



江苏师范大学

JIANGSU NORMAL UNIVERSITY

本科毕业设计

UNDERGRADUATE DESIGN

设计题目： 基于 SSM 的校园众包递送系统的设计与实现

软件设计

姓名： 李国瑞

学院： 智慧教育学院

专业： 软件工程

年级、学号： 20 智 72 209007039

指导教师： 李小斌

江苏师范大学教务处印制

基于 SSM 的校园众包递送系统的设计与实现

摘 要

本文是关于一种基于 SSM 的校园众包递送系统的设计与实现的论文。SSM 框架是一种常用的 JavaWeb 开发框架，包括 SpringMVC、Spring 和 MyBatis-Plus 三个组件，具有良好的性能和广泛的应用场景。本文旨在设计和实现一种基于 SSM 的校园众包递送系统，以解决校园快递递送中存在的信息传递不畅、管理混乱等问题。首先，本文介绍了校园快递递送的现状和存在的问题，包括信息传递不畅、递送效率低下、安全问题等。然后，本文对 SSM 框架进行了介绍和比较，并分析了采用 SSM 框架进行系统设计的可行性。接着，本文详细介绍了系统的功能和模块，并针对每个模块进行了详细的设计和实现。具体来说，该系统包括快递信息收集、快递信息管理和快递信息传递三个模块。其中，快递信息收集模块实现了快递信息的收集和入库；快递信息模块实现了快递信息的查询、修改和删除；快递信息传递模块实现了快递信息的发布、更新和删除。最后，本文对系统的性能和安全性进行了评估，并提出了改进建议。通过对系统的性能测试发现，该系统的响应速度较快，能够在短时间内完成大量的快递信息处理。同时，通过对系统的安全性测试发现，该系统具有良好的安全性，避免了用户的个人信息被泄露等问题。综上所述，本文设计和实现了一种基于 SSM 的校园众包递送系统，该系统具有良好的性能和安全性，能够为校园快递递送提供更加便捷和高效的服务。

关键词： 众包递送 SSM 框架 SpringBoot Vue3

Design and Implementation of a Campus Volunteer Delivery System Based on SSM Framework

Abstract

This paper is about the design and implementation of a campus volunteer delivery system based on the SSM framework. The SSM framework is a commonly used JavaWeb development framework that includes three components: SpringMVC, Spring, and MyBatis-Plus. It has good performance and a wide range of applications. The aim of this paper is to design and implement a campus volunteer delivery system to solve the problems of information transmission, management chaos, and security issues in campus express delivery. Firstly, this paper introduces and compares the current situation and problems of campus express delivery, including information transmission difficulties, delivery efficiency problems, and security issues. Then, this paper introduces and analyzes the feasibility of using the SSM framework for system design. Next, this paper comprehensively describes the functions and modules of the system, and designs and implements each module in detail. Specifically, the system includes three modules: express information collection, express information management, and express information transmission. The express information collection module realizes the collection and input of express information; the express information management module realizes the search, modification, and deletion of express information; and the express information transmission module realizes the publication, update, and deletion of express information. Finally, this paper assesses the performance and security of the system and proposes improvement suggestions. Through the performance testing of the system, this paper discovers that the system has a fast response speed and can complete large-scale express information processing in a short time. At the same time, through the security testing of the system, this paper discovers that the system has good security, avoiding issues such as the leakage of user information. In summary, this paper designs and implements a campus volunteer delivery system based on the SSM framework, and the system has good performance and security, providing more convenient and efficient services for campus express delivery.

Keywords: Delivery System SSM Pattern Design SpringBoot Vue3.

目 录

摘 要	I
Design and Implementation of a Campus Volunteer	II
Delivery System Based on SSM Framework	II
Abstract	II
1 绪论	1
1.1 研究背景	1
1.2 研究目的及意义	1
1.3 课题研究现状	2
2 开发工具介绍	4
3 可行性分析与需求分析	5
3.1 校园众包递送系统现状分析	5
3.2 校园众包递送系统功能需求分析	错误！未定义书签。
3.3 校园众包递送系统软件的技术可行性分析	5
4 功能模块设计	7
4.1 UML 软件建模	7
4.2 数据库设计	9
4.3 功能设计	11
5 前端演示及后端实现原理	14
5.1 登录	23
5.2 注册界面	23
5.3 派单界面	24
5.4 抢单界面	25
5.5 订单评价界面	错误！未定义书签。
6 软件测试	27
7 总结	37
参考文献	38

1 绪论

1.1 研究背景

随着互联网和移动设备的普及，众包 (crowd-sourcing) 作为一种新型的生产方式，逐渐受到了广泛关注。校园众包作为一种典型的众包形式，是指利用校园资源，通过校园网络平台将任务分配给学生团队或个人，以完成各种校园任务。目前，随着互联网技术的不断发展，校园众包已经逐渐成为一种有效的校园服务模式。然而，传统的校园众包存在诸多问题，例如任务完成质量不稳定、任务分配不均等。针对这些问题，基于 SSM 的校园众包递送系统应运而生。

SSM 框架是一种常用的 JavaWeb 框架，其强大的功能和易用性受到了广泛的认可。基于 SSM 的校园众包递送系统，将 SSM 框架的技术与校园众包相结合，利用 Java 语言和 MySQL 数据库实现任务发布、任务执行、任务反馈等功能。该系统可以实现任务的分类、任务的发布、任务的分配、任务的执行、任务的反馈等功能，为校园众包提供有效的技术支持，有助于提高校园众包的效率和服务质量。

综上所述，基于 SSM 的校园众包递送系统的设计与实现是一项有意义的研究工作，其可以有效地解决传统校园众包存在的问题，为校园众包提供有效的技术支持，有助于提高校园众包的效率和服务质量。

1.2 研究目的及意义

中国互联网络信息中心所发布的《第 39 次中国互联网络发展状况统计报告》指出，2016 年我国网络购物成员总数达到 4.69 亿，占网民总数的 64.1%，而大学生群体是其中最大消费群体，占比为 25%。网购数量的上涨使得流向高校的快递数量激增，形成特点鲜明的高校快递配送模式。但是，在我国物流配送资源“散”、监管体系“乱”、服务能力“弱”的大环境下^[1]，高校快递配送同样也存在配送混乱、低效等问题，同时取货模式较为单一，多为校门口分散式取货，部分高校采取建立自提柜和统一配送中心的模式。通过对当前高校快递配送模式进行分析，提出构建“信息匹配平台”，优化校园快递分拣过程，向在校师生提供“门到门”式的增值服务。同时，基于众包物流的思想，把握高校人员密集、闲置劳动力充足的特点，将校园内的被动式物流转化为众人参与的主动式物流，提升高校快递的配送效率，并通过增加增值服务在高校快递配送模式中融入新的商流，形成具有商业附加值的高校递送模式。

为改善高校快递配送环境，优化高校快递配送效率，在对高校快递配送模式现状

及存在问题进行分析的基础上,针对高校快递配送模式中存在的杂乱无序、配送渠道过少、无“门到门”服务等问题,结合高校快递配送量巨大、配送时间集中、流向点状集聚、包裹轻便小巧居多的特点,将众包模式引入高校快递配送,提出高校快递众包递送的新思路。

1.3 课题研究现状

众包一词最早起源互联网,由 Jeff Howe 首次提出。国内外学者对众包展开了深入研究,黎娟^[2]认为众包是一种基于网络的新型合作模式;Saxton 等^[3]则认为众包可以理解为,组织或个人利用先进的互联网技术,将传统上交由雇员或承包商所做的工作以公开征集的方式外包给非特定的分布式网络大众来完成的模式。利用众包形成的商业模式中,参与方一共有 3 种类型,分别是发包方(需求提供者)、中介(众包平台)、接包方(众包参与者)。

随着众包的发展,该商业模式不仅帮助互联网公司开拓创新思维,也延伸到其他行业领域,国内很多 O2O 及电子商务企业正在进行众包物流的尝试。国内很多学者对众包物流进行了研究,朱云桦等^[4]通过对城市配送现状的分析,将众包物流与传统物流进行对比,认为众包物流的模式是将原本由专业物流企业负责的城市末端配送,转交给企业之外的民众群体来完成;周金华^[5]从生鲜电商行业的角度,比较研究分析众包物流模式下生鲜电商行业发展的优势和逆势,进而提出基于众包物流模式下生鲜电商发展的可行性建议与对策;纪汉霖^[6]从众包物流的角度分析生鲜 O2O 的配送模式,解决目前生鲜配送中存在的许多难点和问题;任为^[7]提出互联网思维对电子商务发展的影响,认为众包模式可以作为解决“最后一公里”的新方法,提出新时代下城市配送的发展对策。综上所述,众包与物流的结合为城市“最后一公里”配送提供了一种全新的思路,为此,基于众包模式,并结合高校快递配送资源“散”、监管体系“乱”、服务能力“弱”的大环境下^[7]提出高校快递众包模式,达到减少快递寻找时间、取货时间灵活、减少快递摆放杂乱的现象^[8]。

高校快递采用众包模式是指利用高校内的学生为主要配送人力资源,实施高校内的快递配送。利用移动互联技术搭建信息化平台在现代物流中必不可少^[9],因而高校快递众包模式的建立实施,关键在于建立相应的众包资源交互平台。通过平台聚集高校内的配送资源,经过平台认证的高校内的人员都可以成为快递员。平台主要的功能分为 2 个部分,一为普通快递的配送,发包方(校内人员)在平台上可以提出取货需求,接包方(高

校内的众包快递员)根据需求进行快递的配送, 并收取 1~2 元的激励服务费用, 这种显性激励^[10]是普通快递配送的重要促进措施保障; 二为进行校园超市或者校外商品的代购、代送服务, 发包方还可以在平台上提出购买物品的需求, 接包方也可以根据自身条件选择性接单, 完成代购任务, 并按照商品的重量、配送距离获得报酬。平台结算采用线上电子支付的方式。代购、代买服务带动高校内的商流活动, 促使更多的人加入众包平台成为发包方和接包方, 增加物流配送的流通渠道, 以商流的活动带动高校内的快递配送活动。

在时空众包环境下, 任务分配^[11]依然是其核心问题之一。针对这一问题学术界展开了积极研究, Boutsis 等^[12]首次提出了动态场景下的三类对象的在线任务分配问题, 面向众包任务、众包工人以及任务执行地点的合理匹配。可以看出目前研究更多的是三类对象条件下的问题^[13]。刘辉等^[14]提出了一种能够对任务分配的当前情况自动调整阈值的算法, 但分配总效用不高; Hassan 等^[15]提出一种基于统计预测的自适应阈值算法, 进一步提高了分配总效用。不难看出, 自适应阈值的应用能使问题解决方案得到优化。

众包在高校快递配送平台适用性主要体现在以下方面: ①高校快递的货物流向集中、货品轻便, 学生可以在收取快递的同时代收快递或帮助配送快递; ②众包模式的应用能够满足“门到门”式配送, 减少快递堆积, 改善校园内外环境; ③利用众包在校园内进行快递派送, 有着广泛的配送人员数量基础, 相对封闭的区域范围也保证了配送的准确性和时效性, 学生为主体的配送队伍降低了众包配送中丢件、偷件的发生概率。然而, 对于涵盖自适应思想的高校快递配送平台尚缺乏深入的研究。

2 开发工具介绍

Idea 是一款由 JetBrains 公司开发的 Java 集成开发环境。它拥有完善的 Java 开发环境，支持 Java、Kotlin 等语言开发，提供了各种智能化的编码工具、快速代码重构等功能，被广泛应用于 Java 开发领域。WebStorm 是一款由 JetBrains 公司开发的 JavaScript 开发环境。它是一款专门为 Web 开发人员设计的 IDE，支持 JavaScript、HTML、CSS 等前端开发语言，拥有智能化的代码编辑工具、自动化测试、代码重构等特性，可提高 Web 开发效率。Axios 是一个用于浏览器和 node.js 的基于 Promise 的 HTTP 客户端，用于可以发送异步请求。它可以减少代码的重复程度，提供了许多常用的 HTTP 方法如 GET、POST、PUT、DELETE 等，并支持 Promise API，可以非常便捷的进行数据交互。SpringBoot 是由 Pivotal 团队（Spring Framework 的核心开发者）所创建的。它是一个基于 Spring 框架的快速开发 web 应用的微服务框架。SpringBoot 支持很多开发框架及协议，如 MVC、REST、JPA 和 WebSocket 等，是一个集成度很高的框架。Vue 是一个用于构建用户界面的渐进式框架，也被称为 MVVM 模式的前端 Web 框架。Vue3 是 Vue.js 的最新版本，强调了性能的提升、开发体验的改进和代码体积的优化，并增加了 Composition API 等新的 API，使开发者更加容易进行模块化开发和组合多个组件。Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境，可用于开发服务器端应用程序。它使用事件驱动、非阻塞 I/O 模型等特点，可以非常高效地处理并发请求，且支持很多常用的开发框架和库，例如 Express 框架、socket.io 库等。Node.js 是现代 Web 开发中必不可少的工具之一。

这些工具、框架和环境在现代 Web 开发中都扮演着重要角色，并广泛应用于各种 Web 应用的开发、测试和维护过程中。有了这些工具的支持，开发者可以更加高效地进行 Web 开发工作，提高开发效率和质量。

3 可行性分析与需求分析

3.1 校园众包递送系统现状分析

校园众包递送系统是一种利用互联网技术实现校园众包服务的平台，其可以将任务分配给学生团队或个人，以完成各种校园任务。随着互联网技术的不断发展，校园众包递送系统已经成为一种重要的校园服务模式，其具有以下现状：

一、平台数量不断增加。目前，国内有许多校园众包递送平台，如猪八戒、学生赚、校园任务等，这些平台提供了丰富的任务类型和便捷的服务，深受学生群体的喜爱。

二、任务类型不断丰富。校园众包递送系统的任务类型不断丰富，包括平面设计、编程开发、文案策划、翻译翻译等多种类型，能够满足不同学生的需求。

三、任务单价不断提高。随着市场的竞争，校园众包递送平台的任务单价不断提高，为学生们提供了更多的赚钱机会。

四、服务质量逐渐提升。随着校园众包递送系统的不断发展，许多平台开始注重服务质量的提升，通过提高任务的审核标准、严格的任务执行流程等方式，提高任务的完成质量。

五、平台监管不断加强。为了保障学生的权益，许多校园众包递送平台开始加强监管，对任务的真实性、合法性等方面进行严格审核，防止诈骗等违法行为的发生。

综上，校园众包递送系统已经成为校园服务的重要方式之一，其数量不断增加，任务类型不断丰富，服务质量逐渐提升，平台监管不断加强，为学生们提供了更多的赚钱机会和更高效的校园服务。

3.3 校园众包递送系统软件的技术可行性分析

使用 Spring Boot 作为后端框架，结合 Vue.js 作为前端框架，同时使用 Axios 和 Node.js 实现后台服务和数据传输，是一个完全可行的校园众包系统软件开发方案。Spring Boot 是一种快速开发框架，其简化了 Spring 框架的配置和部署，同时提供了丰富的插件和工具集，能够快速开发高质量的 Web 应用程序。Vue.js 是一种流行的 JavaScript 前端框架，可用于开发灵活、高效的单页 Web 应用程序。Axios 是一种流行的 JavaScript 库，用于在浏览器和 Node.js 中执行 HTTP 请求。Node.js 是一种基于 Chrome V8 引擎

构建的 JavaScript 运行时环境，可用于开发高性能的后端服务。使用 Spring Boot 和 Vue.js 结合开发校园众包系统软件，可以实现前后端分离，从而使得开发更加高效和灵活。同时，Axios 和 Node.js 的使用可以保证后台服务和数据传输的高效和稳定性。此外，Spring Boot 和 Vue.js 都具有良好的可扩展性，可以轻松地扩展新功能和模块。总之，结合 Spring Boot、Vue.js、Axios 和 Node.js，开发校园众包系统软件是一个技术可行的方案，可以提供高效、稳定、灵活和易于扩展的校园众包系统软件。

3.2 校园众包递送系统功能需求分析

校园众包递送系统是一种为校园用户提供众包服务的平台，其主要功能包括任务发布、任务执行、任务反馈等。下面是一个校园众包递送系统的功能需求分析：

一、任务发布功能：用户可以通过系统发布任务，包括任务类型、任务金额、任务期限等信息。系统应该支持自定义任务类型和任务金额，并且能够对任务进行期限的限制。同时，系统应该能够自动生成任务链接，方便用户传递任务信息。

二、任务搜索功能：用户可以通过系统搜索任务，按照任务类型、任务金额、任务期限等进行筛选，方便用户快速找到符合自己需求的任务。

三、任务执行功能：完成任务后，系统应该支持任务的自动执行，包括任务的提交、审核、完成等过程。同时，系统应该能够实时追踪任务的状态，方便用户及时了解任务进展。

四、任务反馈功能：用户可以对任务进行评价，并且可以对任务的质量进行反馈。系统应该能够根据用户的反馈，对任务进行审核和评定，从而提高任务的完成质量和效率。

五、任务评价功能：用户可以对完成任务进行评价，包括任务的完成质量、完成任务的时间等。系统应该能够根据用户的评价，对任务进行评分，并且能够作为任务的审核和评定标准之一。

六、用户管理功能：系统应该支持用户注册、登录、个人信息修改等功能。同时，系统应该能够记录用户的个人信息、完成任务的情况等信息，方便用户管理和查询。

七、财务管理功能：系统应该支持任务的付款和收款，包括任务的悬赏金、完成任务后的佣金等。同时，系统应该能够自动生成财务记录，方便用户管理和查询。

八、数据分析功能：系统应该能够对用户的数据进行分析，包括用户行为、任务情况等，从而为平台的运营和优化提供数据支持。

综上所述，校园众包递送系统的功能需求包括任务发布、任务搜索、任务执行、任务反馈、任务评价、用户管理、财务管理、数据分析等方面，为校园用户提供高效、便捷的众包服务。

4 功能模块设计

4.1 UML 软件建模

在功能模块设计时，借助 **Rational Rose**，采用面向对象建模的软件建模方法对软件进行初步建模。软件的用户主体为学生，因此在用例图中只有学生一个角色。根据业务场景需要，学生应该具备的用例包括派单，接单，订单评价以及登录。如图 4-1 所示。

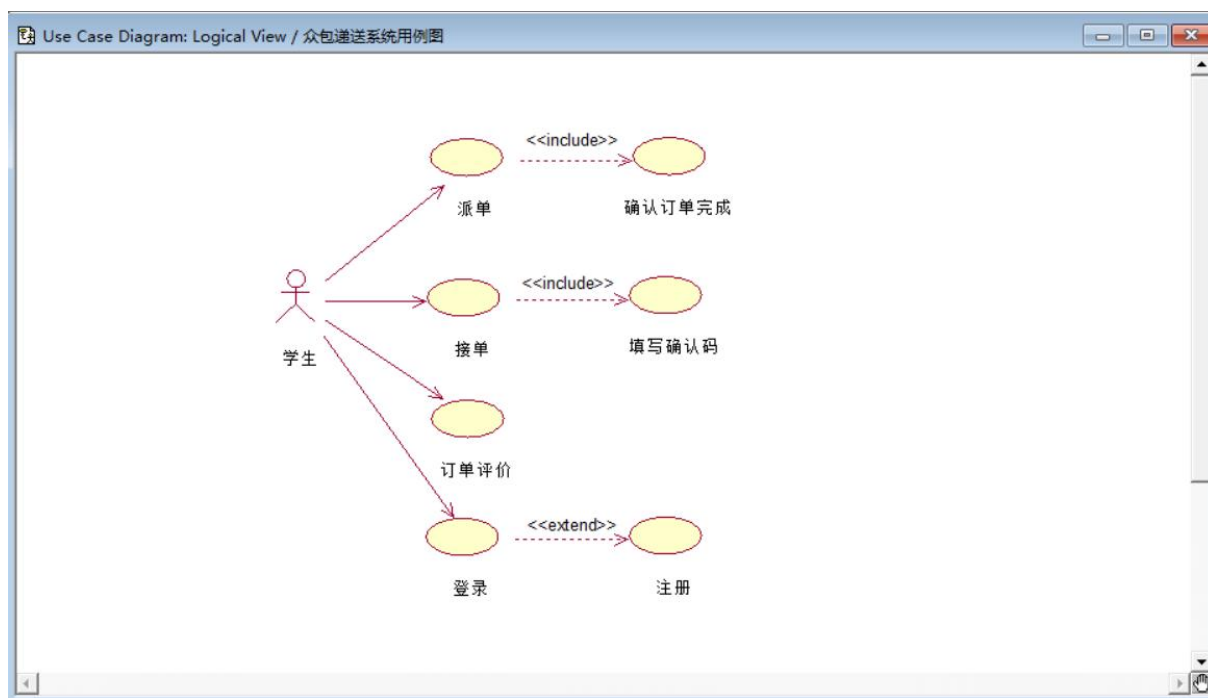


图 4-1 校园众包系统用例图

在进行用例图构建后，需要根据业务的场景执行顺序，根据系统的时间执行顺序建立序列图。系统的场景顺序如下：用户在发布订单后，系统收到这个订单。随后骑手可以在接单池进行接单和抢单操作。此时可以设置订单的状态为“已接单”状态。当且仅当在“已接单”状态的用户才能够进行付款以及评价订单的操作。当骑手将物品送往用户时，用户告知骑手收件码(确认码)，在骑手端即完成一个订单。此外，用户也可以在付款后立刻确认订单完成，这样不需要骑手确认订单完成也可以视为一个订单处于“完成”状态。本系统的 UML 建模序列图如图 4-2 所示。

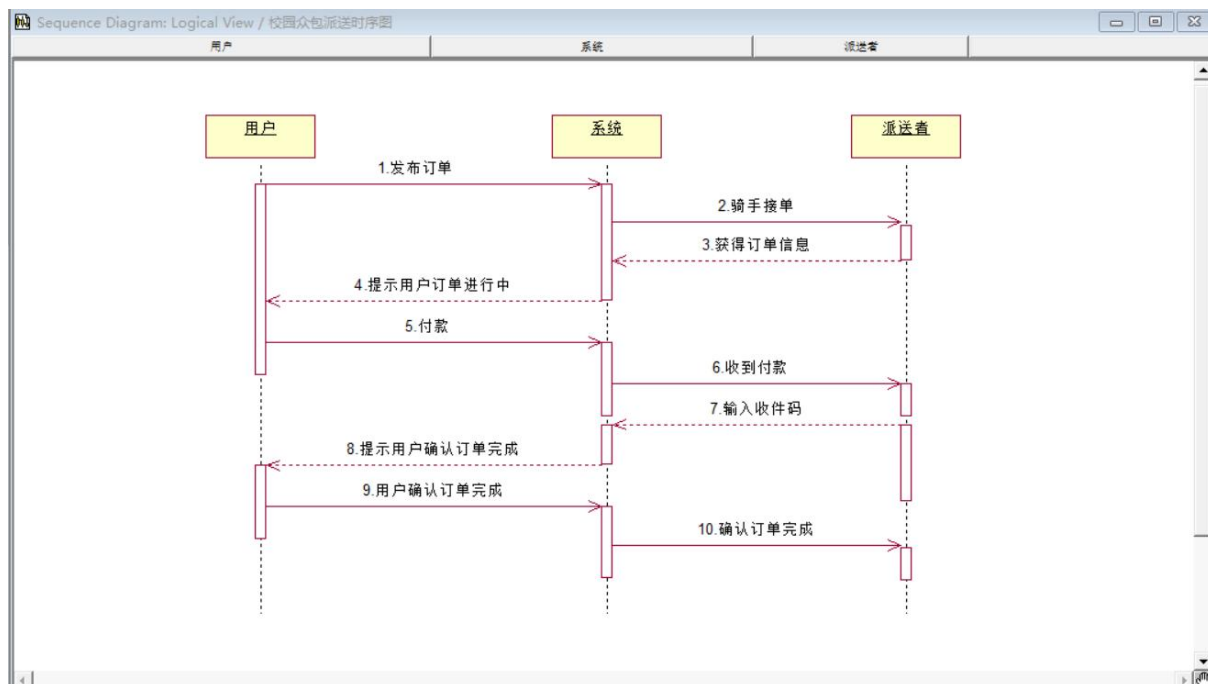


图 4-2 校园众包系统序列图

对于校园众包系统注册登录的 UML 活动图，其逻辑 UML 建模的详细设计思路与描述可以如下：

注册逻辑 UML 建模：

- (1) 创建一个用户类，包括姓名、邮箱、密码等属性。
- (2) 创建一个注册页面，包括输入姓名、邮箱、密码的文本框和一个“注册”按钮。
- (3) 用户在注册页面中填写相关信息并点击“注册”按钮。
- (4) 系统接收到用户的注册请求，验证用户填写的信息是否符合要求。
- (5) 如果用户填写的信息符合要求，系统将该用户的信息保存到数据库中，并提示用户注册成功。
- (6) 如果用户填写的信息不符合要求，系统将提示用户重新填写。

登录逻辑 UML 建模：

- (1) 创建一个用户类，包括邮箱、密码等属性。
- (2) 创建一个登录页面，包括输入邮箱、密码的文本框和一个“登录”按钮。
- (3) 用户在登录页面中填写邮箱、密码并点击“登录”按钮。
- (4) 系统接收到用户的登录请求，根据用户填写的邮箱和密码在数据库中查找是否有该用户。
- (5) 如果数据库中存在该用户，系统将该用户的信息加载到内存中，并提示用户登录成功。
- (6) 如果数据库中不存在该用户或者用户填写的密码不正确，系统将提示用户重新输入。

详细逻辑如图 4-3 所示。

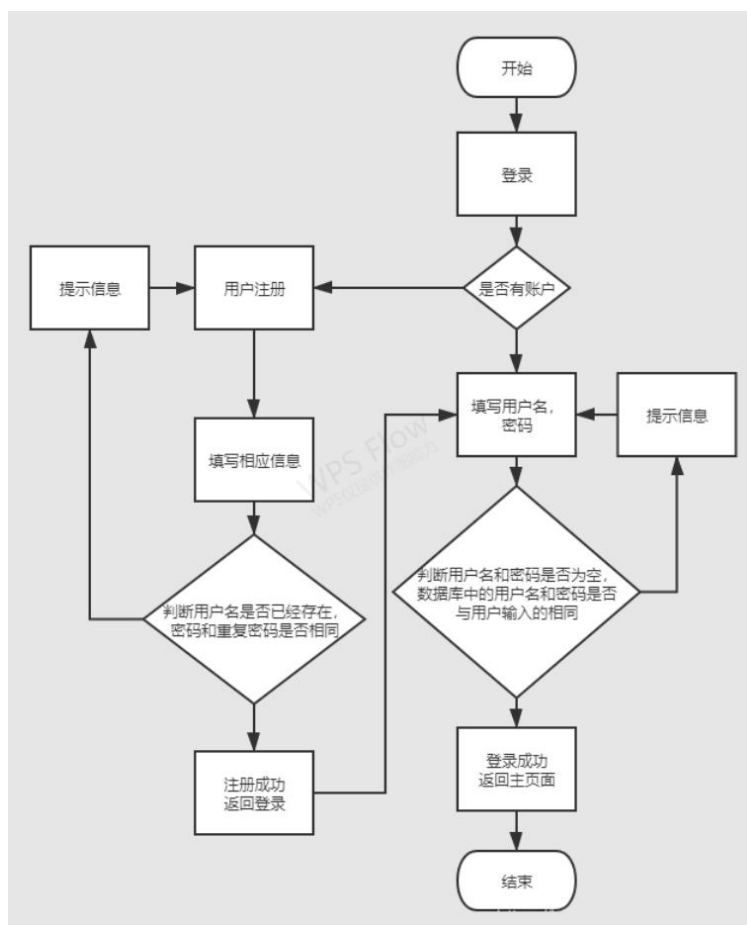


图 4-3 校园众包系统注册登录活动图

4.2 数据库设计

将概念结构设计中的各个模型转化为 DBMS 支持的表结构，同时保持不会出现插入异常、删除异常和修改异常。表结构应该做到符合第三范式。在本系统中，一共有两个实体，实体转化为数据库模型为如下所示：

(1) 实体转化为关系模式：

学生(学生 ID, 学生姓名, 学生性别, 生日, 邮箱, 手机号, 密码, 积分, 信用分)

订单(订单 ID, 物品名, 来源, 目的地, 状态, 接单人, 评论内容, 抢单人, 截止日期, 报酬, 支付状态)

(2) 一个一对多联系转化为一个关系模式

学生-订单(学生 ID, 订单 ID)

(3) 利用以上关系模式得到的所有数据表如下所示：

列名	数据类型	数据长度	可否为空	备注
Order_id	Integer	11	Not null	订单 ID (主键)
Name	varchar	11	Not null	接单人 ID (外键)



Source	varchar	45	Not null	来源
Destination	varchar	45	Not null	目的地
Status	Integer	1	null	订单状态
Student_id	Integer	4	null	接单者 ID (外键)
comments	varchar	11	null	评价内容
levels	Integer	1	Not null	评价级别
deadline	Datetime	18	Not null	截止日期
payment	Double	12	Not null	费用
Deliver_Id	Integer	14	Not null	下单用户
Note	varchar	45	Not null	备注
IsPaid	Integer	1	Not null	支付状态
PickUpCode	Integer	4	Not null	确认码

表 4-1 订单表

列名	数据类型	数据长度	可否为空	备注
Student_id	Integer	11	Not null	学生 ID (主键)
Username	Varchar	11	Not null	学生姓名
Gender	Integer	45	null	性别
Birthday	Date	20	null	生日
Email	Varchar	45	null	邮箱
points	Double	45	Not null	积分
credit	Integer	11	Not null	信誉分
password	Varchar	18	Not null	密码

表 4-2 学生表

本系统采用数据库的 E-R 图设计见图 4-4.

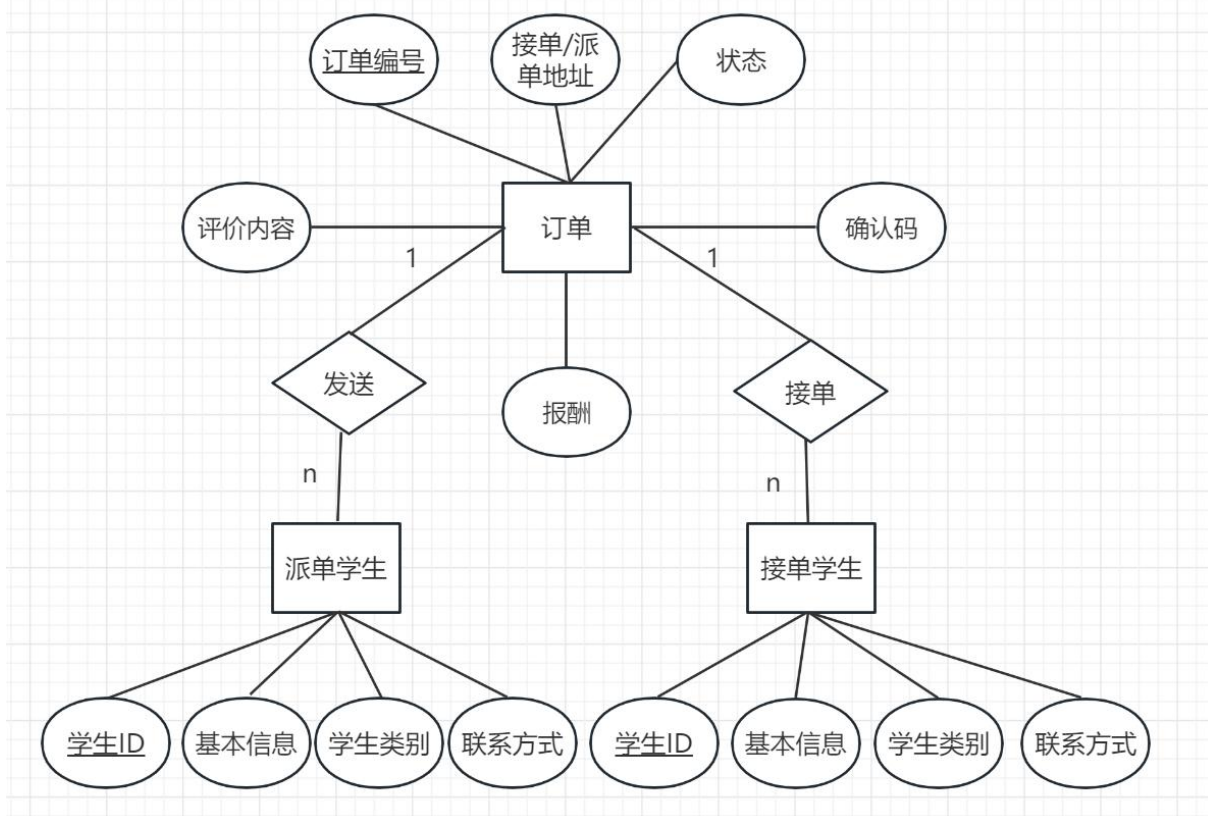


图 4-4 数据库设计 E-R 图

4.3 功能设计

4.3.1 登录

待登录的学生用户访问登录页面，首先在 Vue 前端的登录组件表单中输入用户名和密码。接着，学生用户提交登录信息，使用 Axios 向后端发送一个登录 POST 请求。后端系统验证用户提供的信息是否匹配。有两种情况，如果验证通过，系统将允许用户访问其个人账户，并进入操作界面。如果验证失败，则系统将提示用户重新输入他们的用户名和密码，且如果一直验证失败将会重复这一过程。

4.3.2 注册

用户访问注册页面时，前端将向后端发送一个 GET 请求，以获取注册页面的模板和数据。随后填写必要的注册信息，如用户名、密码、电子邮件地址等。前端将验证用户填写的数据是否合法，包括用户名是否唯一、密码是否符合规则、电子邮件地址格式是否正确等。如果验证通过，前端将向后端发送一个 POST 请求，以提交注册信息。如果验证失败，前端将向用户显示相应的错误信息。

后端将为注册页面提供一个 RESTful API，用于向前端提供注册页面的模板和数据。当收到前端提交的注册信息时，后端将对用户提供的数据进行验证，并确保用户名是唯一的，密码符合规则，电子邮件地址格式正确等。如果验证通过，后端将创建一个新的用户账号，并将该用户的信息保存到用户数据库中。后端将向用户发送一封确认电子邮件，以确保该用户提供的电子邮件地址有效。如果验证失败，后端将向前端返回相应的

错误信息。前端使用 Vue.js 作为主要技术，通过 Vue Router 进行页面路由管理，使用 Vuex 进行状态管理，使用 Axios 进行 HTTP 请求。后端使用 Spring Boot 作为主要技术，通过 Spring Security 进行用户认证和授权，使用 Spring Data JPA 进行数据持久化，使用 Java Mail API 发送电子邮件。通过前后端分离的设计方案，可以将前端和后端分开开发，降低系统的耦合度，使得系统更易于维护和扩展。同时，使用 Vue.js 和 Spring Boot 这两个流行的技术，可以提高开发效率和系统的稳定性。

4.3.3 派单

首先在前端设计表单。派单功能需要设计一个表单，该表单需要收集货物信息和派送地址信息。在 Vue 中可以使用 v-model 指令绑定表单元素到 Vue 组件的数据属性中。为了更好的用户体验，可以使用 Element UI 等前端框架来设计表单样式。随后设计一个提交按钮，用户在填写完毕表单后，可以点击按钮提交表单数据。在 Vue 中可以使用 @click 绑定按钮点击事件，同时可以使用 v-if 或 v-show 指令来控制按钮的状态，例如禁用或启用。也要进行异常处理，因为在用户提交表单数据时，可能会出现异常情况，例如网络连接中断或服务器错误。为了更好的用户体验，需要在前端设计一个异常处理机制，例如提示用户出现异常情况并且提供重试机制。

派单功能需要保存表单数据到数据库中。可以设计一个数据表用于存储货物信息和派送地址信息。在 MyBatis 中可以使用 Mapper 接口和 SQL 语句来操作数据库。控制层是整个后端的核心部分，负责接收前端的请求，处理业务逻辑，并返回数据。在 SpringMVC 中，控制层使用 @Controller 注解定义。派单功能的控制层需要设计一个接口来接收前端表单数据，然后将表单数据保存到数据库中。设计服务层来控制层和数据访问层之间的中间层，负责处理业务逻辑，例如数据验证和数据转换。在 Spring 中，服务层使用 @Service 注解定义。派单功能的服务层需要对表单数据进行验证，例如验证货物信息是否为空，验证派送地址是否合法。最后，设计数据访问层负责与数据库进行交互，使用 MyBatis 框架进行开发。在 Spring 中，数据访问层使用 @Repository 注解定义。派单功能的数据访问层需要实现 Mapper 接口，使用 SQL 语句来操作数据库。后端也要设计异常处理，因为在后端处理请求过程中，可能会出现异常情况，例如数据库连接失败或数据操作异常。为了更好的用户体验，需要在后端设计一个异常处理机制，例如使用 @ControllerAdvice 注解定义一个异常处理器来处理异常情况。

4.3.4 接单

首先进行前端 Vue 设计。对接单池页面设计，前端 Vue 需要实现一个接单池页面，该页面展示所有待接单的订单列表。每个订单需要展示订单号、订单状态、起点和终点等基本信息。当学生骑手点击某个订单时，前端需要发送一个请求给后端 SpringBoot，请求接单。请求需要包含订单号和学生骑手的信息（例如学生骑手的 ID）。后端 SpringBoot 需要根据请求内容进行订单分配，将订单状态改为“已接单”，并将订单分配给对应的学生骑手。然后设计夺取订单操作，当学生骑手想夺取已被其他学生骑手接单的订单时，前端需要发送一个请求给后端 SpringBoot，请求夺取。请求需要包含订单号和学生骑手的信息。后端 SpringBoot 需要判断学生骑手的信誉分是否足够高，如果足够高，则将该订单的状态改为“已接单”，并将订单分配给新的学生骑手。如果信誉分不足，则不进行操作。

随后进行后端 SpringBoot 设计。首先是订单信息的存储。后端 SpringBoot 需要设计

一个数据库模型来存储订单信息。每个订单需要包含订单号、订单状态、起点和终点等基本信息。同时,每个订单还需要存储该订单的接单学生骑手信息(例如学生骑手的 ID)。然后设计接单操作,后端 **SpringBoot** 需要实现一个接口,用于接收前端 **Vue** 发送的接单请求。接口需要根据请求内容进行订单分配,将订单状态改为“已接单”,并将订单分配给对应的学生骑手。如果订单已经被其他学生骑手接单,则返回错误信息。夺取订单操作的实现,后端 **SpringBoot** 需要实现一个接口,用于接收前端 **Vue** 发送的夺取订单请求。接口需要根据请求内容判断学生骑手的信誉分是否足够高,如果足够高,则将该订单的状态改为“已接单”,并将订单分配给新的学生骑手。如果信誉分不足,则返回错误信息。最后是订单状态的更新后端 **SpringBoot** 需要实现一个定时任务,用于更新订单状态。例如,如果一个订单已经被接单 30 分钟,但是学生骑手还没有完成配送,则该订单的状态需要改为“超时未完成”。同时,系统需要将该订单重新放回接单池,让其他学生骑手可以接单。

4.4.5 付款

系统需根据订单的金额计算出需要扣除的积分,例如,每 1 元扣除 1 积分。在 **Spring Boot** 中,通过使用 **MyBatis-Plus** 框架操作数据库,将用户的积分进行扣除。如果支付成功,系统会返回支付成功的信息给用户,并将订单状态改为已付款。在 **Vue3** 中,可以通过异步请求的方式获取支付结果,根据结果提示用户支付成功或失败。如果支付失败或者取消订单,系统会将扣除的积分进行退回。在 **Spring Boot** 中,可以通过使用数据库的事务操作保证退款操作的原子性。

在校园众包递送平台中,采用积分支付的方式可以避免对接第三方支付平台的复杂度和安全性问题。积分可以通过用户消费、签到等方式获得,可以激励用户的参与度和忠诚度。订单审核则可以保证平台的安全性和服务质量,避免出现虚假订单或者恶意下单的情况。在审核过程中,需要对订单信息进行校验,验证订单的合法性和可行性。付款时,通过计算订单金额,可以确定需要扣除的积分数量。在扣除积分之前,需要对用户的积分余额进行校验,确保用户积分充足。如果积分不足,则提示用户充值或者选择其他支付方式。在订单取消或者支付失败的情况下,需要将扣除的积分进行退回。在退回积分之前,需要对订单状态进行校验,确保订单处于已支付状态。在退回积分之后,需要更新用户的积分余额。

在进行付款操作时候,应当使用事务操作。在扣除积分和退回积分的过程中,需要对用户的积分余额进行修改,如果其中任意一步出现异常,可能会导致用户积分的不一致性。因此,在这些操作中,使用数据库的事务操作可以保证所有操作的原子性,即要么全部执行成功,要么全部回滚。这样可以保证数据的一致性和可靠性。在订单支付过程中,可能会涉及到多个线程的操作,例如,订单审核、积分扣除、积分退回等等。为了提高系统的并发性能,可以使用多线程的方式进行处理。在 **Java** 中,可以使用线程池的方式来管理线程的创建和销毁,避免频繁的线程创建和销毁带来的性能损耗。

4.4.5 用户评价

设计用户评价功能时候,首先在订单详情页面添加评价表单。在订单详情页面中,我们可以添加一个评价表单,供学生用户填写评价内容和评分,并提交表单以保存评价。然后实现订单评价数据的存储。在后端,我们需要使用 **Spring Boot** 框架来处理提交的评价表单数据,并将其存储到订单评价表中。在订单列表页面中,我们可以添加一个新

的列，用于展示该订单的评分。当学生用户填写并提交了评价表单后，该列会自动更新为该订单的平均评分。为了实现接收订单的人可以通过订单详情页面查看该订单的评价，可以在订单详情页面的底部添加一个评价列表，用于展示所有已提交的评价。最后，实现评价数据的更新。如果学生用户对订单的评价发生了变化，他们可以通过提交一个新的评价表单来更新评价内容和评分。我们需要在后端实现相应的更新逻辑，并将更新后的评价数据存储到订单评价表中。

5 编码

5.1 登录

以下代码实现了校园众包平台的登录功能。代码包含两个类：LoginController 和 LoginService。LoginController 类是 Spring Boot 的一个 RestController，它通过 HTTP POST 请求处理来自前端的登录请求。如果登录成功，则返回 HTTP 状态码 200 和消息“Login Success”，如果登录失败，则返回 HTTP 状态码 401 和消息“Login failed”。LoginService 类是一个服务类，它包含了处理登录请求的实际业务逻辑。它通过使用 StudentsMapper 类提供的方法，从数据库中获取用户输入的用户名和密码。如果数据库中不存在该用户名，且输入的密码与数据库中存储的密码匹配，则返回 true，否则返回 false。LoginService 类还包括其他两个方法：getId 和 getPoints。getId 方法接收一个学生对象，返回该学生在数据库中的 ID。getPoints 方法接收一个用户名，返回该用户在数据库中的积分值。

核心代码：

```
@RestController
public class LoginController {
    @Autowired
    LoginService loginService;
    @PostMapping("/studentlogin")
    public ResponseEntity<String> checkLogin(@RequestBody Students student)
    {
        if(loginService.logincheck(student.getUsername(), student.getPassword())) {
            return new ResponseEntity<>("Login Success", HttpStatus.OK);
        }
        else{
            return new ResponseEntity<>("Login failed",HttpStatus.UNAUTHORIZED);
        }
    }

    @Service
```




```
public class LoginService {
    @Autowired
    StudentsMapper studentsMapper;
    public boolean logincheck(String username,String password)
    {
        return Objects.equals(studentsMapper.getPasswordByuserName(username),
password);
    }

    public int getId(String username)
    {
        return studentsMapper.getIdByuserName(username);
    }

    public float getPoints(String username)
    {
        return studentsMapper.getPointByuserName(username);
    }
}

@PostMapping("/studentfindId")
public int findIdByName(@RequestBody Students student)
{
    return loginService.getId(student.getUsername());
}

@GetMapping("/studentfindPoints/{username}")
public float getPoints(@PathVariable("username") String username){
    return loginService.getPoints(username);
}
}
```

5.2 注册

以下代码实现了校园众包平台的注册功能。代码包含两个类：RegisterController 和 RegisterService。RegisterController 类是 Spring Boot 的一个 RestController, 它通过 HTTP POST 请求处理来自前端的注册请求。RegisterController 类中的 register 方法接收一个学生对象，并调用 RegisterService 类的 register 方法将学生信息插入到数据库中。RegisterService 类是一个服务类，它包含了处理注册请求的实际业务逻辑。它通过使用 StudentsMapper 类提供的 insert 方法，将学生信息插入到数据库中。

核心代码：

```
@RestController
public class RegisterController {
    @Autowired
    RegisterService registerService;
```

```
@PostMapping("/userregister")
public void register(@RequestBody Students students)
{
    System.out.println(students);
    registerService.register(students);
}
}
@Service
public class RegisterService {
    @Autowired
    StudentsMapper studentsMapper;
    public void register(Students student)
    {
        studentsMapper.insert(student);
    }
}
```

5.3 接单与抢单

下面的代码实现了接单与抢单功能，前端使用 Vue.js 框架实现了页面交互，后端使用 Spring Boot 框架实现了接口。

前端部分：`mounted()`钩子函数用于在组件挂载时获取所有订单信息，即调用后端接口/`getallorders`，并将返回的订单信息赋值给页面中的 `orders` 数组。`grabOrder()`方法用于当用户点击“接单”按钮时将订单信息传递给后端接口/`graborders`，后端接口将根据订单信息进行处理，将订单状态修改为“已接单”并记录接单人的 ID。`steal()`方法用于当用户点击“抢单”按钮时，先获取当前用户的信誉分数，再获取该订单已有接单人的信誉分数。如果当前用户的信誉分数高于已有接单人的信誉分数，则将订单信息传递给后端接口/`graborders` 进行处理，抢单成功；否则提示用户抢单失败。

后端部分：`getAllorders()`方法用于获取所有订单信息，即调用数据访问层的方法查询数据库中的所有订单信息并返回。`grabOrders()`方法用于处理接单请求，将接单人的 ID 存入订单信息中并将订单状态修改为“已接单”，最后调用数据访问层的方法更新数据库中的订单信息。`findAllordersById()`方法用于获取当前用户发送的所有订单信息，即调用数据访问层的方法查询数据库中所有订单信息并根据当前用户的 ID 筛选出该用户所发送的所有订单信息并返回。`findAllmyOrdersById()`方法用于获取当前用户接的所有订单信息，即调用数据访问层的方法查询数据库中所有订单信息并根据当前用户的 ID 筛选出该用户所接的所有订单信息并返回。`getThisOrder()`方法用于根据订单 ID 获取该订单的所有信息，即调用数据访问层的方法查询数据库中该订单的所有信息并返回。

前端核心代码：

```
methods: {
  async grabOrder(order) {
    try {
      order.studentId=localStorage.getItem("student_id")
      order.orderStatus=1;
      const response = await axios.post('http://localhost:8080/graborders', order);
```




```
if (response.status === 200) {
    alert("接单成功");
    order.orderStatus=1;
} else {
    // 接单失败
    this.error = response.data.message;
}
} catch (e) {
    this.error = '接单失败';
}
},
async steal(order) {
    const nowUserId = localStorage.getItem("student_id")
    const existedUserId = order.studentId;
    axios.get(`http://localhost:8080/findStudent/${nowUserId}`)
        .then(response => {
            // 从后端获取当前用户的信誉分
            const nowCredit = response.data.credit;
            axios.get(`http://localhost:8080/findStudent/${existedUserId}`)
                .then(response=>{
                    {
                        // 从后端获取已接单用户的信誉分
                        const existedCredit = response.data.credit;
                        console.log(nowCredit)
                        console.log(existedCredit)
                        if(nowCredit>existedCredit)
                        {
                            order.studentId=localStorage.getItem("student_id")
                            order.orderStatus=1;
                            axios.post('http://localhost:8080/graborders', order);
                            alert("抢单成功!")
                        }
                    }
                    else{
                        alert("抢单失败!您的信誉分不够! ")
                    }
                }
            })
        })
        .catch(error => {
            console.error(error);
        });
}
```



```
},  
mounted() {  
  const API_URL='http://localhost:8080/getallorders'  
  axios.get(API_URL, {withCredentials: true})  
    .then(response=>{  
      this.orders = response.data  
      console.log(this.orders)  
    }).catch(error=>{  
      console.log(error)  
    })  
}
```

后端核心代码:

@RestController

public class GrabController {

@Autowired

GrabService grabService;

@GetMapping("/getallorders")

public List<Orders> getAllOrders(){

return grabService.getAllorders();

}

@PostMapping("/graborders")

public void grabOrder(@RequestBody Orders order){

grabService.grabOrders(order);

}

//获取所有当前用户发送的订单

@GetMapping("/findOrdersById/{id}")

public List<Orders> findOrderById(@PathVariable("id") int id){

return grabService.findAllordersById(id);

}

//获取所有当前用户接的订单

@GetMapping("/findMyOrdersById/{id}")

public List<Orders> findMyOrderById(@PathVariable("id") int id){

return grabService.findAllmyOrdersById(id);

}

// 根据当前订单的 id 获取当前订单的所有信息

@GetMapping("/findOrderById/{id}")

public Orders getThisOrder(@PathVariable("id")int id)

{

return grabService.getThisOrder(id);

}

}

5.4 订单处理

在前端代码中,定义了一些方法来实现各种功能。例如,showDialog 和 showDialogII 方法用于显示对话框;submit 方法用于提交订单评价;payOrder 方法用于付款;confirmSendFinish 方法用于确认订单已经送达;confirmFinish 方法用于确认订单已经完成;toggleSentOrders 和 toggleGetOrders 方法用于切换已发送订单和已接收订单的显示状态。

在付款功能的实现中,前端会向后端发送一条 POST 请求,携带当前学生的信息和需要付款的订单信息,后端接收到请求后会更新该学生的余额信息。同时,前端还会向后端发送一条 GET 请求,获取收款用户的信息,并在收款用户账户上进行相应的收款操作。

在订单评价功能的实现中,前端会向后端发送一条 POST 请求,携带当前订单的信息和评价内容,后端接收到请求后会更新该订单的评价信息。

在后端代码中,定义了两个 RestController 和两个 Service。PayController 用于处理与支付相关的请求,包括获取用户信息和进行收款操作。CommentController 用于处理订单评价相关的请求,包括更新订单评价信息。同时,两个 Service 分别用于处理与支付和评价相关的业务逻辑。

前端核心代码:

```
methods: {
  showDialog() {
    this.dialogVisible = true;
  },
  showDialogII() {
    this.dialogVisibleII = true;
  },
  submit(order) {
    axios.post('http://localhost:8080/comment',order)
      .then(response => {
        alert("评价成功");
      })
      .catch(error => {
        console.log(error);
        alert('评价失败, 请重新尝试! ');
      });
  },
  async payOrder(order) {
    try {
      this.Student.points = this.Student.points - order.payment
      const response = await axios.post('http://localhost:8080/payOrder',
this.Student);
      if (response.status === 200) {
        alert("付款成功");
        // 修改当前订单的状态
        order.isPaid = 1;
        order.orderStatus = 2;
      }
    }
  }
}
```



```
        await axios.post('http://localhost:8080/graborders', order);
    } else {
        this.error = response.data.message;
    }
} catch (e) {
    this.error = '付款失败,请重试';
}
//从后端获收款用户的所有信息
const getPaidId = order.studentId;
axios.get(`http://localhost:8080/findStudent/${getPaidId}`)
    .then(async response => {
        this.Deliver = response.data;
        //收款开始
        //收款用户收款
        try {
            this.Deliver.points = this.Deliver.points + order.payment
            const response = await axios.post('http://localhost:8080/payOrder',
this.Deliver);

            if (response.status === 200) {
                // console.log('收费方的余额:', this.Deliver)
            } else {
                this.error = response.data.message;
            }
        } catch (e) {
            this.error = '收款失败,请重试';
        }
        //收款结束
    })
    .catch(error => {
        console.error(error);
    });
this.$forceUpdate();
},
async confirmSendFinish(order) {
    // 获得当前订单的取件码
    axios.get(`http://localhost:8080/findOrderById/${order.orderNumber}`)
        .then(response => {
            order.pickupCode = response.data.pickupCode;
        })
        .catch(error => {
            console.error(error);
        });
    if (order.pickupCode === this.checkedCode)
    {
```



```
// 取件码匹配
try {
  order.orderStatus = 3;
  const response = await axios.post('http://localhost:8080/graborders', order);
  if (response.status === 200) {
    alert("确认此商品已送达,待对方确认");
  } else {
    this.error = response.data.message;
  }
} catch (e) {
  this.error = '订单未完成,请重新确认';
}
}
else{
  alert("取件码不匹配,请重新输入!");
}
},
async confirmFinish(order) {
  if(order.isPaid === 1)
  {
    try {
      order.orderStatus = 3;
      const response = await axios.post('http://localhost:8080/graborders', order);
      if (response.status === 200) {
        alert("此订单已完成");
        order.orderStatus = 3;
      } else {
        this.error = response.data.message;
        alert("订单完成失败!")
      }
    } catch (e) {
      this.error = '订单未完成,请重新确认';
    }
  }
  else
  {
    alert("请先付款!");
  }
},
toggleSentOrders(){
  this.sentOrdersShow = !this.sentOrdersShow
},
toggleGetOrders(){
  this.getOrdersShow = !this.getOrdersShow
```



},

后端核心代码:

@RestController

public class PayController {

 @Autowired

 PayService payService;

 @GetMapping("/findStudent/{id}")

 public Students findStudentById(@PathVariable int id)

 {

 return payService.findStudentById(id);

 }

 @PostMapping("/payOrder")

 public void payOrder(@RequestBody Students student)

 {

 payService.payOrder(student);

 }

}

@Service

public class PayService {

 @Autowired

 StudentsMapper studentsMapper;

 public void payOrder(Students student){

 studentsMapper.updateById(student);

 }

 public Students findStudentById(int id) {

 return studentsMapper.selectById(id);

 }

}

@RestController

public class CommentController {

 @Autowired

 CommentService commentService;

 @PostMapping("/comment")

 public ResponseEntity<String> setcomment(@RequestBody Orders order)

 {

 commentService.setComment(order);

 return new ResponseEntity<>("Comment Success", HttpStatus.OK);

 }

}

@Service

public class CommentService {

 @Autowired


```
OrdersMapper ordersMapper;  
public void setComment(Orders order)  
{  
    ordersMapper.updateById(order);  
}  
}
```

6. 用户使用说明

6.1 登录

项目启动后，用户进入系统后可以直接看到登录界面，如图 6-1 所示。只有用户在输入了正确的用户名和密码后，才能够进入功能界面。如果用户没有账户，则可以点击“点击这里注册”来注册一个新的账户，从而进行登录操作。

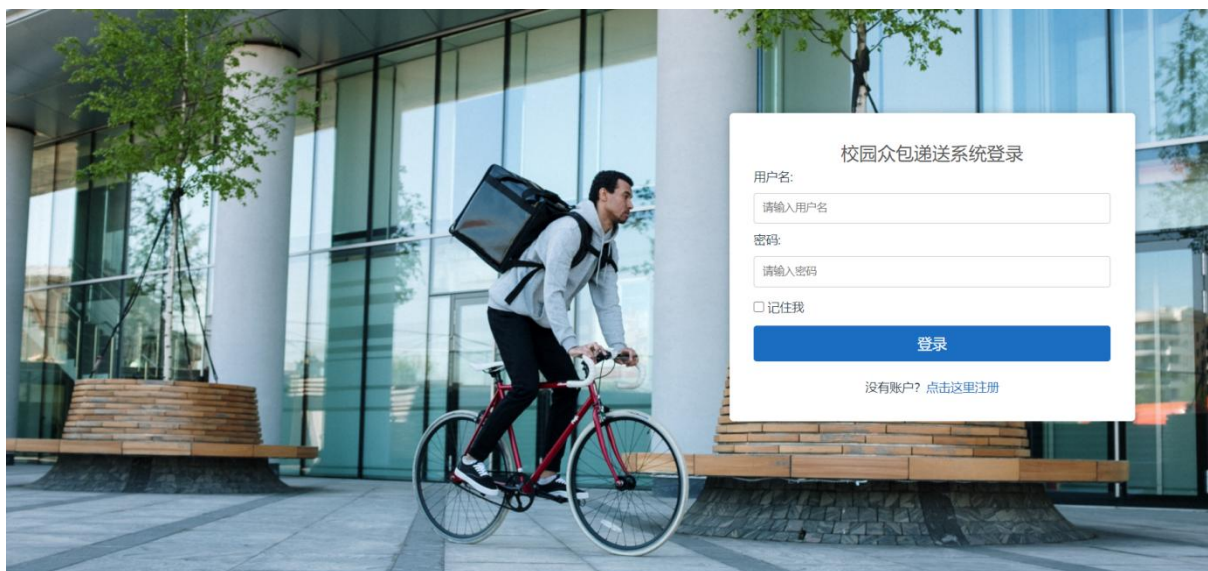
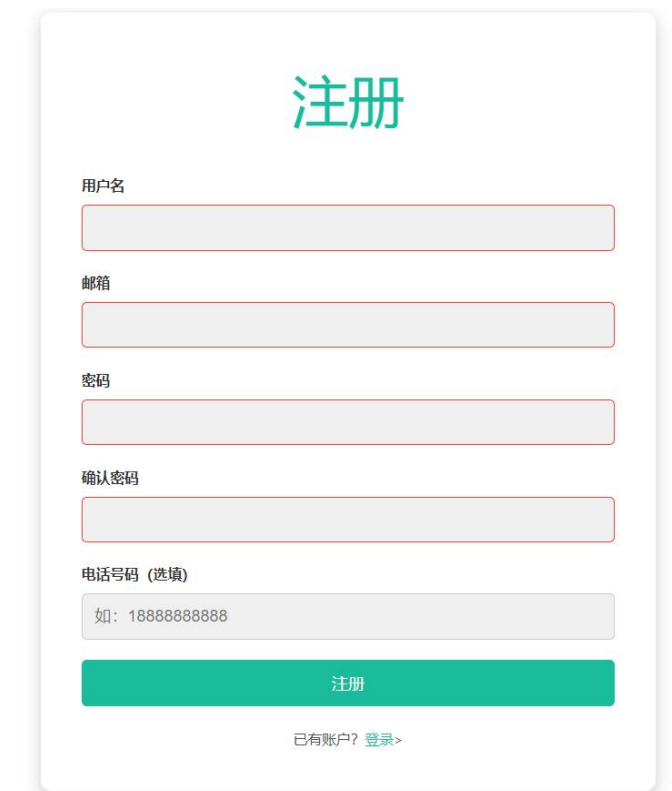


图 6-1 用户登录界面

6.2 注册

用户在进入注册页面后，输入数据库中不存在用户名，合法且格式正确的邮箱，两次输入的密码都正确才能够进行注册成功的跳转，否则将会根据不同的错误提示用户修改注册的信息，以便达到注册成功的目的，如图 6-2 所示。




The image shows a registration form titled "注册" (Register) in green. It contains the following fields and elements:

- 用户名** (Username): A text input field.
- 邮箱** (Email): A text input field.
- 密码** (Password): A text input field.
- 确认密码** (Confirm Password): A text input field.
- 电话号码 (选填)** (Phone Number, optional): A text input field with a placeholder example "如: 18888888888".
- 注册** (Register): A green button.
- 已有账户? 登录>** (Already have an account? Login >): A link below the register button.

图 6-2 注册

6.3 订单发布

在用户成功登录后，可以在菜单栏左上角看到自己的用户名和 ID，默认进入订单发布界面。用户可以根据需求发布自己的订单。需要填写的信息主要有物品名，取货地址，送货地址，截止送达的日期，报酬积分以及备注(如填写快递取件码和其他类似的重要信息等)。如果有一行信息填写有误，系统会进行默认检查并反馈给用户。但取货地址与送往地址必须由用户自己按照真实的地址填写。在填写完所有信息后，点击“发送订单”后，订单就可以在订单池中展示，以便其他学生进行后续的接单和抢单操作，如图 6-3 所示。用户如果想退出登录状态，则可以点击左下角的“退出登录”。



你好!周杰伦 用户id: 1

个人信息修改

订单发布

抢单管理

我的订单

我的积分: 9659

退出登录

物品:

取货地址:

送往地址:

截止送达日期:

年 / 月 / 日

报酬(积分数):

备注: (填写快递取件码等)

图 6-3 订单发布界面

6.4 抢单与接单

用户进入接单与抢单界面，如图 6-4 所示。用户可以接取“未被接单”的订单，也可以“试试抢单”。订单若从未被处理，则优先分配给先接单的用户。但如果用户的信誉较高，则可以抢走一个已经被接取的单。即订单的处理机制是双重的，这也是本项目的创新点之一。



你好!周杰伦 用户id: 1

个人信息修改

订单发布

抢单管理

我的订单

我的积分: 9659

退出登录

抢单管理

订单编号	商品名称	购买地点	收货地点	截止日期	报酬	操作	抢单
17	耳机	学海卢小区3	泉山别墅	2023-03-30T16:00:00.000+00:00	0	此单被接	此单完成
18	头盔	徐海路	云龙路	2023-03-29T16:00:00.000+00:00	19	此单被接	试试抢单
19	面罩	云龙湖	江苏师范大学	2023-04-04T16:00:00.000+00:00	222	立即接单	此单完成

图 6-4 抢单管理界面

6.5 我的订单

用户可以在我的订单查看所有“我发出的订单”与“我接的订单”。订单的详情信息包括订单编号，名称，截止日期，来源地，目的地备注，以及确认码都会直接显示在用户“我发出的订单”

界面。其中，确认码是需要收到物品后给予骑手随机生成的四位确认码。这个确认码的目的是确保用户的订单一定被送到才算完成这个订单，防止骑手未送到订单而将订单搁置，如图 6-5 所示。用户也可以在此界面查看当前发布订单的状态，并根据不同的状态进行不同的操作，如付款，评价，确认订单送达等。用户可以在“我接的订单”界面进行“确认订单送达”以及查看订单评价，如图 6-6 所示。当且仅当用户确认或输入确认码后才能够完成一个“我接的订单”。



图 6-5 我发出的订单界面

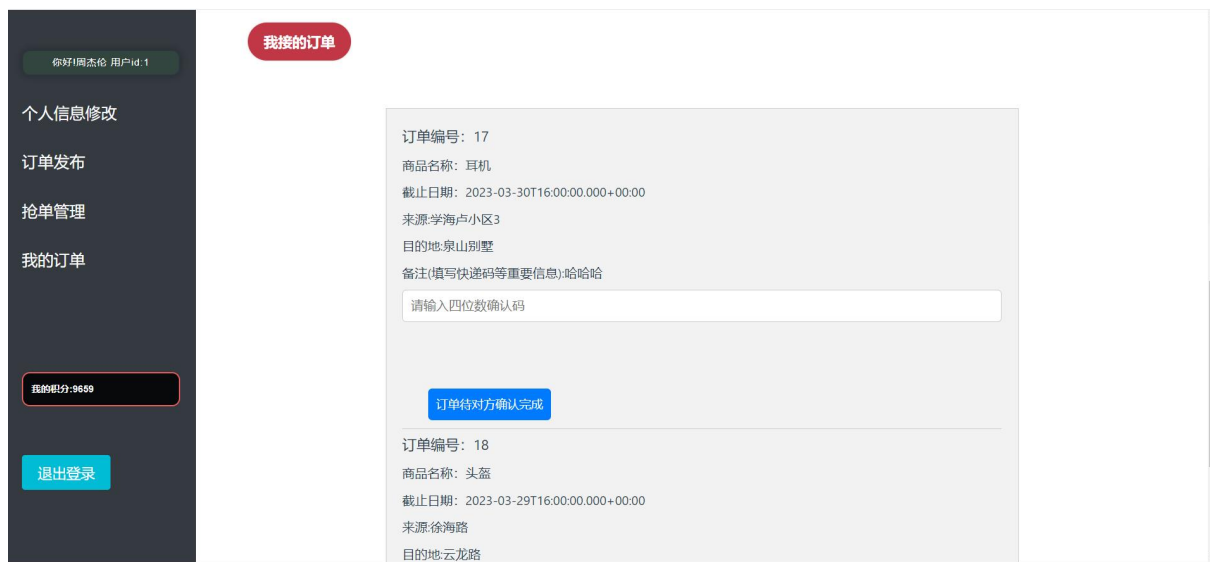


图 6-6 我接的订单界面

6.6 个人信息修改

点击菜单栏的“个人信息修改”，即可进入个人信息修改界面。用户可以根据需要修改自己的基本信息如姓名，生日，邮箱和重置密码等，如图 6-7 所示。

你好!周杰伦 用户id: 1

个人信息修改

订单发布

抢单管理

我的订单

我的积分: 9659

退出登录

姓名: 周杰伦

性别: ☒ 男 ☐ 女

生日: 2023/04/04

邮箱: 665444@qq.com

密码:

确认密码:

手机号: 110

保存

图 6-7 个人信息修改界面

7 软件测试

为了进行有效的测试，我们可以根据不同的测试原理，设计不同的测试用例。

首先设计黑盒测试用例：

黑盒测试基于软件的外部行为进行测试，我们可以设计以下测试用例：

登录测试用例

输入正确的用户名和密码，验证是否能够成功登录。测试结果符合期望结果。

输入错误的用户名和密码，验证是否会提示登录失败。测试结果符合期望结果。

输入不完整的用户名和密码，验证是否会提示登录失败。测试结果符合期望结果。

注册测试用例

输入符合要求的用户名、密码和电子邮箱，验证是否能够成功注册。测试结果符合期望结果。

输入不符合要求的用户名、密码和电子邮箱，验证是否会提示注册失败。测试结果符合期望结果。

订单派送测试用例

添加一个新订单并派送，验证是否能够成功添加和派送订单。测试结果符合期望结果。

添加一个订单但未派送，验证是否会提示订单未派送。测试结果符合期望结果。

添加一个已存在的订单，验证是否会提示订单已存在。测试结果符合期望结果。

付款测试用例

在积分足够的情况下，尝试付款，验证是否能够成功扣费并支付订单。测试结果符合期望结果。

在积分不足的情况下，尝试付款，验证是否会提示积分不足。测试结果符合期望结果。

抢单测试用例

根据不同的用户信誉大小，尝试抢单，验证是否会按照用户信誉大小派送订单。测试结果符合期望结果。

尝试抢单但未登录，验证是否会提示需要先登录才能抢单。测试结果符合期望结果。

订单评价测试用例

在完成订单后进行评价，验证是否能够成功评价订单。测试结果符合期望结果。

尝试评价未完成的订单，验证是否会提示订单未完成。测试结果符合期望结果。

查看订单评价测试用例

查看已经评价的订单，验证是否能够成功查看订单评价。测试结果符合期望结果。

查看未评价的订单，验证是否会提示订单未评价。测试结果符合期望结果。

然后设计白盒测试用例：

白盒测试基于软件内部的逻辑进行测试，我们可以设计以下测试用例：

代码覆盖率测试用例。

对所有函数和分支进行测试，验证是否能够覆盖所有代码。

条件覆盖测试用例：

针对所有的条件进行测试，验证是否能够正确判断条件。

路径覆盖测试用例：

针对所有的路径进行测试，验证是否能够正确处理所有的路径。

最后设计边界值测试用例：

边界值测试基于测试参数的边界值进行测试，我们可以设计以下测试用例：

登录测试用例：

输入超过最大长度的用户名和密码，验证是否会提示登录失败。测试结果符合期望结果。

输入最小长度的用户名和密码，验证是否会提示登录失败。测试结果符合期望结果。

注册测试用例：

输入超过最大长度的用户名、密码和电子邮箱，验证是否会提示注册失败。测试结果符合期望结果。

输入最小长度的用户名、密码和电子邮箱，验证是否会提示注册失败。测试结果符合期望结果。

订单派送测试用例：

添加订单时，验证订单金额是否超过最大金额和最小金额的限制。测试结果符合期望结果。

添加订单时，验证订单数量是否超过最大数量和最小数量的限制。测试结果符合期望结果。

付款测试用例：

在积分足够的情况下，尝试支付超过最大金额和最小金额的订单，验证是否会提示订单金额超过限制。测试结果符合期望结果。测试结果符合期望结果。

在积分不足的情况下，尝试支付订单，验证是否会提示积分不足。测试结果符合期望结果。

抢单测试用例：

在用户信誉最大值的情况下，尝试抢单，验证是否会成功抢单。测试结果符合期望结果。
在用户信誉最小值的情况下，尝试抢单，验证是否会提示抢单失败。测试结果符合期望结果。

订单评价测试用例：

尝试评价订单时，评分是否在最大值和最小值之间。测试结果符合期望结果。

尝试评价订单时，评论是否超过最大长度和最小长度的限制。测试结果符合期望结果。

根据以上测试用例，可以对系统进行测试，并得出以下测试结论：

系统能够成功地处理用户的登录和注册操作，并能够按照用户信誉大小派送订单。

系统能够成功地处理订单派送和支付操作，并能够限制订单金额和数量的范围。

系统能够成功地处理订单评价和查看订单评价操作，并能够限制评价的范围和长度。

在测试过程中，未发现系统存在严重的安全漏洞和功能缺陷，系统运行稳定，性能表现良好。通过对系统的黑盒测试、白盒测试和边界值测试，可以全面地验证系统的正确性、可靠性和安全性，并为后续的开发和优化提供有价值的参考。

表 7-1 部分软件测试用例

测试模块	测试用例设计	预期结果	实际结果
登录	用户名: ‘#¥#’，密码: ‘ ’	登录失败	登录失败
登录	用户名: ‘李璠’，密码 ‘123456’	登录成功	登录成功
注册	输入手机号为 ‘2333’ 两次输入密码为 ‘#’ 和 ‘22’	提示错误	提示错误
接单	登录后接取未被接单的订单	成功	成功
注册	输入手机号为 ‘#’ 输入用户名为 ‘¥%¥¥’	提示错误	提示错误
接单	不登录接单	无权限	无权限
抢单	点击已被接的单，信誉分小于已接单用户的信誉分	失败	失败
抢单	点击已被接的单，信誉分大于已接单用户的信誉分	成功	成功
支付	积分为 0 支付	提示错误	提示错误
评价	未登录评价	无权限	无权限
评价	评价非自己的订单	无权限	无权限
支付	积分足够点击支付	成功	成功

8 软件环境配置

8.1 前端环境配置

8.1.1 vite 环境配置

Vite 是一款基于 Vue3 的构建工具，它可以帮助您快速构建高性能、可维护性和可扩展性的应用程序。在安装 Vite 之前，您需要先安装 Node.js 和 npm 包管理器。

可以使用以下命令安装 Vite 及其依赖项：

```
npm install vite
npm install vite-client
npm install vite-optimizer
npm install vite-server
```

安装完成后，您可以使用以下命令启动 Vite 服务器：

```
vite
```

这将启动一个本地服务器，开发者可以在浏览器中访问该服务器，以查看 Vite 应用程序。要配置 Vite 环境，可以使用 vite.config.js 文件。该文件允许开发者更改 Vite 的默认设置，例如设置构建选项、缓存策略、代码分割等。要配置 Vite 环境，您可以编辑 vite.config.js 文件。该文件通常位于项目的根目录中。可以在该文件中设置以下选项：

build: 该选项用于配置构建选项。您可以在该选项中设置构建参数，例如压缩代码、提取公共模块、生成静态资源等。

cache: 该选项用于配置缓存策略。您可以在该选项中设置缓存文件夹、缓存条件、缓存清除策略等。

preload: 该选项用于配置预加载策略。您可以在该选项中设置预加载模块、预加载顺序等。

server: 该选项用于配置 Vite 服务器。您可以在该选项中设置服务器参数、端口号、请求拦截器等。

optimizer: 该选项用于配置 Vite 的代码分割器。您可以在该选项中设置代码分割策略、分割文件名等。

除了 vite.config.js 文件外，您还可以使用 Vite 提供的 config-file 命令来编辑配置文件。该命令将打开一个文本编辑器，您可以在其中编辑配置文件。

要使用 Vite 构建应用程序，需要创建一个 Vue3 组件库，并将其导出。可以使用以下命令创建该库：

```
npx create-vite-client-app my-app
```

该命令将在项目根目录中创建一个名为 my-app 的应用程序，并将其导出为 Vue3 组件库。可以在该应用程序中使用 Vite 构建应用程序。

Vite 提供了一种简单且高效的方式来构建 Vue3 应用程序。使用 vite 命令启动 Vite 服务器，并使用 vite.config.js 文件进行环境配置。还可以使用 config-file 命令来编辑配置文件，以更改 Vite 的默认设置。最后，使用 Vite 构建应用程序，并将其导出为 Vue3 组件库。

```
import { fileURLToPath, URL } from 'node:url'
import { defineConfig } from 'vite'
```

```
import vue from '@vitejs/plugin-vue'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [vue()],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    }
  }
})
```

8.1.2 node.js 配置与 npm 配置

在 Vue3 应用程序中，Node.js 和 npm 是必需的开发工具。下面是本项目中 Node.js 和 npm 配置的描述：

要使用 Vue3 应用程序，需要安装 Node.js。可以使用以下命令安装 Node.js：

```
npm install -g node@latest
```

安装完成后，可以在命令行中运行以下命令来启动 Node.js：

```
node --version
```

确保 Node.js 版本与正在使用的 Vue3 版本兼容。例如，如果正在使用 Vue3 版本 3.x，则需要安装 Node.js 版本 8.x 或更高版本。

npm 是 Vue3 应用程序的包管理器。要使用 npm，需要安装它。可以使用以下命令安装 npm：

```
npm install -g npm@latest
```

安装完成后，可以在命令行中运行以下命令来启动 npm：

```
npm --version
```

确保 npm 版本与正在使用的 Vue3 版本兼容。注意，如果正在使用 Vue3 版本 3.x，则需要安装 npm 版本 6.x 或更高版本。要在 Vue3 应用程序中使用 npm，需要将 npm 命令添加到 Vue3 应用程序的构建环境中。这可以通过使用 vite.config.js 文件来实现。在该文件中，添加以下代码来配置 npm：

```
import npm from '@/config/npm'
```

```
export default {
  build: {
    npm: npm
  }
}
```

这将配置 npm 命令，以便在 Vue3 应用程序的构建环境中使用。

安装 Node.js 和 npm 是开发 Vue3 应用程序的必要条件。在安装完成后，需要将 Node.js 和 npm 配置为在 Vue3 应用程序中使用。通过使用 vite.config.js 文件，可以配置 npm 并在 Vue3 应用程序的构建环境中使用它。

8.1.3 Vue3 组件路由配置

配置组件路由以确认 HTML 组件渲染的 DOM 交给全局事件总线 BUS 处理，并且使得 Vue3 中的 VM 处理 DOM。

```
const routes=[
  {path:'/',name:Login,component:Login},
  {path:'/register',name:Register,component: Register},
  {path:'/personalInfo',name:PersonInfo,component:
PersonInfo,meta: {requiresAuth:true}},
  {path:'/comment',name:Comment,component: Comment,meta: {requiresAuth:true}},
  {path:'/grab',name:Grab,component: Grab,meta: {requiresAuth:true}},
  {path:'/release',name:Release,component: Release,meta: {requiresAuth:true}},
  {path:'/myorders',name:MyOrders,component: myOrders,meta: {requiresAuth:true}}
];
```

RequiresAuth 为权限设置参数，默认为 True，防止用户输入域名恶意跳转，操作数据库等操作。

8.2 后端环境配置

8.2.1 解决跨域请求问题

前后端分离项目跨域请求是指前端应用程序（浏览器）向后端服务器发送请求，请求的数据来自于不同的域名，例如，前端应用程序的域名为 `example.com`，而后端服务器的域名为 `api.example.com`。由于网络协议的限制，不同域名下的网页无法直接相互访问，因此出现了跨域问题。

跨域问题会对前后端分离的应用程序造成一些影响，例如：

后端接口限制：由于跨域问题，前端应用程序无法直接向后端服务器发送请求，因此需要使用代理方式或者跨域请求库来解决跨域问题。

性能问题：由于跨域问题，前端应用程序需要发送额外的请求来获取数据，这会增加应用程序的响应时间和性能问题。

安全问题：跨域问题可能会导致一些安全问题，例如跨域脚本攻击（CSRF）等。

常见的解决跨域问题的方法包括：

JSONP:JSONP(JSON with Padding) 是一种利用<script>标签的 src 属性没有跨域限制的特性，实现跨域访问的技术。

代理：在本地服务器上建立一个代理服务器，将前端应用程序的请求发送到代理服务器，再由代理服务器向后端服务器发送请求，从而达到跨域访问的目的。

CORS:CORS(Cross-Origin Resource Sharing) 是一种允许浏览器向跨域服务器发送请求的技术，它通过给请求添加头部信息，使得服务器可以允许浏览器发送跨域请求。

WebSocket:WebSocket 是一种双向通信协议，它可以让前端应用程序与后端服务器进行实时通信，并且不受跨域限制。

API 密钥：对于一些需要授权访问的 API，可以使用 API 密钥来进行身份验证，

从而避免跨域问题。

本项目采取前端代理和后端 CORS 配置解决跨域请求问题。

```
import Vue from 'vue'
export function send(data) {
  return new Promise((resolve, reject) => {
    const url = window.location.href
    axios.post(url, data, { headers: { 'Access-Control-Allow-Origin':
'*' } })).then(resolve, reject)
  })
}

import Vue from 'vue'
import axios from 'axios'
export function makeRequest(url, data) {
  return new Promise((resolve, reject) => {
    axios.post(url, data, { headers: { 'Access-Control-Allow-Origin':
'*' } })).then(resolve, reject)
  })
}
```

后端解决跨域问题:

```
/**
 * 解决跨域问题
 */
@Configuration
public class CrossConfig implements WebMvcConfigurer {
  @Override
  public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**")
      .allowedOriginPatterns("*")
      .allowedMethods("GET", "HEAD", "POST", "PUT", "DELETE",
"OPTIONS")
      .allowCredentials(true)
      .maxAge(3600)
      .allowedHeaders("*");
  }
}
```

8.2.2 Spring Boot 数据库环境配置

要在 Spring Boot 应用程序中连接数据库，需要定义一个 yml 文件来配置数据库连接信息。以下是一个基本的 Spring Boot 应用程序的 yml 文件示例，其中连接到一

个名为"mydb"的 MySQL 数据库:

spring:

application:

name: myapp

database:

name: mydb

url: jdbc:mysql://localhost:3306/mydb

username: root

password: password

mybean:

name: databaseConnection

proxyBeanName: databaseConnection

要使用这个 yml 文件, 需要将其复制到 Spring Boot 应用程序的根目录中, 并将其命名为"application.yml"。然后, 可以使用这个 yml 文件来配置应用程序的各种参数, 包括数据库连接信息。

要连接到数据库, 可以使用 Spring 提供的 DatabaseOperations 类。以下代码段将创建一个数据库连接:

```
@Autowired
private DatabaseOperations databaseOperations;

public void connectToDatabase() {
    databaseOperations.createConnection("mydb", "root", "password");
}
```

在本项目中, 直接使用 yml 文件中定义的参数来创建一个数据库连接。

spring:

datasource:

url: jdbc:mysql://localhost:3306/deliverysystem

driverClassName: com.mysql.cj.jdbc.Driver

username: root

password: 1111

type: com.alibaba.druid.pool.DruidDataSource

spring:

thymeleaf:

cache: false

server:

port: 8080

8.2.3 Maven 配置

Maven 是一个流行的开源项目构建工具，它可以帮助开发人员构建、测试和部署应用程序。它提供了一种简单而可靠的方式来构建复杂的应用程序，可以在不同的构建环境中 (如 IDE、命令行、Web 浏览器等) 运行。Maven 具有以下优点:

自动化构建:Maven 可以自动处理依赖项和构建。

可重复性:Maven 可以确保每次构建都相同，并且可以记录构建历史记录。

低学习曲线:Maven 与其他构建工具 (如 Ant 和 Gradle) 相比，其使用方式更加简单。

灵活性:Maven 可以自定义构建规则，以适应不同的项目需求。

要使用 Maven，需要完成以下步骤:

安装 Maven:可以从 Maven 官方网站 (<https://maven.apache.org/>) 下载最新版本的 Maven。

安装后，启动 Maven。

创建 Maven 项目:选择“File”菜单，然后选择“New Project”。在出现的对话框中，选择项目名称、所在的目录以及项目类型。

配置 Maven 项目:在创建 Maven 项目后，需要配置项目的一些参数。例如，需要配置项目的依赖项、构建参数、测试参数等。可以使用 Maven 属性文件或 XML 配置文件来配置这些参数。

运行 Maven:要运行 Maven，需要在命令行中进入 Maven 项目目录，然后输入以下命令:

mvn [运行命令]

例如，要运行 Maven 构建，可以输入以下命令:

mvn clean package

要运行 Maven 测试，可以输入以下命令:

mvn test

要运行 Maven 部署，可以输入以下命令:

mvn deploy

以上是 Maven 的详细介绍以及配置方法。使用 Maven，可以快速构建和部署应用程序，而且可以帮助开发人员简化构建过程。本项目所用到的重要的 Maven 核心依赖如下:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.7.5</version>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
```



```
<scope>test</scope>
</dependency>
<!--      <dependency>-->
<!--      <groupId>org.springframework.boot</groupId>-->
<!--      <artifactId>spring-boot-starter-data-jpa</artifactId>-->
<!--      </dependency>-->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>3.4.3.1</version>
</dependency>
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-generator</artifactId>
    <version>3.4.1</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid</artifactId>
    <version>1.1.4</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.3.23</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>2.0.24</version>
</dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
```

9 总结

本文主要总结了一个基于 SSM 框架的校园众包递送系统的设计与实现。该系统能够管理校园递送服务的信息，提供任务派单、进度管理、订单管理和用户信息管理等功能，并与本地数据库 MySQL 连接，采用了 Spring Boot、Axios、Vue3 和 Node.js 等技术。首先，本文介绍了系统的需求分析和设计过程。通过分析校园递送服务行业的实际需求，确定了系统的功能模块和设计方案。其中，使用了 MVC 架构模式，前端采用 Vue3 框架进行开发，后端采用 Spring Boot 框架，实现了前后端分离的设计方案。同时，通过 Axios 实现前后端数据的交互，并与本地 MySQL 数据库进行连接，实现了数据的持久化存储。其次，本文介绍了系统的具体实现过程。在前端部分，通过 Vue3 实现了页面的布局和交互逻辑，使用组件库优化了用户体验。在后端部分，使用 Spring Boot 框架搭建了基于 RESTful 风格的 API 接口，通过 Maven 管理依赖，实现了项目的快速构建和部署。同时，采用了 MyBatis-Plus 作为数据持久化框架，实现了对 MySQL 数据库的访问和操作。通过对订单、用户信息和任务派单信息的管理，实现了对校园递送服务的有效协调和管理。最后，本文总结了该校园众包递送系统的优点和不足之处。该系统具有良好的可扩展性和可维护性，能够满足校园递送服务行业的基本需求。但是，在安全性、性能和个性化方面还有待进一步优化。对于安全性问题，需要加强对数据的保护和用户身份验证;对于性能问题，需要优化数据库的访问和查询效率，提高系统的响应速度和并发能力;对于个性化问题，需要在软件总体架构设计方面继续规划，设计出更加符合校园递送服务行业和用户的需求的 API 接口。

综上所述，本文介绍了一个基于 Spring Boot、Axios、Vue3 和 Node.js 等技术的校园众包递送系统的设计和实现。该系统具有一定的实用价值和参考意义，可以为其他类似系统的开发提供借鉴和参考。未来，可以进一步优化该系统的功能和性能，以适应不断增长的校园递送服务需求。

参考文献

- [1] 胡小召. 基于集成场理论的校园网购物流整合实践研究: 以西安欧亚学院为例[D]. 西安: 长安大学, 2015.
- [2] 黎 娟. 互联网众包对现代企业管理模式创新的启示及应用[J]. 商业经济研究, 2017(2): 113-115.
- [3] SAXTON G D, KISHORE R. Rules of Crowdsourcing: Models Issues and Systems of Control[J]. Information Systems Management, 2013, 30(1): 2-20.
- [4] 朱云桦, 栾迎霞, 孙晓君. 众包物流开拓城市配送蓝海的可行性研究[J]. 物流管理, 2016(6): 23-24.
- [5] 周金华. 众包物流模式下生鲜电商行业发展研究[J]. 科技和产业, 2016, 16(5): 33-35.
- [6] 纪汉霖, 周金华, 张深. 生鲜电商行业众包模式研究[J]. 物流工程与管理, 2016, 38(1): 93-95.
- [7] 任为. 基于快递众包的城​​市配送模式初探[J]. 物流工程与管理, 2015, 37(6): 122-124.
- [8] 程光, 孙培焱, 王位等. 高校快递代取服务终端的优化: 以江南大学为例[J]. 物流科技, 2014, 37(9): 111-114.
- [9] 王欣悦. 我国智慧物流发展问题及对策研究[J]. 铁道运输与经济, 2017, 39(4): 37-41.
- [10] 唐智鹏. 我国铁路物流发展形势及对策分析[J]. 铁道货运, 2017, 35(1): 31-34.
- [11] 童咏昕, 袁野, 成雨蓉, 等. 时空众包数据管理技术研究综述[J]. 软件学报, 2017, 28(1): 35-58.
- [12] BOUTSIS I, KALOGERAKI V. On task assignment for real-time reliable crowdsourcing[C]//2014 IEEE 34th International Conference on Distributed Computing Systems (ICDCS). Madrid, Spain, 2014: 1-10.
- [13] 宋天舒, 童咏昕, 王立斌, 等. 空间众包环境下的 3 类对象在线任务分配[J]. 软件学报, 2017, 28(3): 611-630.
- [14] 刘辉, 李盛恩. 时空众包环境下基于统计预测的自适应阈值算法[J]. 计算机应用, 2018, 38(2): 415-420.
- [15] HASSAN U U, CURRY E. Efficient task assignment for spatial crowdsourcing: a combinatorial fractional optimization approach with semi-bandit learning[J]. Expert systems with applications, 2016, 58: 36-56.