

# 数据库系统原理课程设计报告

——“帮帮我”校园互助系统数据库



|         |                  |
|---------|------------------|
| 学    院： | 电子信息与工程学院        |
| 专    业： | 计算机科学与技术         |
| 学    号： | 1952339          |
| 姓    名： | 张馨月              |
| 指导教师：   | 李文根              |
| 完成日期：   | 2022 年 06 月 14 日 |

# 目录

|          |                          |           |
|----------|--------------------------|-----------|
| <b>1</b> | <b>背景</b>                | <b>4</b>  |
| <b>2</b> | <b>需求分析</b>              | <b>4</b>  |
| 2.1      | 现状分析 . . . . .           | 4         |
| 2.2      | 功能需求 . . . . .           | 5         |
| 2.3      | 数据字典 . . . . .           | 6         |
| 2.3.1    | User . . . . .           | 6         |
| 2.3.2    | Course . . . . .         | 6         |
| 2.3.3    | Cue . . . . .            | 7         |
| 2.3.4    | CourseMaterial . . . . . | 7         |
| 2.3.5    | Task . . . . .           | 7         |
| 2.3.6    | Notification . . . . .   | 7         |
| 2.4      | 数据流图 . . . . .           | 8         |
| 2.4.1    | 顶层数据流图 . . . . .         | 8         |
| 2.4.2    | 增删资源数据流图 . . . . .       | 8         |
| 2.4.3    | 增删课程数据流图 . . . . .       | 9         |
| 2.4.4    | 发布、认领、完成需求数据流图 . . . . . | 9         |
| 2.4.5    | 通知数据流图 . . . . .         | 9         |
| 2.4.6    | 提交申诉数据流图 . . . . .       | 10        |
| <b>3</b> | <b>可行性分析</b>             | <b>10</b> |
| 3.1      | 技术可行性 . . . . .          | 10        |
| 3.2      | 应用可行性 . . . . .          | 10        |
| 3.3      | 经济可行性 . . . . .          | 11        |
| <b>4</b> | <b>概念设计</b>              | <b>11</b> |
| 4.1      | 实体 . . . . .             | 11        |
| 4.1.1    | 用户实体 . . . . .           | 11        |
| 4.1.2    | 课程实体 . . . . .           | 12        |
| 4.1.3    | 资源实体 . . . . .           | 12        |
| 4.1.4    | 任务实体 . . . . .           | 12        |
| 4.1.5    | 申诉实体 . . . . .           | 13        |
| 4.1.6    | 通知实体 . . . . .           | 13        |
| 4.2      | 实体属性局部 E-R 图 . . . . .   | 13        |
| 4.2.1    | 用户局部 E-R 图 . . . . .     | 13        |
| 4.2.2    | 课程局部 E-R 图 . . . . .     | 14        |
| 4.2.3    | 课程资料局部 E-R 图 . . . . .   | 14        |
| 4.2.4    | 通知局部 E-R 图 . . . . .     | 14        |

|          |                             |           |
|----------|-----------------------------|-----------|
| 4.2.5    | 任务局部 E-R 图 . . . . .        | 15        |
| 4.2.6    | 申诉局部 E-R 图 . . . . .        | 15        |
| 4.3      | 实体局部联系 E-R 图 . . . . .      | 15        |
| 4.3.1    | 课程与课程资料局部联系 E-R 图 . . . . . | 15        |
| 4.3.2    | 任务与通知局部联系 E-R 图 . . . . .   | 16        |
| 4.3.3    | 用户与课程局部联系 E-R 图 . . . . .   | 16        |
| 4.3.4    | 用户与课程资料局部联系 E-R 图 . . . . . | 16        |
| 4.3.5    | 用户与申诉局部联系 E-R 图 . . . . .   | 16        |
| 4.3.6    | 用户与任务局部联系 E-R 图 . . . . .   | 17        |
| 4.3.7    | 用户与通知局部联系 E-R 图 . . . . .   | 17        |
| 4.4      | 全局 E-R 图 . . . . .          | 18        |
| <b>5</b> | <b>逻辑设计</b>                 | <b>18</b> |
| 5.1      | 关系模型 . . . . .              | 18        |
| 5.1.1    | E-R 图向关系模型的转换 . . . . .     | 18        |
| 5.1.2    | 数据模型的优化及规范化设计 . . . . .     | 18        |
| 5.2      | 表结构设计 . . . . .             | 19        |
| 5.2.1    | User . . . . .              | 19        |
| 5.2.2    | Course . . . . .            | 20        |
| 5.2.3    | Cue . . . . .               | 20        |
| 5.2.4    | CourseMaterial . . . . .    | 20        |
| 5.2.5    | Task . . . . .              | 20        |
| 5.2.6    | Message . . . . .           | 21        |
| <b>6</b> | <b>物理设计</b>                 | <b>21</b> |
| 6.1      | 概述 . . . . .                | 21        |
| 6.2      | 索引设计 . . . . .              | 21        |
| <b>7</b> | <b>前端技术</b>                 | <b>22</b> |
| 7.1      | 前端结构设计 . . . . .            | 22        |
| 7.2      | 登录注册 . . . . .              | 23        |
| 7.3      | 主页 . . . . .                | 23        |
| 7.4      | 通知 . . . . .                | 24        |
| 7.5      | 用户信息 . . . . .              | 25        |
| 7.6      | 任务查看与接收 . . . . .           | 26        |
| 7.7      | 任务发布 . . . . .              | 27        |
| <b>8</b> | <b>后端技术</b>                 | <b>28</b> |
| 8.1      | 跨域问题 . . . . .              | 29        |
| 8.2      | 与数据库的连接 . . . . .           | 30        |

|           |                 |           |
|-----------|-----------------|-----------|
| <b>9</b>  | <b>测试</b>       | <b>31</b> |
| 9.1       | 数据库测试 . . . . . | 31        |
| <b>10</b> | <b>总结</b>       | <b>31</b> |

# 数据库系统原理课程设计报告

## ——“帮帮我”校园互助系统数据库

### 1 背景

基于目前校园生活的刚需，旨在合理利用同学们日常生活中因事务繁忙和空闲时间的差异化而产生的时间需求差，充分调查同学们在繁忙的学业和工作中产生的需求、的需求，了解好友间的互相代操作的工作流程。同时，参考现有的知识共享框架，将知识共享的内容融入系统中，以提升系统的实用性。本文档从系统的目的、架构、功能、可支持性、安全性等多方面，对系统进行了全方位分析和整合，确定了系统的适用对象、功能需求和性能需求等内容，为下一步的软件开发和测试打下了较好的基础并提供指向性内容。进而确定并完善一份可实现、可扩展、完整且合理的方案。

利用碎片化时间，满足了同学们日常生活中的代取快递、代购需求、跑腿及资料共享等需求。

### 2 需求分析

#### 2.1 现状分析

同学们日常的生活在校区内处于宿舍、食堂、教学楼（图书馆）三点一线式生活，学校的食堂处于校区中心位置，快递点分为校内快递集中点（递易中心）和校外快递集中点（旋转门外），两个快递点距离食堂都有一定路程（一个在教学楼一角，一个在校门外）。同学们还经常需要去学院楼办理一些事项，学院楼不同于教学楼在校区中心，而是分布在校区内各个地方，这使得单独一次前往学院楼的时间花费极大。

每天需要按时前往教学楼上课，课间 20 分钟还需要赶往下一个教室上课，所以拿取快递只能放在饭后时间进行，但是很多小组讨论、休息、社团聚集等事件又只能在饭后进行，所以抽出时间去取快递也变成了一件相对比较麻烦的事情。需要前往学院楼的事项，多半以班级为单位会有很多同学具有同样的需求，每个人单独去也会浪费许多时间，将这些任务全部交给班委也会增加班委的负担。

校区大为大家的生活带来了舒适的环境，但是因为忙碌的每日安排，也给大家带来了诸多不便。

现在很多同学会选择在寝室楼群里请求同学前去帮忙拿取快递，然后放在指定地点，因此一位同学可以一次拿回多位同学的快递，节省了大家的时间。对于借出物品归还等事件，很多同学容易忘记，需要物主反复催促，同学之间的关系容易因此产生嫌隙。很多同学喜欢校区内的猫猫狗狗，为它们准备了猫粮狗粮，却经常碍于自己有事在身而无法及时投喂，将粮食闲置过期。课程相关的资料目前仍然靠同学之间 1 对 1 交换、发送，没有对每门课程建立起良好的沟通和资料传阅环境，很多用过的教材会被闲置然后扔掉，十分可惜。

目前存在图书漂流活动可以方便同学们传阅资料，但仍仅限于图书资料，很多电子资料等无法方便求助于人。

## 2.2 功能需求

- **资料分享：**查看现有课程，添加新课程，查看现有课程资料，评价现有课程资料，添加资料，删除资料。

同学可以直接根据课号搜索对应课程的资料，然后进入资料库查看资料内容、下载资料，并且可以对该资料进行评分，评分细则可以显示在对应资料处，方便其他人查看。上传资料者可以新建对应课程和文件目录，然后上传资料并自主命名。

- **代取快递：**发布代取快递需求，查看现有快递需求，接取代取快递需求，评价功能

同学可以在任务发布一栏发布代取快递的任务，并写明取件相关信息，以及取到快递后要怎样交付。发布任务后可以获得发布成功的通知信息，并且能够从任务栏中看到自己发布的信息以及别人发布的信息。其余同学也可以在任务栏选择自己感兴趣的、方便的求助任务来进行接收，接收后能够收到接收成功的通知，然后任务栏中就无法再看到该条任务，避免重复接收。之后任务双方可以从通知栏中进入任务详情界面，发布者可以在任务完成之前取消任务，接收者可以选择放弃任务。被放弃的任务会重新回到任务栏中供其余同学接收。

- **代购物品：**发布代购需求，查看现有代购需求，接取代购需求，评价功能

同学可以在任务发布一栏发布代购物品的任务，并写明购买物品的相关信息，以及购买好后要怎样交付，包括价格等。发布任务后可以获得发布成功的通知信息，并且能够从任务栏中看到自己发布的信息以及别人发布的信息。其余同学也可以在任务栏选择自己感兴趣的、方便的求助任务来进行接收，接收后能够收到接收成功的通知，然后任务栏中就无法再看到该条任务，避免重复接收。之后任务双方可以从通知栏中进入任务详情界面，发布者可以在任务完成之前取消任务，接收者可以选择放弃任务。被放弃的任务会重新回到任务栏中供其余同学接收。

- **校区跑腿：**发布校区跑腿需求，查看校区跑腿需求，接取校区跑腿需求，评价功能

同学可以在任务发布一栏发布校区跑腿的任务，并写明需要前往的校区，以及取到物品后要怎样交付。发布任务后可以获得发布成功的通知信息，并且能够从任务栏中看到自己发布的信息以及别人发布的信息。其余同学也可以在任务栏选择自己感兴趣的、方便的求助任务来进行接收，接收后能够收到接收成功的通知，然后任务栏中就无法再看到该条任务，避免重复接收。之后任务双方可以从通知栏中进入任务详情界面，发布者可以在任务完成之前取消任务，接收者可以选择放弃任务。被放弃的任务会重新回到任务栏中供其余同学接收。

- **账号信息处理：**注册账号、登录账号、管理账号、退出账号、找回账号、封禁账号、申诉账号

进入系统时需要登录账号，无账号者可以注册账号，登录完成之后可以进入主页面，在用户详情页面可以对账号信息进行修改和提交。管理员可以对非管理员账号进行封禁，被封禁账号无法再用于登录，需要提交申诉信息，等待管理员处理，管理员将其解封后，原账号才可以正常登录。可以右上角退出账号，然后重新登陆其他账号。

- **系统信息提醒：**发布任务时通知、任务接受时通知、任务完成时通知、任务取消时通知  
通知栏查看通知的具体内容，每条通知有时间、标题、内容等信息，点击查看后可以看到通知的详细内容，可以通过通知跳转到对应的任务详情页，并对任务进行修改等。通知能够及时追溯任务状态变化，当任务被发布、被接收、被完成时，发布者都会收到通知，方便大家追踪求助任务信息。

## 2.3 数据字典

### 2.3.1 User

| 属性名     | 字段         | 类型      | 长度  | 约束                                 |
|---------|------------|---------|-----|------------------------------------|
| 用户编号    | id         | int     |     | primary key auto_increment notnull |
| 用户名     | username   | varchar | 45  | not null                           |
| 密码      | password   | varchar | 45  | not null                           |
| 电话号码    | telenumber | varchar | 45  | not null                           |
| 邮箱地址    | email      | varchar | 255 | not null                           |
| 学校      | university | varchar | 255 |                                    |
| 学号      | stuid      | varchar | 45  | not null                           |
| 院系      | department | varchar | 255 |                                    |
| 城市      | city       | varchar | 255 |                                    |
| 街道      | schdis     | varchar | 255 |                                    |
| 寝室号     | dorm       | varchar | 255 |                                    |
| 任务发布数   | publishing | INT     | 10  | unsigned zero fill                 |
| 任务接收数   | receiving  | INT     | 10  | unsigned zero fill                 |
| 发布任务完成数 | published  | INT     | 10  | unsigned zero fill                 |
| 接收任务解决数 | solved     | INT     | 10  | unsigned zero fill                 |

### 2.3.2 Course

| 属性名 | 字段         | 类型      | 长度  | 约束                    |
|-----|------------|---------|-----|-----------------------|
| 课程号 | courseid   | int     |     | primary key, not null |
| 课程名 | coursename | varchar | 255 | not null unique       |

**2.3.3 Cue**

| 属性名  | 字段     | 类型      | 长度  | 约束                                  |
|------|--------|---------|-----|-------------------------------------|
| 编号   | id     | int     |     | not null primary key auto_increment |
| 用户号  | userid | int     |     | 外键: User.id, not null               |
| 申诉原因 | reason | varchar | 255 | not null                            |

**2.3.4 CourseMaterial**

| 属性名  | 字段         | 类型           | 长度  | 约束                                     |
|------|------------|--------------|-----|--|
| 编号   | id         | int          |     | not null primary key                   |
| 用户号  | userid     | int          |     | foreign key: User.id, not null         |
| 课程号  | courseid   | int          |     | foreign key: Course.courseid, not null |
| 资料名  | name       | varchar      | 255 | not null                               |
| 存储路径 | path       | varchar      | 255 | not null                               |
| 得分   | score      | numeric(4,2) |     | not null default 0                     |
| 评价人数 | evaluators | int          |     | not null default 0                     |
| 下载人数 | downloads  | int          |     | not null default 0                     |

**2.3.5 Task**

| 属性名    | 字段          | 类型      | 长度   | 约束                                  |
|--------|-------------|---------|------|-------------------------------------|
| 编号     | id          | int     |      | not null primary key auto_increment |
| 发布者用户号 | publisherid | int     |      | foreign key: User.id, not null      |
| 认领者用户号 | receiverid  | int     |      | foreign key: User.id                |
| 任务标题   | title       | varchar | 255  | not null                            |
| 任务描述   | description | varchar | 2047 | not null                            |
| 任务状态   | status      | varchar | 127  | not null                            |

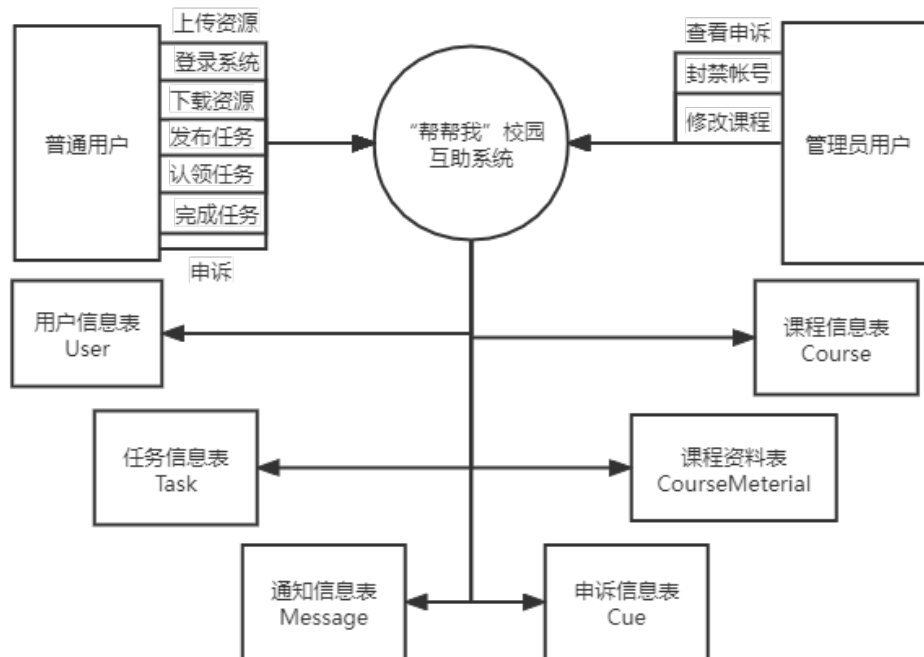
**2.3.6 Notification**

| 属性名    | 字段         | 类型      | 长度   | 约束                                  |
|--------|------------|---------|------|-------------------------------------|
| 编号     | id         | int     |      | not null primary key auto_increment |
| 接收者用户号 | receiverid | int     |      | foreign key: User.id, not null      |
| 任务编号   | taskid     | int     |      | foreign key: Task.id, not null      |
| 消息标题   | title      | varchar | 255  | not null                            |
| 消息内容   | content    | varchar | 1024 |                                     |
| 时间     | time       | DATE    |      | not null                            |

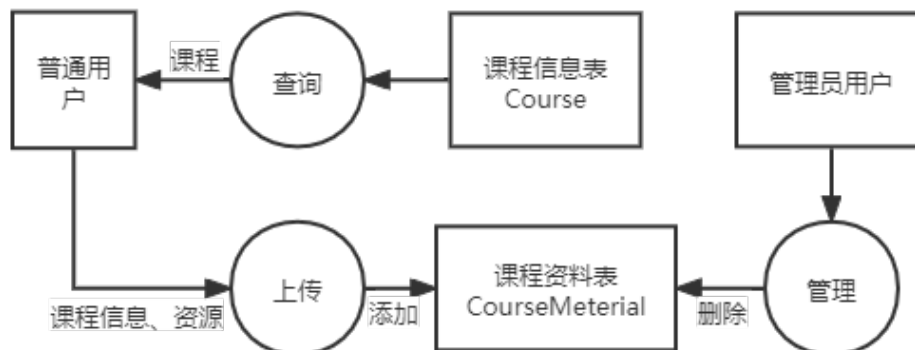


## 2.4 数据流图

### 2.4.1 顶层数据流图

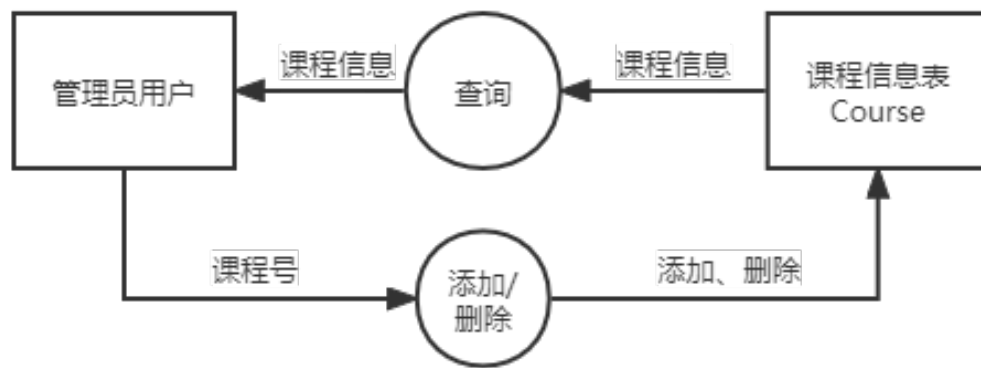


### 2.4.2 增删资源数据流图



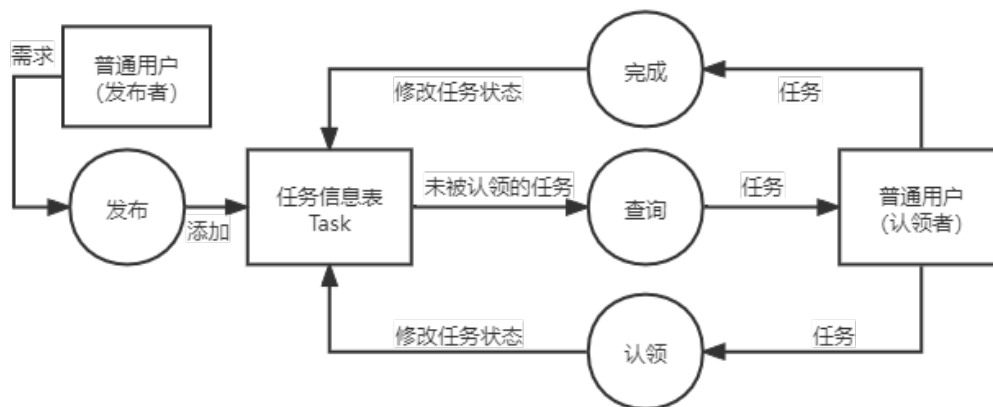
普通用户可以查询课程列表，查询课程对应的资料列表，对于没有列表的课程，可以先添加课程，然后再上传资料。管理员用户可以对资料进行删除操作。查询数据库表对象为 material

### 2.4.3 增删课程数据流图



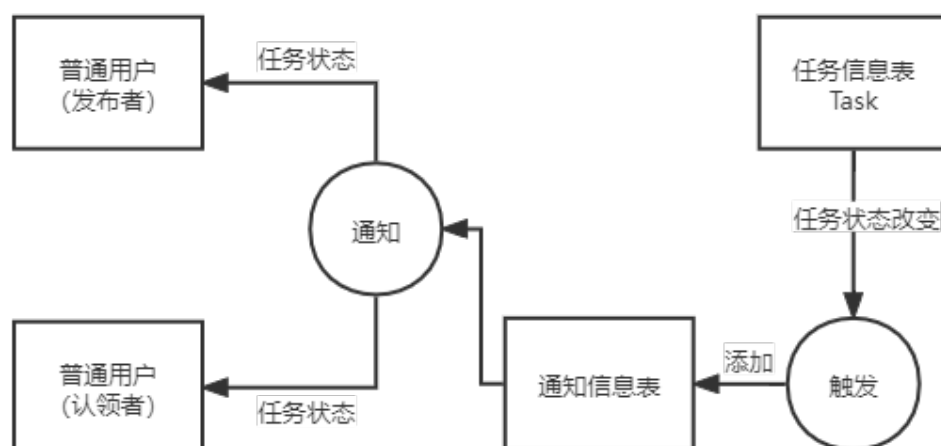
管理员用户可以查询课程列表，也可以增加、删除对应的课程项。查询数据库表对象为 course

### 2.4.4 发布、认领、完成需求数据流图



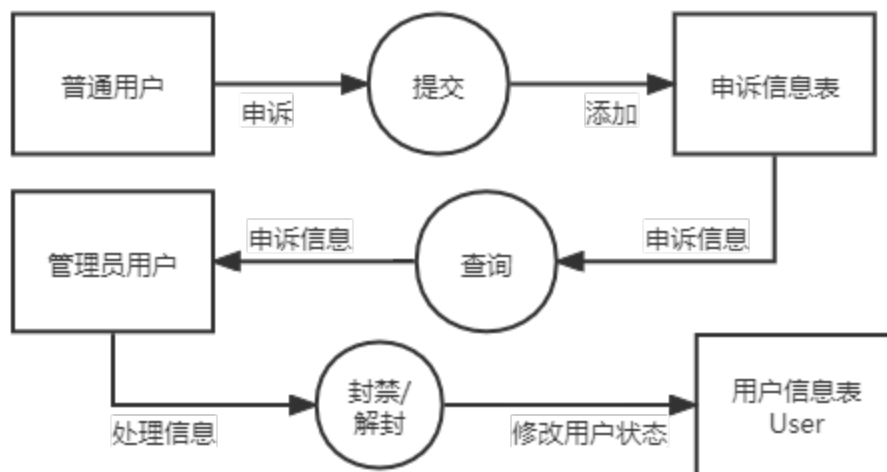
普通用户可以查询任务列表，并对任务进行发布操作，对应增加任务项操作；对任务进行接收，对应修改任务项操作；发布者对任务进行取消，对应删除任务项操作。查询数据库表对象为 task

### 2.4.5 通知数据流图



普通用户在对任务进行增加、修改、删除等操作后，都会收到系统通知，在通知栏对应查询通知列表操作。查询数据库表对象为 notification

### 2.4.6 提交申诉数据流程图



普通用户在账号被封禁后可以提交申诉，对应增加申诉项操作。查询数据库表对象为 cue

## 3 可行性分析

### 3.1 技术可行性

本次为“帮帮我”校园互助系统的后台数据库，存储资源文件、用户信息、课程信息、申诉、通知等内容，管理相对简单，借鉴之前软件工程课程中的知识设计该数据库，使用 mySQL 开发，运用数据库 SQL 语言灵活修改管理。

考虑使用 vue 前端以及 spring boot+mySQL 的前后端分离开发方式，具有相当的可行性。

### 3.2 应用可行性

该产品与现有的外卖跑腿网站有一定的类似度，但是区别于盈利软件，该软件更重视互帮互助，没有明显的服务提供者和消费者的界限，每个人都拥有双重身份。该产品很好地整合了学生们的互助需求，为同学们的日常生活提供便利，是一个具有自含性的新型软件，能够成为共享经济的一个重要组成部分。在可以预见的未来，该软件能够接入学校系统，作为志愿活动和勤工俭学的一部分。在更远的将来，可以形成全国性的软件，为不同学校的学生提供一个校友圈和互助平台，并实现资料的跨校交流和共享，真正成为我帮人人，人人帮我的共享平台。

### 3.3 经济可行性

系统中应用的开发工具为 IntelliJ，以及技术框架 Vue+Spring Boot+mySQL，使用 maven 进行管理，都是免费的，这无疑为系统的成本压缩了空间，从成本分析上看，该系统充分体现了将产品利益最大化的企业原则。开发过程中，采用前后端分离的方式，使得前后端代码的独立性更强，避免互相影响，对各个模块的功能设计区分界限清楚，做到了高聚类、低耦合，避免功能交叉和过程中反复修改，能够实现单元测试，符合软件开发流程。

## 4 概念设计

### 4.1 实体

#### 4.1.1 用户实体

描述：用户实体中包含了用户的用户名、密码、电话号码、邮箱地址、等级、状态等个人信息，属性。

- 编号：用户号，类型为 integer，主键，非空递增
- 用户名：用户名称，类型为 varchar，最大长度为 255
- 密码：用户登录密码，类型为 varchar，最大长度 255，非空
- 电话号码：用户电话号码，类型为 varchar，最大长度 255
- 邮箱地址：用户邮箱地址，类型为 varchar，最大长度 155，非空且唯一
- 等级：用户的等级，最低级为普通用户，然后是管理员用户
- 用户状态：当前帐号的状态，0 为被封禁帐号，其余为正常状态
- 学校：用户所属的学校名称，类型为 varchar，最大长度为 255
- 学号：用户在校期间学生编号，类型为 varchar，最大长度为 45，非空
- 院系：用户所属学院、专业，类型为 varchar，最大长度为 255
- 城市：用户居住地所在城市，类型为 varchar，最大长度为 255
- 街道：用户居住地所在街道，类型为 varchar，最大长度为 255
- 寝室号：用户所在学校寝室编号，类型为 varchar，最大长度为 255
- 任务发布数：用户在本系统中发布的全部任务数量，包括已经被接收的和未被接收的，类型为 INT，最大长度为 10，非负
- 任务接收数：用户在本系统中接收的全部任务数量，包括已经完成的和未完成的，类型为 INT，最大长度为 10，非负

- 发布任务完成数：用户在本系统中发布的任务被完成的数量，最大长度为 10，非负
- 接收任务解决数：用户在本系统中接收的任务成功解决的数量，最大长度为 10，非负

#### 4.1.2 课程实体

描述：课程实体中包含了课程的编号，课程号和课程名。

- 编号：编号，类型为 `integer`，主键，非空递增
- 课程号：学校的课程代码，类型为 `int`，非空
- 课程名：学校的课程名，类型为 `varchar`，最大长度 255，非空且唯一

#### 4.1.3 资源实体

描述：资源实体中包含了资源的编号、用户号、课程号、资料名、存储路径、得分、评价人数、下载人数等资料信息。

- 编号：资料编号，类型为 `integer`，主键，非空
- 用户名：用户号，类型为 `integer`，外键：User.id，非空
- 课程号：学校的课程代码，类型为 `int`，外键：Course.courseid，非空
- 资料名：资料名称，类型为 `varchar`，最大长度 255，非空
- 存储路径：资料存储地址，类型为 `varchar`，最大长度 255，非空
- 得分：用户对该资料的平均评分，类型为 `numeric(4,2)`，非空，默认值为 0
- 评价人数：对此资料做出评价的用户数，类型为 `int`，非空，默认值为 0
- 下载人数：对此资料进行下载的用户数，类型为 `int`，非空，默认值为 0

#### 4.1.4 任务实体

描述：任务实体中包含了任务的编号、发布者用户号、认领者用户号、任务标题、任务描述以及任务状态。

- 编号：资料编号，类型为 `integer`，主键，非空自增
- 发布者用户名：发布该任务的用户号，类型为 `integer`，外键：User.id，非空
- 认领者用户号：认领该任务的用户号，类型为 `integer`，外键：User.id，非空
- 任务标题：任务的标题，类型为 `varchar`，最大长度 255，非空
- 任务描述：任务的内容描述，类型为 `varchar`，最大长度 512，非空
- 任务状态：任务的状态，类型为 `integer`，描述是否被认领

#### 4.1.5 申诉实体

描述：申诉实体包含了申诉的编号、用户号和申诉原因

- 编号：资料编号，类型为 `integer`，主键，非空自增
- 用户号：用户号，类型为 `integer`，外键：User.id，非空
- 申诉原因：申诉解封、封禁帐号的原因，类型为 `varchar`，最大长度 255，非空

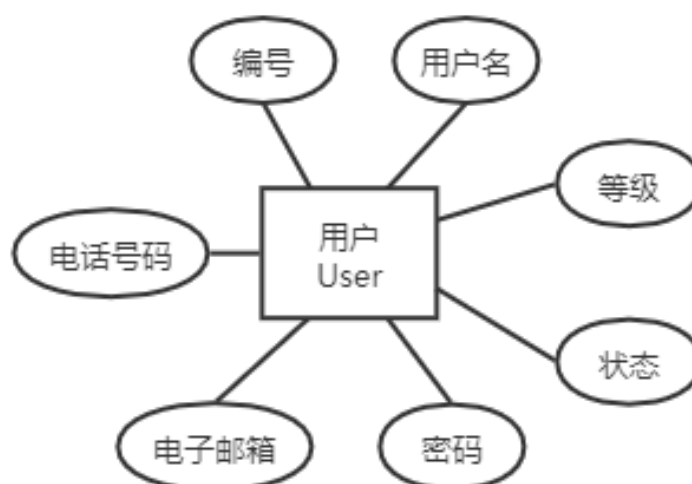
#### 4.1.6 通知实体

描述：通知实体包含了通知的编号、接收者用户号、消息标题和消息内容。

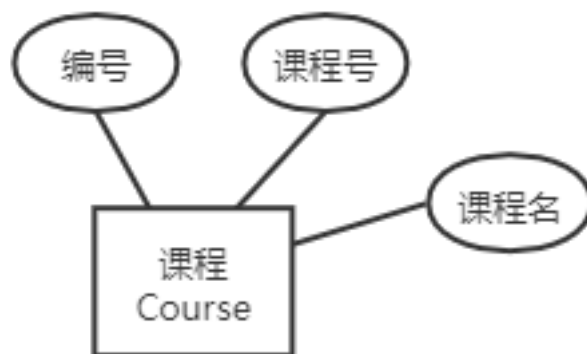
- 编号：资料编号，类型为 `integer`，主键，非空自增
- 接收者用户号：接收者用户号，类型为 `integer`，外键：User.id，非空
- 任务编号：该通知相关的任务的任务号，类型为 `integer`，外键：Task.id，非空
- 消息标题：通知的标题，类型为 `varchar`，外键：Course.courseid，非空
- 消息内容：通知的具体内容，类型为 `varchar`，最大长度 512，非空
- 时间：该通知的产生时间，类型为 `Date`，非空

### 4.2 实体属性局部 E-R 图

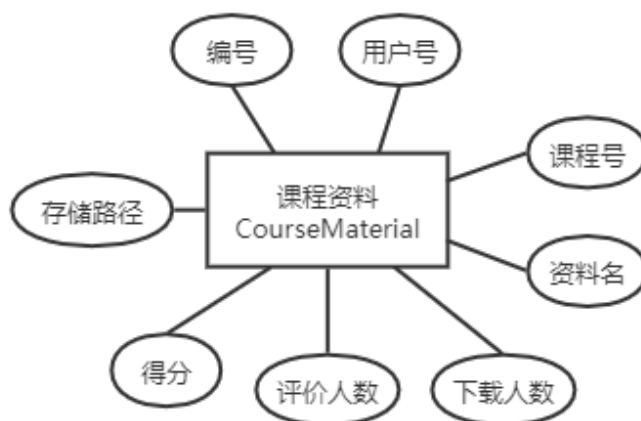
#### 4.2.1 用户局部 E-R 图



#### 4.2.2 课程局部 E-R 图



#### 4.2.3 课程资料局部 E-R 图



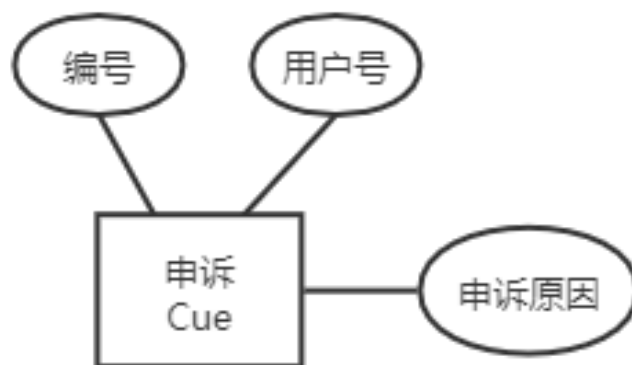
#### 4.2.4 通知局部 E-R 图



#### 4.2.5 任务局部 E-R 图



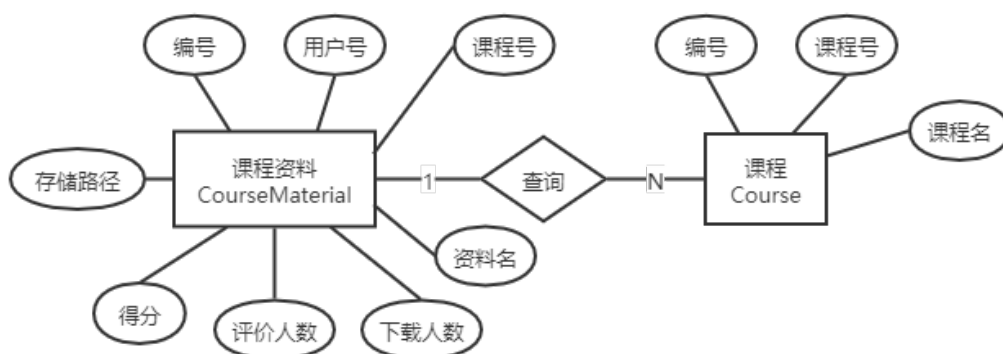
#### 4.2.6 申诉局部 E-R 图



### 4.3 实体局部联系 E-R 图

#### 4.3.1 课程与课程资料局部联系 E-R 图

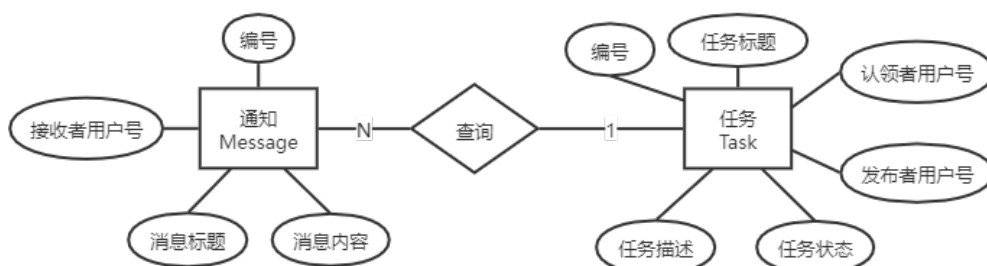
每门课程可以有很多课程资料，而一份课程资料只能对应一门课程，所以为一对多的关系。





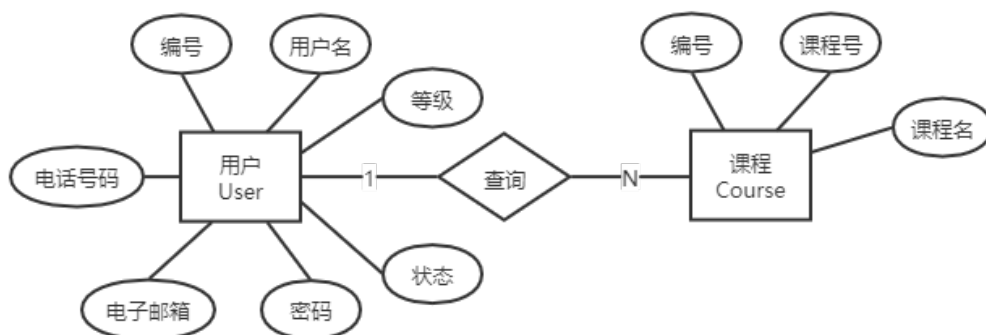
### 4.3.2 任务与通知局部联系 E-R 图

每个任务可以对应多个通知，每当任务状态发生变化就会向用户发送通知，所以为一对多的关系。



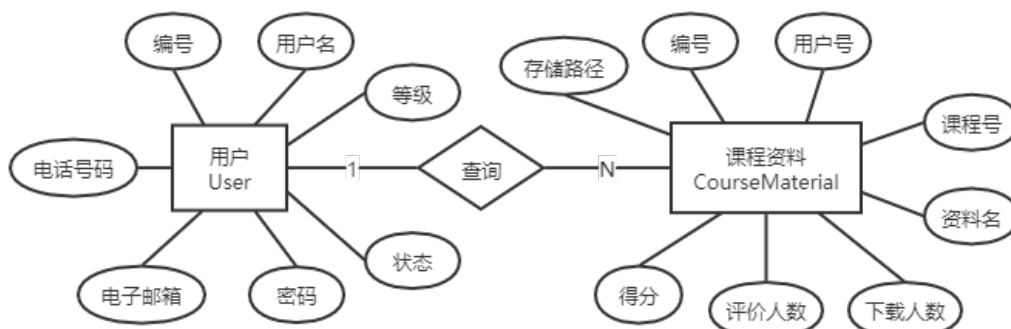
### 4.3.3 用户与课程局部联系 E-R 图

一位用户每次只能查询一门课程，一门课程可以同时被多个用户查询，所以是一对多的关系。



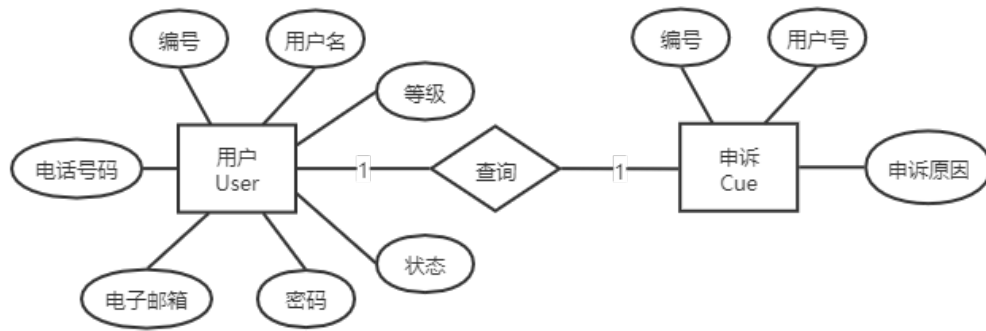
### 4.3.4 用户与课程资料局部联系 E-R 图

一位用户每次只能查询一份资料，一份资料可以同时被多个用户查询，所以是一对多的关系。



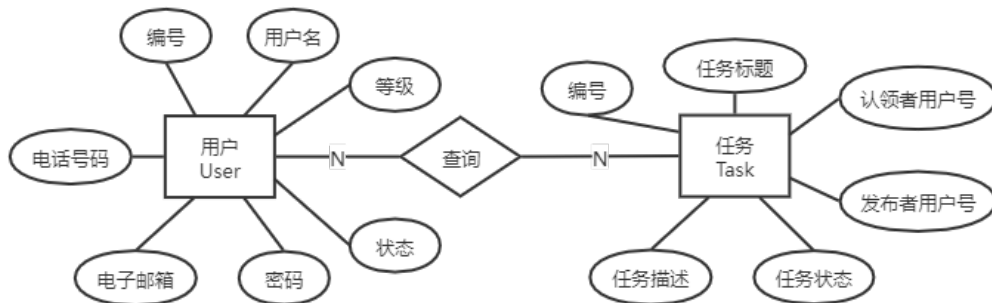
### 4.3.5 用户与申诉局部联系 E-R 图

一位用户只能发送一个申诉，一个申诉也只能对应一位用户，所以是一一对一的关系。



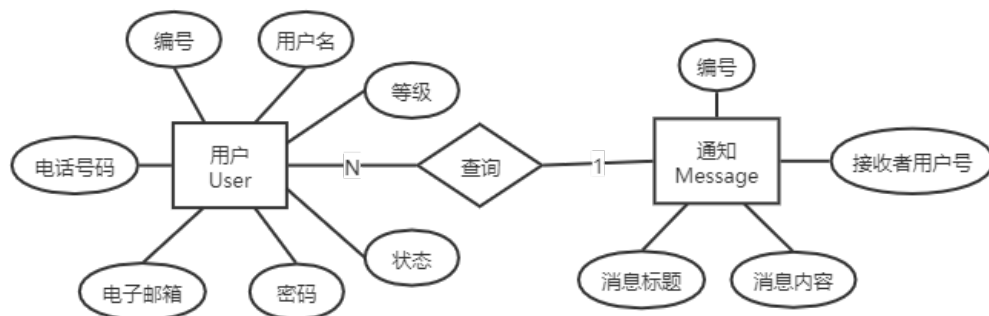
#### 4.3.6 用户与任务局部联系 E-R 图

一位用户可以查询到多个任务，一个任务也可以被多个用户查询到，所以是多对多的关系。

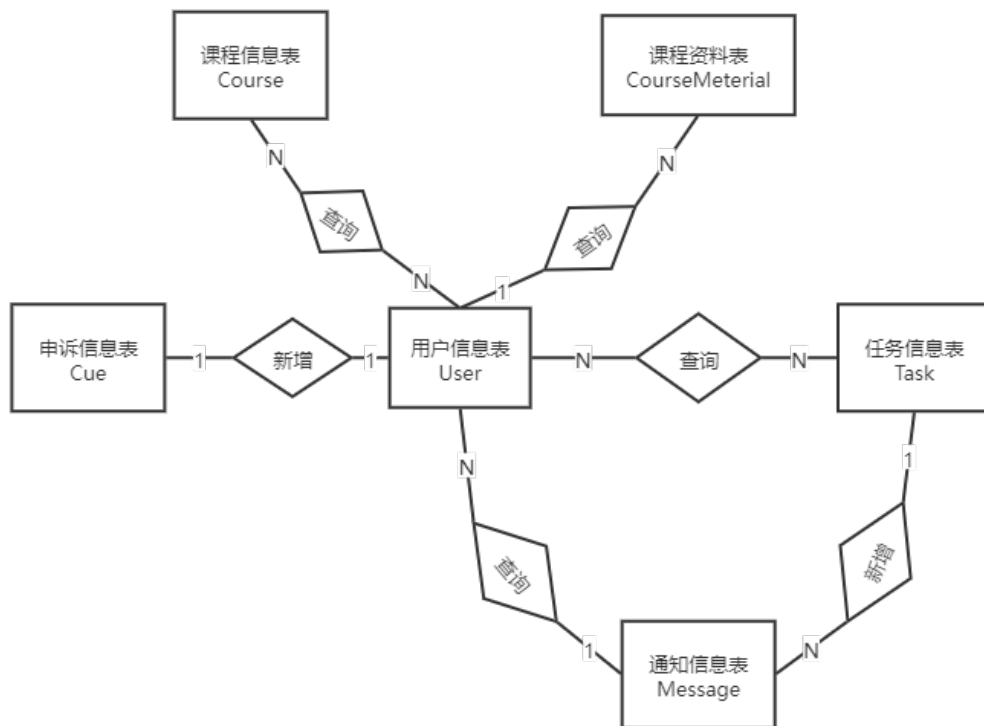


#### 4.3.7 用户与通知局部联系 E-R 图

一位用户可以有多条通知，但一条通知只能发给一位用户，所以是多对一的关系。



## 4.4 全局 E-R 图



## 5 逻辑设计

### 5.1 关系模型

#### 5.1.1 E-R 图向关系模型的转换

E-R 图向关系模型的转换如下：

- User 表 (id, username, password, telnumber, email, degree, status)
- Course 表 (id, courseid, coursename)
- Cue 表 (id, userid, reason)
- CourseMaterial 表 (id, userid, courseid, name, path, score, evaluators, downloads)
- Task 表 (id, publisherid, receiverid, title, description, status)
- Message 表 (id, receiverid, title, content)

#### 5.1.2 数据模型的优化及规范化设计

- User(id, username, password, telnumber, email, degree, status)

id->username, password, telenumner, email, degree, status

id 为主码，该关系模式属于 BCNF 模式，无需再分解。

- Course(courseid, coursename)

courseid->coursename

courseid 为主码，该关系模式属于 BCNF 模式，无需再分解。

- Cue(id, userid, reason)

id->userid, reason

Cue 为主码，该关系模式属于 BCNF 模式，无需再分解。

- CourseMaterial(id, userid, courseid, name, path, score, evaluators, downloads)

id->userid, courseid, name, path, score, evaluators, downloads

id 为主码，该关系模式属于 BCNF 模式，无需再分解。

- Task(id, publisherid, receiverid, title, description, status)

id->publisherid, receiverid, title, description, status

id 为主码，该关系模式属于 BCNF 模式，无需再分解。

- Message(id, receiverid, title, content)

id->receiverid, title, content

id 为主码，该关系模式属于 BCNF 模式，无需再分解。

## 5.2 表结构设计

### 5.2.1 User

| 属性名  | 字段         | 类型      | 长度  | 约束   |
|------|------------|---------|-----|--|
| 编号   | id         | int     |     | primary key auto_increment<br><br>not null<br><br>not null unique<br>not null<br>default 1 |
| 用户名  | username   | varchar | 255 |  |
| 密码   | password   | varchar | 255 |  |
| 电话号码 | telenumner | varchar | 255 |  |
| 邮箱地址 | email      | varchar | 255 |  |
| 等级   | degree     | varchar | 100 |  |
| 用户状态 | status     | int     |     |  |

**5.2.2 Course**

| 属性名 | 字段         | 类型      | 长度  | 约束                    |
|-----|------------|---------|-----|-----------------------|
| 课程号 | courseid   | int     |     | primary key, not null |
| 课程名 | coursename | varchar | 255 | not null unique       |

**5.2.3 Cue**

| 属性名  | 字段     | 类型      | 长度  | 约束                                  |
|------|--------|---------|-----|-------------------------------------|
| 编号   | id     | int     |     | not null primary key auto_increment |
| 用户号  | userid | int     |     | 外键: User.id, not null               |
| 申诉原因 | reason | varchar | 255 | not null                            |

**5.2.4 CourseMaterial**

| 属性名  | 字段         | 类型           | 长度  | 约束                                     |
|------|------------|--------------|-----|--|
| 编号   | id         | int          |     | not null primary key                   |
| 用户号  | userid     | int          |     | foreign key: User.id, not null         |
| 课程号  | courseid   | int          |     | foreign key: Course.courseid, not null |
| 资料名  | name       | varchar      | 255 | not null                               |
| 存储路径 | path       | varchar      | 255 | not null                               |
| 得分   | score      | numeric(4,2) |     | not null default 0                     |
| 评价人数 | evaluators | int          |     | not null default 0                     |
| 下载人数 | downloads  | int          |     | not null default 0                     |

**5.2.5 Task**

| 属性名    | 字段          | 类型      | 长度  | 约束                                  |
|--------|-------------|---------|-----|-------------------------------------|
| 编号     | id          | int     |     | not null primary key auto_increment |
| 发布者用户号 | publisherid | int     |     | foreign key: User.id, not null      |
| 认领者用户号 | receiverid  | int     |     | foreign key: User.id, not null      |
| 任务标题   | title       | varchar | 255 | not null                            |
| 任务描述   | description | varchar | 512 | not null                            |
| 任务状态   | status      | int     |     |                                     |

### 5.2.6 Message

| 属性名    | 字段         | 类型      | 长度  | 约束                                  |
|--------|------------|---------|-----|-------------------------------------|
| 编号     | id         | int     |     | not null primary key auto_increment |
| 接收者用户号 | receiverid | int     |     | foreign key: User.id, not null      |
| 消息标题   | title      | varchar | 255 | not null                            |
| 消息内容   | content    | varchar | 512 | not null                            |

## 6 物理设计

### 6.1 概述

在逻辑设计中，已对项目的后台存储部分进行了规范化的数据库设计。在逻辑设计的基础上，需要根据项目的功能，确定数据库中某些字段由于功能而需要的约束。同时，在实现前端功能与后端数据库的交互时，有一些数据会经常被访问，因此需要在这些数据建立索引。

由于在对操作的约束方面，前端的检查比后端更方便实现、维护和记录，因此这里的物理设计暂时不考虑一些复杂约束和触发器、函数等一系列过程类功能的实现，只对索引部分进行设计。

在项目初期，功能部分还需要进一步细化精化，所以这里只给出索引部分的设计框架和几个简单功能的索引设计，后续有可能再进行补充。由于索引在设计时需要考虑多方面问题，例如对于某些字段是否建立索引、是否是唯一性索引、应该如何尽量设计较合适的索引类型等等。

### 6.2 索引设计

此处仅为初步设计报告，所以这里的索引设计可能后续还需要经过大幅度的修改。这里将列出需要建立索引的表、建立索引的表的字段以及它们的查询类型和索引类型。当然在具体实现时，也许索引类型是由软件自行分配，但这里还是要考虑得更周到一些。如果需要优化则按照自己的设计进行优化。

| 表名           | 字段名     | 查询类型  | 索引类型     |
|--------------|---------|-------|----------|
| 用户信息表 User   | 用户号     | equal | 排序-二分-唯一 |
| 课程信息表 Course | 课程号     | range | 排序       |
| 申诉信息表 Cue    | 编号      | equal | 排序       |
| 课程资料表        | 下载次数/评分 | range | 排序       |
| 任务表          | 编号      | equal | 排序       |

## 7 前端技术

目前市场三大前端主流框架分别是 Angular、React 和 Vue。Vue 之所以被开发者青睐，主要是 Vue 秉承了 Angular 和 React 框架两者的优势，并且 Vue 的代码简洁、上手容易，在市场上也得到大量应用。

本次采用 vue 框架搭建前端。

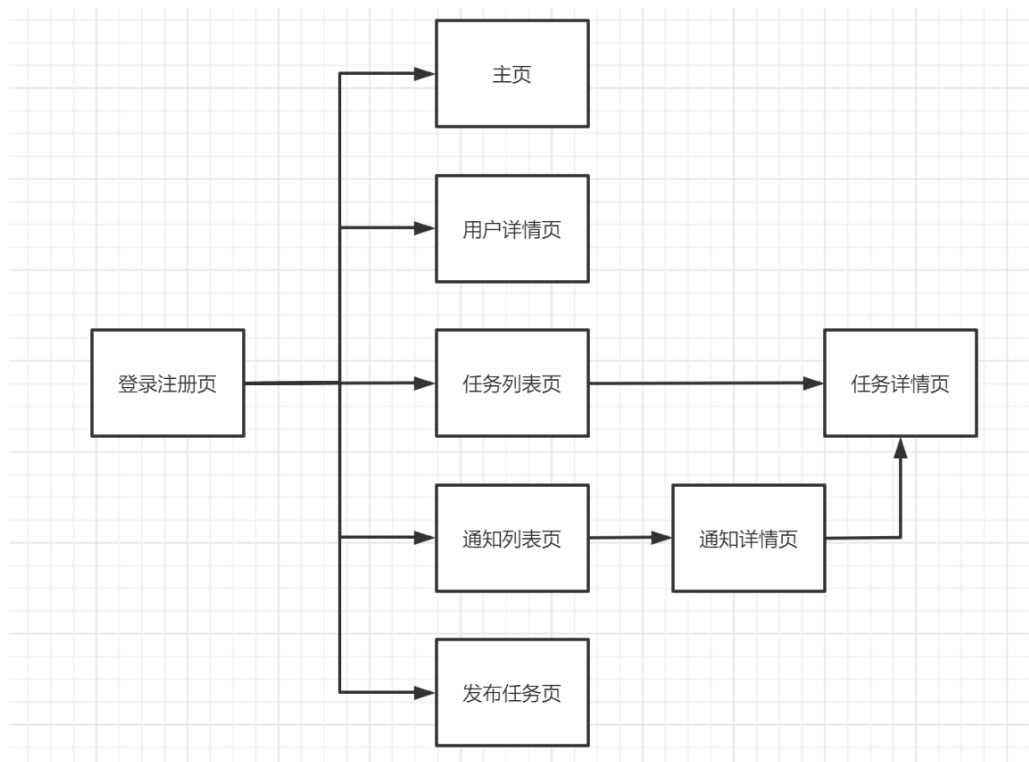
Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

Vue 是一个现代 JavaScript 框架，提供了有用的设施渐进增强。用户可以使用 Vue 增强现有的 HTML，可以使用 Vue 作为 jQuery 等库的临时替代品。可以使用 Vue 编写整个单页应用程序 (SPAs)。这允许创建标记完全由 Vue 管理，可以提高开发人员的经验和性能在处理复杂的应用程序。当需要的时候它还允许利用其他库对客户端路由和状态进行管理。此外，Vue 需要“中间地带”的方法工具客户端路由和状态管理。虽然 Vue 核心团队维护了建议的函数库，但他们并没有直接捆绑到 Vue 里。这样就可以选择一个其他路由/状态管理库，来更好地适应应用程序。除了允许逐步将 Vue 集成到应用程序中，Vue 还提供了一种渐进的方式编写标记。像大多数框架，Vue 通过组件允许创建可重用块标记。大多数时候，Vue 组件是使用一个特殊的 HTML 模板的语法写的。当需要比 HTML 语法允许的更多的控制时，可以编写 JSX 或纯 JavaScript 函数来定义组件。

Vue.js 的核心是一个允许采用简洁的模板语法来声明式地将数据渲染进 DOM 的系统，数据和 DOM 建立了关联后，所有东西都是响应式的。

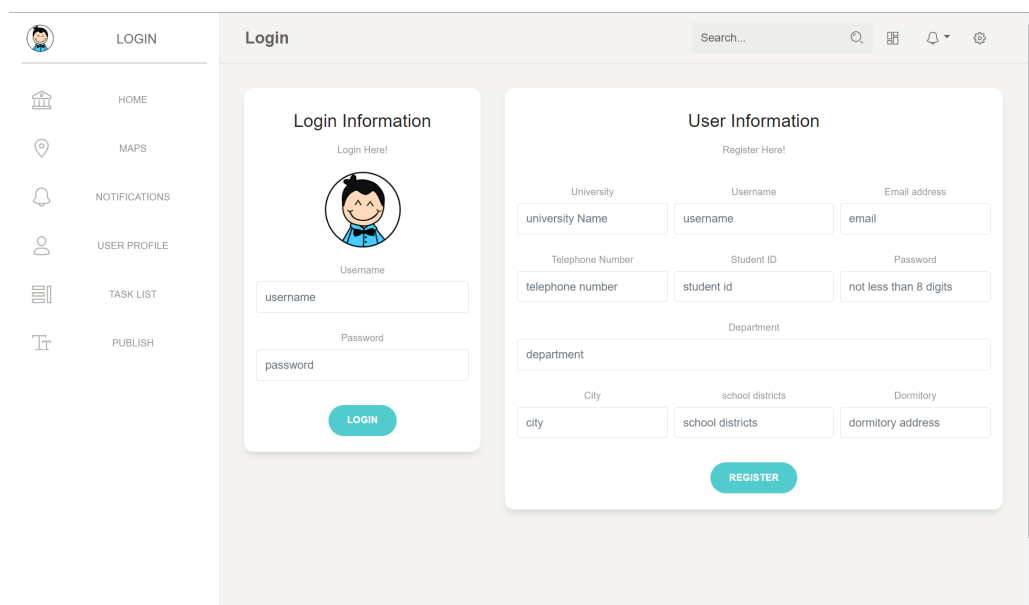
### 7.1 前端结构设计

采用左侧边导航栏的方式，在用户成功登陆后，方便用户直接使用左侧导航栏的全部功能。登陆后默认进入主页面，根据用户下一步在左侧导航栏的选择，利用 vue 的 router 路由跳转到对应功能页面，路由结构如下：



## 7.2 登录注册

登录注册页面展示如下：



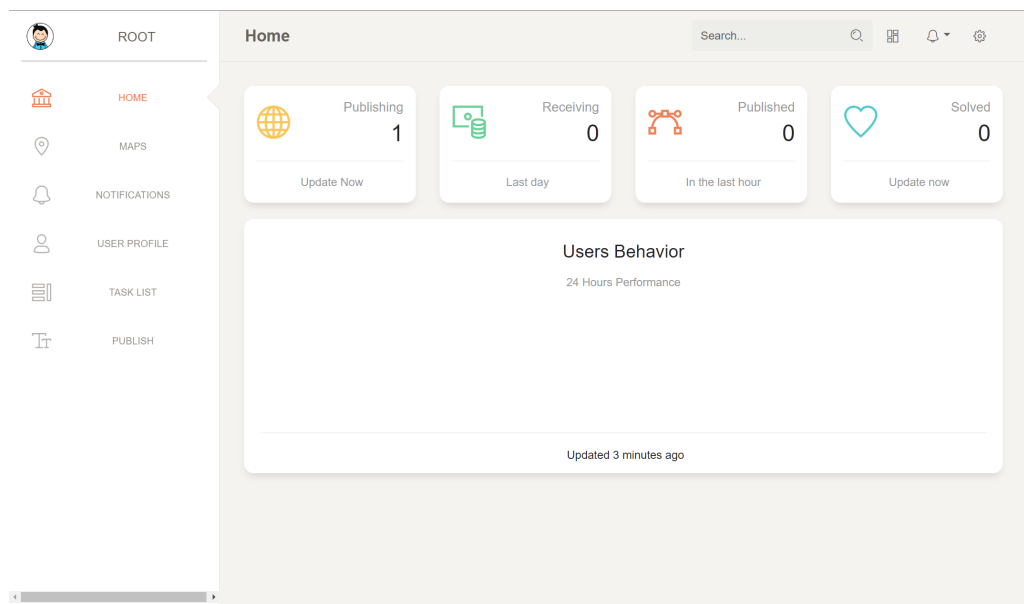
The screenshot shows a web interface for login and registration. On the left is a sidebar with icons and labels: LOGIN, HOME, MAPS, NOTIFICATIONS, USER PROFILE, TASK LIST, and PUBLISH. The main content area is titled 'Login' and contains two forms: 'Login Information' and 'User Information'. The 'Login Information' form has fields for 'Username' (labeled 'username') and 'Password' (labeled 'password'), with a 'LOGIN' button below. The 'User Information' form has fields for 'University' (labeled 'university Name'), 'Email address' (labeled 'email'), 'Telephone Number' (labeled 'telephone number'), 'Student ID' (labeled 'student id'), 'Password' (labeled 'not less than 8 digits'), 'Department' (labeled 'department'), 'City' (labeled 'city'), 'school districts' (labeled 'school districts'), and 'Dormitory' (labeled 'dormitory address'), with a 'REGISTER' button below.

如果仅需要登录功能，则在页面中间填写登录需要的用户名 `username` 和密码 `password` 字段，即可登录，如果密码、用户名不匹配，则弹窗提示登录失败。如果需要注册新账号，则在页面右方填写注册所需的个人信息，个人信息的格式提示已经展现在框中，通过前端检查无报错后，即可提交注册，注册完成后需要再次登录才可以进入网站。

## 7.3 主页

个人主页面展示如下：

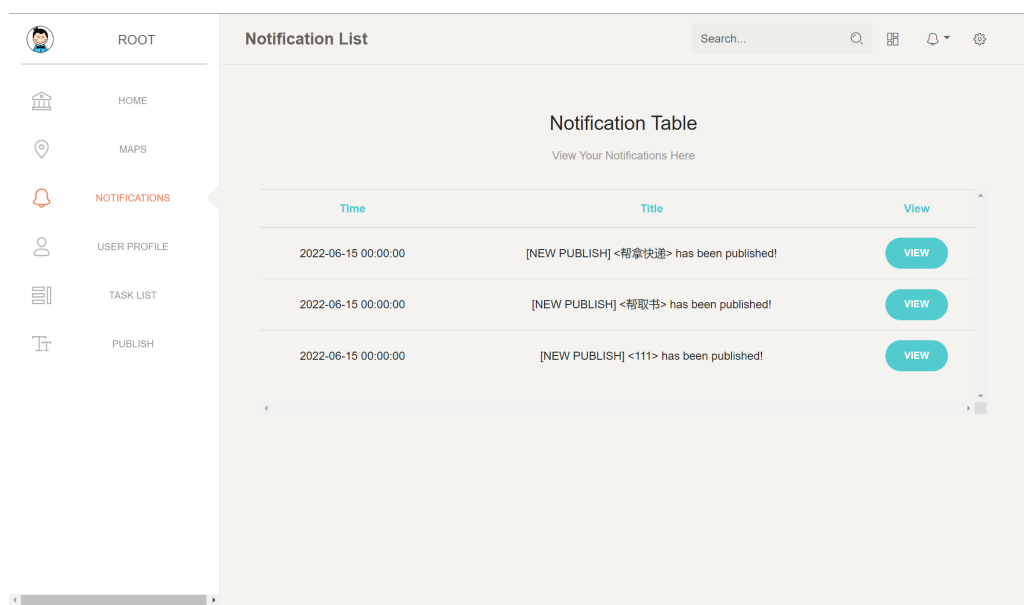




左侧导航栏可以选择“通知栏查看 Notification”、“用户信息查看 User Profile”、“任务栏查看 Task List”、“发布任务 Publish”进入这四个板块。当前板块上方四项展示该用户从注册以来，一共发布了多少项任务 Publishing（包括取消的任务数），接收了多少项任务 Receiving（包括取消的任务数），发布的任务中被完成的任务数量 Published，接收并完成了的任务数量 Solved。下方展示该用户在过去 24 小时内，在本网站的活动状态，包括发布、接收、取消、完成任务等操作。方便用户掌握账户状态，及时处理账户异常操作。

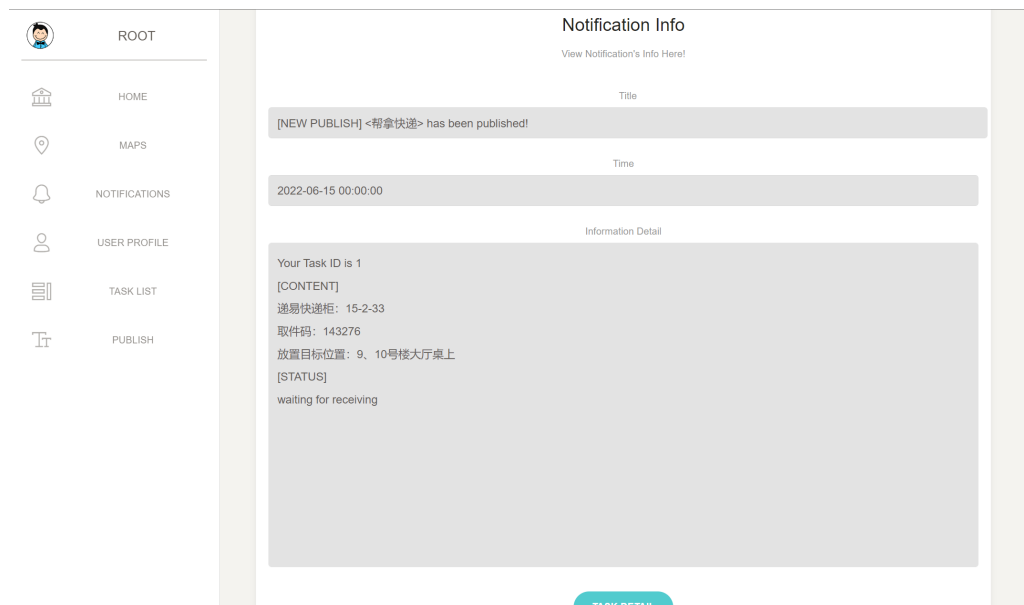
## 7.4 通知

通知栏页面展示如下：



左侧导航栏可以选择“主页面查看 Home”、“用户信息查看 User Profile”、“任务栏查看 Task List”、“发布任务 Publish”进入这四个板块。当前板块用户查看用户所有通知内容，按时间顺序排列，左侧显示该通知发送给用户的时间 Time，中间显示该通知的标题内容 Title，右侧给予一个按钮可以单击查看对应的通知的详细内容、进入通知详情页。

通知详情页展示如下：

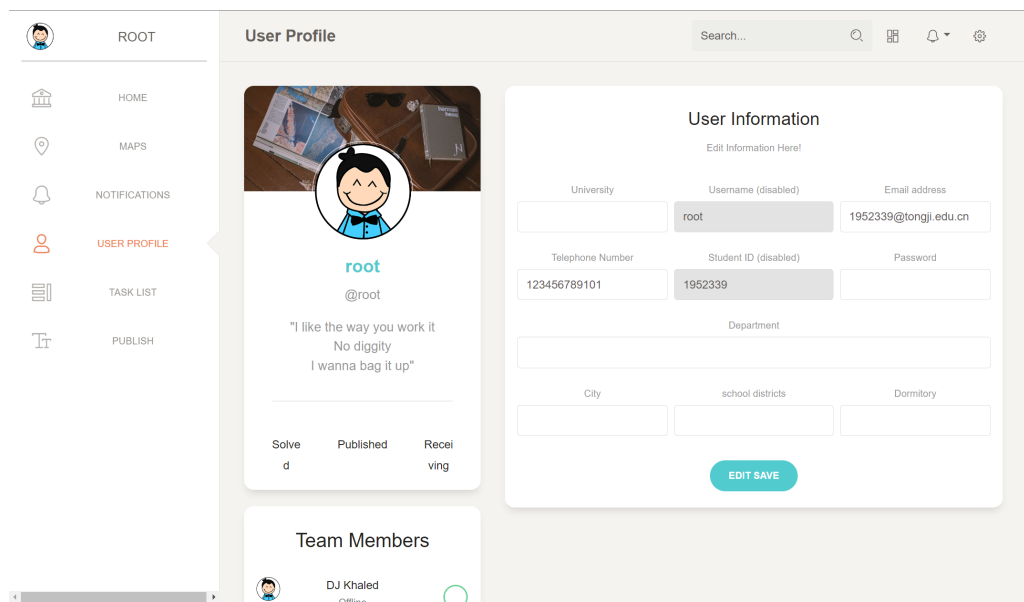


左侧导航栏可以选择“主页面查看 Home”、“通知栏查看 Notification”、“用户信息查看 User Profile”、“任务栏查看 Task List”、“发布任务 Publish”进入这五个板块。当前页面可以查看该通知的详细信息，根据种类不同，通知的标题上会使用 [] 显示该通知的种类：如果是任务发布成功的通知，则表示为 [NEW PUBLISH]；如果是任务接收成功的通知（有人接收该任务，则系统会向发布者和接收者都发送通知），则表示为 [TASK RECEIVED]；如果是删除任务成功的通知，则表示为 [TASK CANCELLED]；如果是任务完成的通知，则表示为 [TASK COMPLETE]；如果是放弃任务的通知（接收者放弃了该任务，则系统会向发布者和接收者都发送通知），则表示为 [TASK GIVEUP]。

通知内容中第一行会给出当前通知关联的任务 ID 号，然后通过 [CONTENT] 给出任务的具体内容，通过 [STATUS] 给出该任务当前的状态，方便查看。下方 VIEW 按钮用于跳转到任务详情页。

## 7.5 用户信息

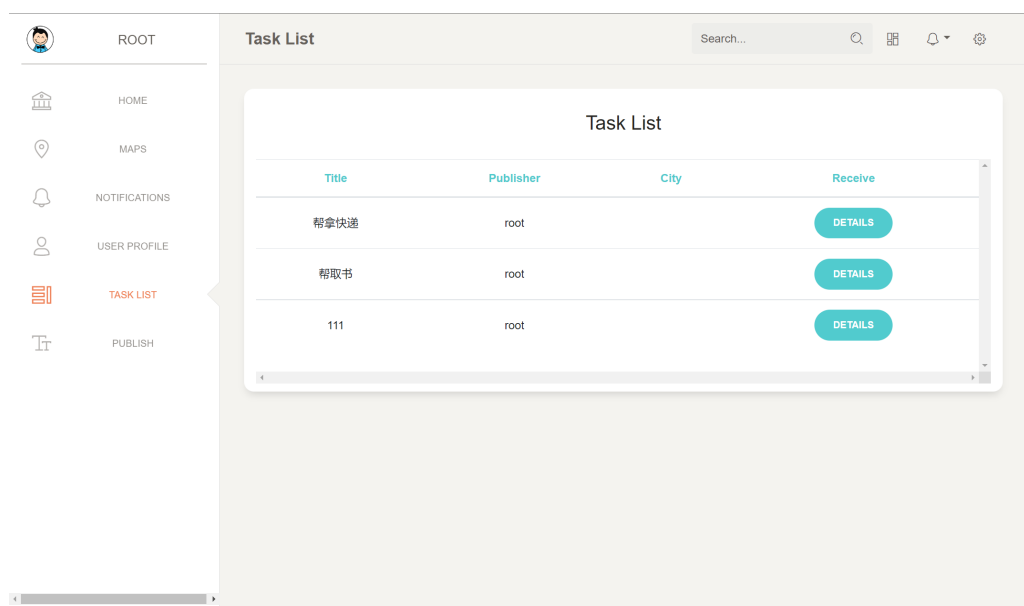
用户信息页面展示如下：



左侧导航栏可以选择“主页面查看 Home”、“通知栏查看 Notification”、“任务栏查看 Task List”、“发布任务 Publish”进入这四个板块。当前板块用于用户查看和修改个人信息，用户的用户名 username 和学号 studentID 一经注册就不再允许修改，其余信息都可以进行修改，需要注意的是，在填写好修改信息后，点击 EDIT SAVE 按钮保存信息时，需要保证密码栏中是填写了密码的。如果要修改密码，就直接填写新密码，如果不修改密码，则填写旧密码用于验证身份信息。下方的 TEAM Members 用于展示系统根据学校等用户信息匹配到的可能的用户。如果需要退出当前登录，则单击本版块左侧蓝色用户名即可退出登录，返回登录注册界面。

## 7.6 任务查看与接收

任务栏页面展示如下：

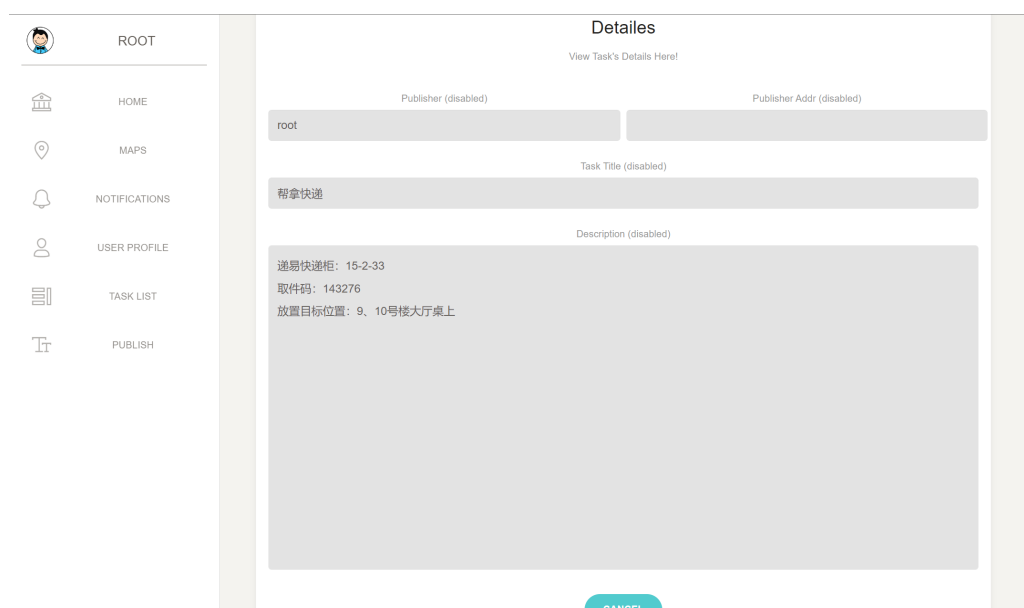


左侧导航栏可以选择“主页面查看 Home”、“通知栏查看 Notification”、“用户信息查看 User Profile”、“任务栏查看 Task List”、“发布任务 Publish”进入这五个板块。当前板块用于用户

查看系统中已经发布、但是没有被接收的任务列表，左侧第一列信息为任务标题 **Title**，第二列为任务发布者 **Publisher**，第三列为任务发布者所在城市（离线任务与发布者所在的城市相关），方便接收任务者筛选，第四列给出查看该任务详情的 **DETAILS** 按钮，单击后进入任务详情页。任务在被接收后将不会再显示在任务栏中，将只能在通知栏的相关通知信息详情页下方点击跳转到任务详情页进行操作。

对于任务发布者来说，在任务详情页下方会出现 **CANCEL** 按钮，用于取消该任务；对于浏览者来说，任务详情页下方会出现 **RECEIVE** 按钮用于接收任务；对于任务接收者来说，任务详情页下方会出现 **GIVEUP** 按钮用于放弃该任务，被放弃的任务将重新回到任务栏中。过程中系统都会向相关用户发布各种通知信息。

任务详情页面展示如下：

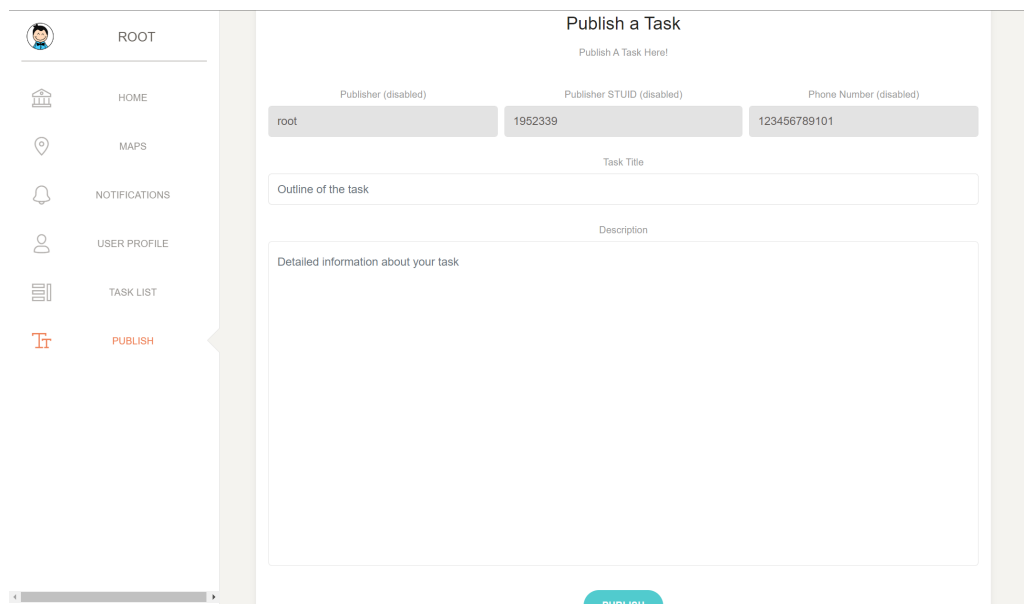


左侧导航栏可以选择“主页面查看 **Home**”、“通知栏查看 **Notification**”、“用户信息查看 **User Profile**”、“任务栏查看 **Task List**”、“发布任务 **Publish**”进入这五个板块。当前页面用于查看任务的详细信息，可以查看任务发布者的用户名，任务发布者所在的地址信息（用于线下任务），任务标题 **Task Title**，任务详情描述（一般包含事件具体内容、完成事件所需相关信息、事件交付方式等内容，由发布者自行编辑）。

对于任务发布者来说，在任务详情页下方会出现 **CANCEL** 按钮，用于取消该任务；对于浏览者来说，任务详情页下方会出现 **RECEIVE** 按钮用于接收任务；对于任务接收者来说，任务详情页下方会出现 **GIVEUP** 按钮用于放弃该任务，被放弃的任务将重新回到任务栏中。过程中系统都会向相关用户发布各种通知信息。

## 7.7 任务发布

任务发布页面展示如下：



左侧导航栏可以选择“主页面查看 Home”、“通知栏查看 Notification”、“用户信息查看 User Profile”、“任务栏查看 Task List”进入这四个板块。当前页面用于编辑任务标题、任务描述、发布任务。第一行是发布者的个人信息，根据登陆账号信息，系统进行了预填写，不允许修改。第二行为任务标题，通常期望简明扼要概括当前想要发布的任务的主要内容，方便其他用户通过任务栏筛选查看。下方是任务详情描述框，一般包含事件具体内容、完成事件所需相关信息、事件交付方式等内容，期望用户仔细认真说清楚需求，方便接收者理解、更高效、更有质量地完成任务。

编辑完毕后，点击下方 PUBLISH 按钮后，任务即发布成功，可以在左侧导航栏点击“任务栏查看 Task List”，查看刚才发布的任务，同时也可以点击“通知栏查看 Notification”，能够看到任务发布成功的系统通知，在该通知的详情页中也可以跳转进入任务详情页。

## 8 后端技术

后端选择 Spring Boot 框架，采用 Java 开发。

Spring Boot 是一个构建在 Spring 框架顶部的项目。它提供了一种简便，快捷的方式来设置，配置和运行基于 Web 的简单应用程序。它是一个 Spring 模块，提供了 RAD(快速应用程序开发) 功能。它用于创建独立的基于 Spring 的应用程序，因为它需要最少的 Spring 配置，因此可以运行。

Spring Boot 是 Spring Framework 和嵌入式服务器的组合。在 Spring Boot 不需要 XML 配置(部署描述符)。它使用约定优于配置软件设计范例，这意味着可以减少开发人员的工作量。本次使用 IntelliJ IDEA 进行开发 Spring Boot Java 后端服务器。

Spring Boot 中使用了依赖项注入方法。它包含强大的数据库事务管理功能。它简化了与其他 Java 框架(如 JPA/Hibernate ORM, Struts 等)的集成。它减少了应用程序的成本和开发时间。与 Spring Boot 框架一起，其他许多 Spring 姐妹项目也有助于构建满足现代业务需求的应用程序。Spring 姐妹项目如下: Spring Data: 它简化了来自关系数据库和 NoSQL 数据库的数据访问。Spring Batch: 它提供了强大的批处理处理。Spring Security: 这是一个安全框架，

可为应用程序提供强大的安全性。Spring Social: 它支持与 LinkedIn 等社交网络集成。Spring Integration: 它是企业集成模式的实现。使用轻量级消息传递和声明性适配器, 它有助于与其他企业应用程序集成。

它创建独立 Spring 应用程序, 这些应用程序可以使用 Java -jar 启动。借助不同的嵌入式 HTTP 服务器 (例如 Tomcat, Jetty 等), 它可以轻松测试 Web 应用程序。我们不需要部署 WAR 文件。它提供了有用的 'starter' POM, 以简化我们的 Maven 配置。它提供了 production-ready 功能, 例如 metrics, health checks 和 externalized configuration。不需要 XML 配置。它提供了一个用于开发和测试 Spring Boot 应用程序的 CLI 工具。它提供了许多插件。它还最大限度地减少了编写多个样板代码 (必须在几乎没有任何改动的情况下将其包含在许多地方), XML 配置和注释的情况。它提高生产力并减少开发时间。

Spring Boot 可以使用应用程序中不会使用的依赖项。这些依赖性增加了应用程序的大小。

Spring Boot 的主要目标是减少开发, 单元测试和集成测试时间。提供有目的的开发方法、避免定义更多的注释配置、避免编写大量导入语句、避免 XML 配置、通过提供或避免上述几点, Spring Boot Framework 减少了开发时间, 开发人员工作量并提高了生产力。

Spring 模块非常适合用于 Web 应用程序开发。可以轻松创建一个独立的 HTTP 应用程序, 该应用程序使用 Tomcat, Jetty 或 Undertow 等嵌入式服务器。可以使用 spring-boot-starter-web 模块快速启动和运行应用程序。SpringApplication 是一个类, 提供了一种方便的方式来引导 Spring 应用程序。可以从 main 方法开始。可以仅通过调用静态 run() 方法来调用应用程序。Spring Boot 使用事件来处理各种任务。它允许创建用于添加侦听器的工厂文件。可以使用 ApplicationListener 键来引用它。Spring Boot 提供了为应用程序启用与管理员相关的功能的功能。它用于远程访问和管理应用程序。可以使用 spring.application.admin.enabled 属性在 Spring Boot 应用程序中启用它。Spring Boot 允许外部化配置, 以便开发者可以在不同环境中使用同一应用程序。该应用程序使用 YAML 文件来外部化配置。Spring Boot 提供了一组丰富的应用程序属性。因此, 开发者可以在项目的属性文件中使用它。该属性文件用于设置诸如 server-port = 8082 等属性。它有助于组织应用程序属性。它提供了一种方便的方法来指定层次结构。它是 JSON 的超集。SpringApplication 类自动支持 YAML。它是属性文件的代替方法。强大的类型安全配置用于管理和验证应用程序的配置。应用程序配置始终是至关重要的任务, 应该是类型安全的。开发者还可以使用此库提供的注释。Spring Boot 对所有内部记录都使用通用记录。默认情况下管理日志记录依赖项。如果不需要自定义, 不应更改日志记录依赖项。Spring Boot 应用程序是 spring 的 Web 应用程序。因此, 默认情况下, 通过所有 HTTP 端点上的基本身份验证, 它是安全的。可以使用一组丰富的端点来开发安全的 Spring Boot 应用程序。

## 8.1 跨域问题

Same Origin Policy, 译为“同源策略”。它是对于客户端脚本 (尤其是 JavaScript) 的重要安全度量标准, 其目的在于防止某个文档或者脚本从多个不同 “origin” (源) 装载。它认为自任何站点装载的信赖内容是不安全的。当被浏览器半信半疑的脚本运行在沙箱时, 它们应该只被允许访问来自同一站点的资源, 而不是那些来自其它站点可能怀有恶意的资源。

具有相同的 Origin, 也即是拥有相同的协议、主机地址以及端口。一旦这三项数据中有一



项不同，那么该资源就将被认为是从不同的 Origin 得来的，进而不被允许访问。

CORS 是一个 W3C 标准，全称是“跨域资源共享”（Cross-origin resource sharing），允许浏览器向跨源服务器，发出 XMLHttpRequest 请求，从而克服了 AJAX 只能同源使用的限制。它通过服务器增加一个特殊的 Header[Access-Control-Allow-Origin] 来告诉客户端跨域的限制，如果浏览器支持 CORS、并且判断 Origin 通过的话，就会允许 XMLHttpRequest 发起跨域请求。

具体添加代码如下：

```
@Configuration
public class CorsConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping(pathPattern: "**")
            .allowedOriginPatterns("**")
            .allowedMethods("GET", "HEAD", "POST", "PUT", "DELETE", "OPTIONS")
            .allowCredentials(true)
            .maxAge(3600)
            .allowedHeaders("**");
    }
}
```

## 8.2 与数据库的连接

数据库选择 MYSQL，使用 Springboot 调用封装好的 Hibernate 能够与 MYSQL 进行很好的连接。Hibernate 对简单的数据库查询进行了一定的封装，包括查询表中全部信息、添加某一元素等基础操作，但是本次“帮帮我”校园系统项目中，对数据库内容的查询大多具有特色限制，导致需要在 SpringBoot 中重写一部分查询操作，并且涉及到多表联合查询等，部分自定义数据库操作代码如下：

```
public interface UserRepository extends JpaRepository<User,Integer> {

    2 usages
    @Query("select u from User u where u.Username=?1")
    User findByUsername(String username);
    1 usage
    @Transactional
    @Modifying
    @Query("update User u set u.publishing=:publishing where u.id=:id")
    void updatePublishing(@Param("publishing")Integer publishing, @Param("id")Integer id);
    1 usage
    @Transactional
    @Modifying
    @Query("update User u set u.receiving=:receiving where u.id=:id")
    void updateReceiving(@Param("receiving")Integer receiving, @Param("id")Integer id);
    1 usage
    @Transactional
    @Modifying
    @Query("update User u set u.published=:published where u.id=:id")
    void updatePublished(@Param("published")Integer published, @Param("id")Integer id);
    1 usage
    @Transactional
    @Modifying
    @Query("update User u set u.solved=:solved where u.id=:id")
    void updateSolved(@Param("solved")Integer solved, @Param("id")Integer id);
}
```

```
public interface TaskRepository extends JpaRepository<Task,Integer> {  
    2 usages  
    @Query("select new com.springboot.helpme.entity.TaskItem(u,t) from User u, " +  
        "Task t where t.publisherid=u.id and t.status='waiting for receiving'")  
    List<TaskItem> listAll();  
  
    2 usages  
    @Query("select new com.springboot.helpme.entity.TaskItem(u,t) from User u, " +  
        "Task t where t.publisherid=u.id and t.id=?1 " +  
        "and t.status='waiting for receiving'")  
    TaskItem itemById(Integer taskId);  
}
```

## 9 测试

### 9.1 数据库测试

对数据库进行了正确性测试，关联测试，以及性能测试。

正确性测试结果表明数据库能够正常工作，并且能够就错误的存储进行拒绝，能够从数据库中取出正确数据用于前端渲染。

关联测试结果表明数据库的外码依赖、index 关系都正常运行，删除某项时能够正确删除关联的其余项，并且能够快速查询关联项。

使用 sysbench 进行性能测试，部分测试结果如下：

```
####以下是每5秒返回一次的结果，统计的指标包括：  
#### 线程数、tps(每秒事务数)、qps(每秒查询数)、  
#### 每秒的读/写/其它次数、延迟、每秒错误数、每秒重连次数  
[ 5s ] thds: 4 tps: 130.16 qps: 2606.30 (r/w/o: 1824.51/520.66/261.13) lat (ms,95%): 104.84 err/s: 0.00 reconn/s:  
[ 10s ] thds: 4 tps: 126.74 qps: 2539.17 (r/w/o: 1778.17/507.52/253.47) lat (ms,95%): 108.68 err/s: 0.00 reconn/s:  
[ 15s ] thds: 4 tps: 136.54 qps: 2736.34 (r/w/o: 1915.25/548.01/273.07) lat (ms,95%): 102.97 err/s: 0.00 reconn/s:  
[ 20s ] thds: 4 tps: 107.44 qps: 2148.65 (r/w/o: 1505.60/428.17/214.89) lat (ms,95%): 132.49 err/s: 0.00 reconn/s:  
  
SQL statistics:  
  queries performed:  
    read:      35098  # 执行的读操作数量  
    write:     10028  # 执行的写操作数量  
    other:      5014  # 执行的其它操作数量  
    total:     50140  
  transactions: 2507  (124.29 per sec.) # 执行事务的平均速率  
  queries:      50140 (2485.82 per sec.) # 平均每秒能执行多少次查询  
  ignored errors: 0    (0.00 per sec.)  
  reconnects:   0    (0.00 per sec.)  
  
General statistics:  
  total time:      20.1694s # 总消耗时间  
  total number of events: 2507 # 总请求数量(读、写、其它)
```

数据库的性能尚可，能够满足本次项目使用需求。

### 9.2 系统测试

前后端、数据库连接良好，尝试了未超过 10 人在网站中进行浏览、收发任务等行为，没有出现异常，系统的正确性有一定保障，也能够对一些错误操作进行修正。

系统前端的引导效果良好，能够有效避免新用户的很多错误操作，提升用户使用体验。



## 10 总结

本次对于“帮帮我”校园互助系统做出了数据库的前期设计包含需求分析以及可行性分析、概念设计、逻辑设计，加深了对于数据库模式、数据库关系等课堂内容的理解。

学习并使用了 VUE 框架进行前端网站搭建，了解了 VUE-cli 脚手架、axios 前后端通信方式、xmlhttps 表单提交、js 动态效果等前端知识，并且能够熟练应用这些 web 内容。

学习并使用了 SpringBoot 这一后端框架进行后端服务器构建，了解了 Hibernate 与数据库的操作方式、Maven 的项目管理方式，学会基础地使用这些成熟的框架进行项目构建，使得项目开发效率得到了有效提升，改善了开发过程体验，并且能够很好的锻炼后端系统的思想。

“帮帮我”校园互助系统，旨在提升同学们互帮互助的过程体验，引导同学们乐于助人的日常行为习惯，节约同学们的时间、方便生活，让校区内同学们的生活品质得到提高。增进大家的交流，促进人与人之间友善、互助的友好品质。

本次“帮帮我”校园互助系统的编写，极大地锻炼了我的全栈开发能力，提高了自学能力与实践能力，希望未来能够做出更完善的系统，更好的服务于同学们的生活。