或HTTP状态）、安全要求（认证方式）、错误处理机制等。接口设计图和接口标识符将用于说明模块间调用关系。

# 5 CSCI详细设计

本章详细描述每个软件配置项（主要模块）的设计，包含关键算法、数据结构和模块内部结构。若设计依赖于系统状态，在相应条中指出。

* **5.1 用户模块**：包括User类，login等服务。介绍注册、登录流程的实现，以及用户个人中心的实现。数据结构：用户表模式定义，密码加盐哈希算法。对于用户信息，设计多个uer相关类，比如uer中保存用户的账号权限密码昵称等，usercollect存储book的id信息等。注册以及登录过程中使用会话管理策略（Flask-Login），注册过程和一般注册功能相似，用户提供一般的信息数据，接收后传递到数据库中完成注册。登录流程类似，用户应提供数据库中已有的账号数据，然后经过验证后，完成登录。对于用户的个人信息修改，以及收藏商品消息，填写的用户地址，有个人中心的信息展示，用户可以通过用户的个人中心完成部分信息的管理。
* **5.2 商品模块**：包括Book类，搜索等服务。前端展示大致书籍信息，包括书名、价格，点击获取书籍详细信息，包括作者、出版社、库存数量、内容简介等。书籍搜索使用SQLAlchemy进行过滤查询，如有关键字则按书名模糊匹配。
* **5.3 购物车模块**：包括订单查看、订单修改、订单删除、购物车更新、购物车结算等服务。能够直观查看已加入购物车的订单信息，包括订单的书名、书籍单价、书籍数量、订单总价等；支持修改订单中书籍的数量，并更新订单总价；支持删除目标订单；能够统计购物车中所有订单的总价格；提供购物车结算。
* **5.4 订单模块**：涉及订单生成流程。包括订单表结构设计、订单状态枚举、下单时与商品库存同步逻辑。对于订单，设计了orderform和orderdetails两个类来管理顶点信息，存储订单的涉及的书的数目（number），以及订单涉及的价格（orderdetails.price）以及需要运送的地址（orderform.address）。说明事务处理策略，对于用户在购物车信息确认，支付成功之后，对于商品的库存容量进行削减，然后对外展示，防止在商品内容不足时完成的订单信息问题。
* **5.5 支付模块**：支付模块目前只有模拟支付功能，详细协议部分没有实际作用。
* **5.6 管理员模块**：包括商品库存管理和订单审核功能。说明后台接口设计、权限校验等。权限管理在设计账号的时候已经设定，由后台操作账号权限，给予特定账号权限，实现管理员账号功能。
* **5.7 公共组件**：介绍关键通用组件设计，如缓存服务、日志和异常处理模块。缓存服务由session提供，在网页刷新时，仍有部分数据维持原状，例如登录状态，为方便使用，在未关闭全部浏览器页面时，保存登录状态，避免多次登录。日志和异常处理模块，设计统一的报错提示函数，在不同情况下返回不同的数值，展示在弹窗上，实现报错信息提示。

# 6需求的可追踪性

本节阐明系统需求与架构元素之间的对应关系（追踪映射），确保架构设计满足所有功能和质量需求。可采用表格或列表形式逐一列举主要需求及其实现模块。例如：

* **需求 R1：用户可以浏览图书目录** → 实现于商品模块的检索接口和前端页面。
* **需求 R2：用户可以下单购买图书** → 实现于购物车模块和订单模块（前端发起下单请求，后端生成订单并更新库存）。
* **需求 R3：系统支持用户注册和登录** → 实现于用户模块的注册/认证流程。
* **性能需求 R4：系统能支持至少1000名并发用户** → 架构设计通过REST接口与数据库连接池、高性能查询优化满足，并规划压力测试评估。
* **安全性需求 R5：用户密码必须加密存储** → 用户模块采用哈希+盐算法存储密码。
* **可维护性需求 R6：系统模块化、代码易于扩展** → 通过模块化设计和使用MVC模式满足，每个功能模块独立部署。

# 7注解

本文又开发组全体成员共同完成。