

# 软工导论学习平台开发 —— 一次成功的软件工程实践

## 过程分析

### 一、过程模型

我们的项目选用敏捷开发方法，在 Scrum 框架的基础上进行改进，具体来说有以下几点：

#### 1. 迭代周期（Sprint）管理

我们的项目设定了明确的 Sprint 周期，根据功能难度不同和团队的可用时间划分为 1-4 周。

项目从第 7 周开始进行：

- 7-8 周：确定技术栈，根据要做的功能确定前端页面原型和后端 API 接口，并搭建前后端的大致框架。
- 8-10 周：完成基本功能。
- 10-12 周：在基本功能上细化，完成附加功能。

#### 2. 需求管理

- 我们所有人首先在一起沟通需求，由于自然语言的不精确性，我们会用简单的原型软件先画出前端页面的布局、跳转逻辑等，然后定好后端接口的规范（JSON）。通过这个方法，我们进行了合理的需求管理。
- 在优先级排序上：我们根据功能价值和必要程度对需求进行排序，分为基础部分和细化/提升部分。

#### 3. 任务分解与细化

- 我们将需求（用户故事）分解为具体的页面，每个页面包含前端要做的组件和逻辑以及后端要实现的接口，确保每个任务都可执行和可测量，但是更细的粒度上不去约束成员，可实现即可。
- 团队协作上，根据团队成员的技能和负载情况合理分配任务，促进协作和知识共享，并且用简便的方法管理任务（后面会细讲）。

#### 4. 日常站会

- 我们每周会开 3 次会，分别在周二、周四、周末，除了第一次会议在四十分钟左右，其他都在 10-20 分钟之间，确保会议简洁高效。
- 每次会议每个成员回答三个问题：
  1. 上个周期完成了什么？
  2. 这个周期计划做什么？
  3. 遇到了哪些障碍？

- 组长会记录并跟踪会议中提到的问题，及时解决阻碍进展的障碍。

## 5. 持续集成/持续交付/项目管理

- 我们前端采用 Git 作为项目版本管理工具，所有代码上传到 GitHub 的私有仓库中，所有成员可及时看到。数据库放在服务器上，交由专门的同学进行维护。更改、合并等活动由组长来监控。
- 前端采用 npm 作为包管理器，后端用 Maven 进行管理，在不同环境下能同样运行，确保了开发、测试和生产环境的一致性。
- 测试上，我们的单元测试采用 JUnit，前后端模块测试采用 Apifox（即前端的每个页面和后端的每个接口）。
- 我们维护详细的 CI/CD 流程文档，向不熟悉项目流程的团队成员讲解这些技术，确保每个人都理解和遵循 CI/CD 实践。

## 6. 测试驱动开发（TDD）与行为驱动开发（BDD）

- 后端测试使用 Apifox，测试后端返回接口与 Apifox 中的接口约定，前端页面利用 Apifox 的 mock 功能或者直接利用后端接口测试，可直观检查是否符合功能需求。
- 单元测试采用 JUnit 或打印日志等方法。

## 7. 迭代评审与回顾

在完成基本功能后，我们评审了已经完成的功能，并且在此基础上讨论需要做哪些细化和扩展。我们讨论后完成了以下功能：

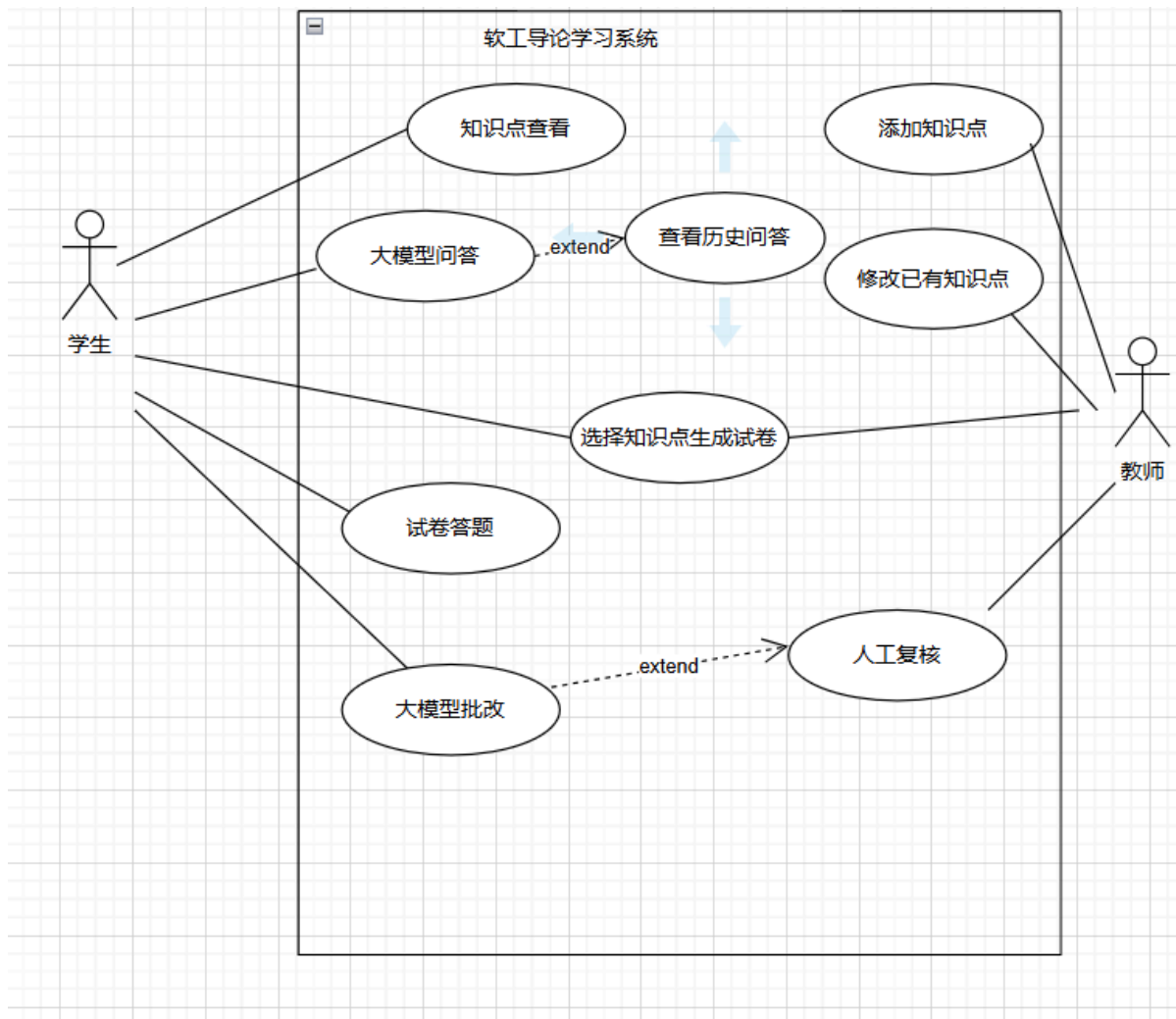
- 选择题与判断题支持
- 知识点修改
- 多用户登录

# 二、需求分析

---

## 1. 系统概述

本系统是一个基于知识图谱的软件项目管理学习系统，旨在帮助学生和教师高效学习、管理与评估软件项目管理知识点。通过系统化的知识点关联和大模型辅助功能，实现个性化学习与高效教学。



## 2. 目标用户

### 2.1 学生

- 查阅、学习知识点及关联内容。
- 通过练习、试卷自我评估学习效果。
- 提问知识点相关问题，获取智能解答。

### 2.2 教师

- 管理知识点与习题内容。
- 生成并发布试卷。
- 对试卷进行人工复核与批改，提升批改准确性。

## 3. 功能性需求

### 3.1 用户角色及权限

#### 学生

- 登录系统后，使用以下功能：
  - 查阅知识点
  - 提问（大模型问答）
  - 生成试卷
  - 提交答案
  - 查看试卷历史记录

## 教师

- 登录系统后，使用以下功能：
  - 管理知识点
  - 生成试卷
  - 批改试卷
  - 复核批改结果

## 3.2 知识点模块

### 功能描述

- **知识点搜索与展示：**用户通过关键词查找知识点并查看详细内容及关联信息。
- **知识点关联：**知识点之间存在关联关系，如前置知识点、后置知识点及关联知识点。

### 需求功能

1. 用户输入关键词，系统展示匹配的知识点列表。
2. 点击知识点，查看详细内容，包括：
  - 知识点名称、标签（难点/重点等）。
  - 认知维度（记忆、理解等）。
  - 详细内容。
  - 前置知识点、后置知识点及关联知识点。
3. 教师可添加、修改或删除知识点内容。

## 3.3 试卷模块

### 功能描述

试卷模块允许用户基于选定知识点生成试卷，并完成练习与批改。系统支持自动批改和历史记录查看。

## 需求功能

### 1. 试卷生成：

- 学生和教师均可根据所选知识点生成试卷。
- 系统根据所选知识点自动提取关联的习题生成试卷。
- 试卷生成后提供预览功能。

### 2. 试卷答题：

- 学生完成生成的试卷，并提交答案。
- 提交后系统利用大模型自动批改，并给出反馈与分数。

### 3. 试卷批改：

- 系统对学生提交的试卷答案进行自动批改。
- 教师可对自动批改的结果进行人工复核和调整。

### 4. 试卷历史记录查看：

- 学生可查看自己提交的试卷、分数及批改反馈。
- 教师可查看所有学生的试卷记录，包括提交时间、批改结果和复核情况。

## 3.4 大模型问答模块

### 功能描述

大模型问答模块为用户提供基于知识点的问答功能，帮助用户快速解决疑问。

### 需求功能

1. 用户输入问题，系统将问题提交给大模型处理。
2. 系统根据知识点内容返回答案。
3. 系统记录用户提问历史，用户可随时查看。

## 4. 非功能性需求

### 4.1 性能需求

- 知识点搜索应在 **1秒** 内返回结果。
- 自动批改的响应时间应控制在 **10秒** 内。

### 4.2 安全性需求

- 用户数据（包括登录信息、试卷答案和历史记录）需加密存储。
- 防止非法用户访问敏感数据，支持权限验证。

## 4.3 可用性需求

- 界面简洁、易于操作，适合学生和教师。

## 5. 界面需求

### 5.1 知识点页面

- 包括搜索框、知识点列表及知识点详情页面。

### 5.2 试卷页面

- **生成试卷页面**：知识点选择及预览。
- **答题页面**：展示试卷内容及提交功能。
- **批改页面**：显示答案及批改反馈。

### 5.3 问答页面

- 提问输入框、提问按钮、回答展示区及历史记录查看。

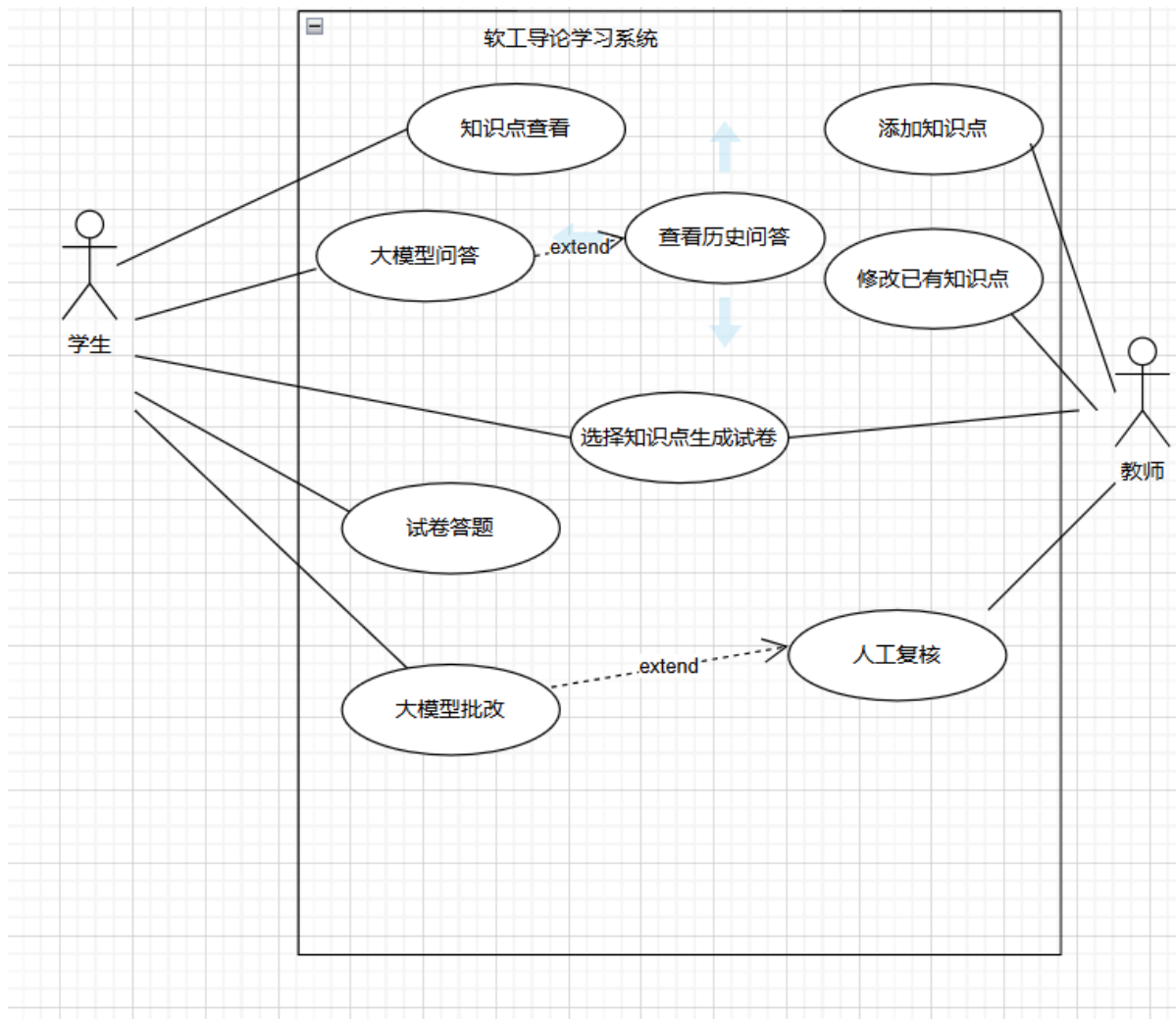
## 三、软件建模与软件设计

---

- 我们在建模上采用了 UML 方法，取得了较好的成效。
- **页面设计工具**：在前期采用 Axure 和墨刀等软件先画出简单的页面原型，使小组中其他人能直观看得到并且进行评价，提出意见。
- **接口设计工具**：使用 Apifox 进行接口约定，所有人都处于团队项目中，可以对接口的设计进行讨论。

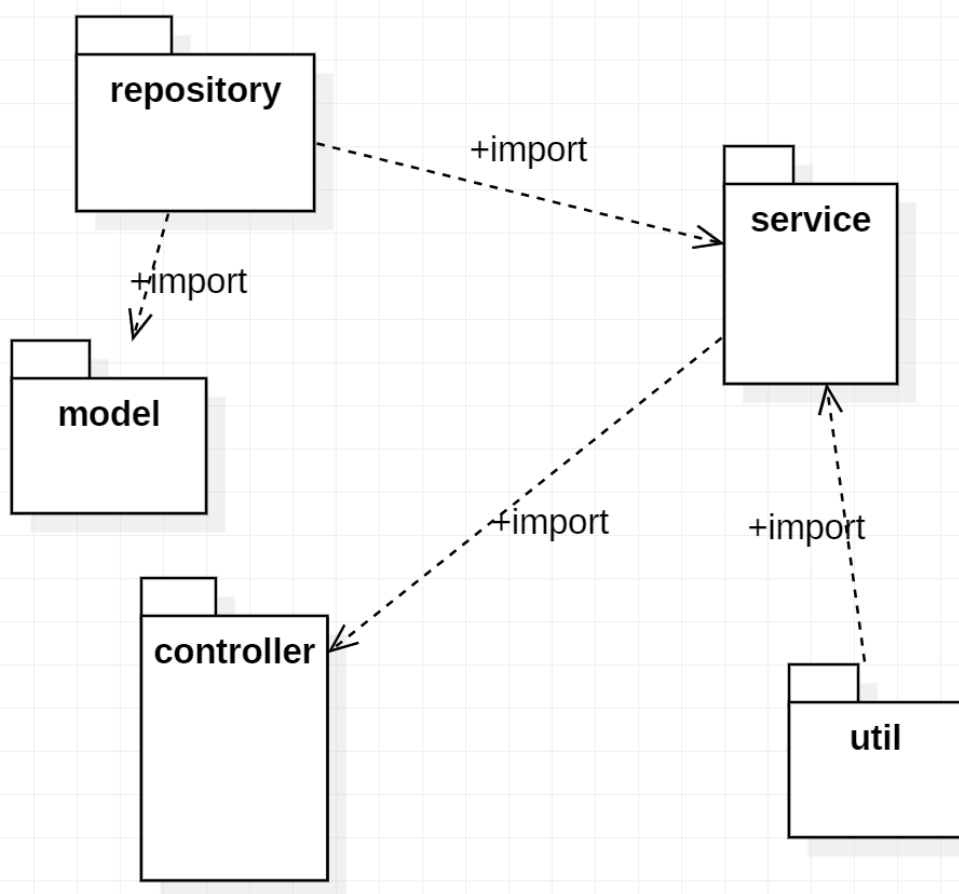
## 用例图

在我们的应用中，功能间并没有太多的依赖关系，所以主要用用例图描述：



## 后端模块包图

以下包图可描述我们后端各个包之间的关系：



## 后端模块的建模与设计

我们后端实现的每个接口都保持了大致相同的结构，每个接口都含有对应的 `controller`、`model`、`service`、`repository` 类，并且依赖于 `util` 包中的工具类。下面仅介绍较为核心的模块

## 试卷生成及用户作答模块

### 1. generateExam 方法

`generateExam` 方法的核心功能是根据传入的知识点 ID 列表，自动生成一份试卷，并将试卷信息存储在 Neo4j 数据库中。

- **功能描述：**根据传入的知识点 ID 列表，生成试卷标题，并从数据库为每个知识点选取最多 5 道题目。生成的试卷包含题目内容、标准答案、题目类型等信息，最终保存到数据库，并返回生成的试卷信息。
- **参数：**
  - `JSONArray knowledgeIds`: 包含多个知识点 ID 的数组。
  - `String username`: 当前用户的用户名。
- **返回值：**
  - `JSONObject`: 返回包含试卷 ID、标题、题目列表及题目 ID 列表的 JSON 对象。



- **实现过程：**
  1. **生成试卷标题：**根据传入的知识点名称构建试卷标题。
  2. **从数据库获取题目：**使用 `ExerciseRepository` 获取每个知识点下的题目列表。
  3. **去重并生成试卷：**避免重复题目，通过集合 `usedQuestionIds` 确保每个题目只添加一次。
  4. **存储试卷数据：**将生成的试卷信息存储到 Neo4j 数据库。

## 2. saveExam 方法

`saveExam` 方法用于保存已生成的试卷，标记试卷为已保存状态。方法会更新试卷节点的 `flag` 属性，表明试卷已完成保存。

- **功能描述：**通过试卷 ID 找到对应的试卷并将其标记为已保存，同时清理数据库中未保存的试卷。
- **参数：**
  - `String examId`: 试卷的唯一标识符。
- **返回值：**
  - `JSONObject`: 返回保存结果，包含试卷 ID 和标题。

## 3. getExam 方法

`getExam` 方法用于根据试卷 ID 和用户名查询试卷信息。该方法支持根据试卷 ID 查询特定试卷信息，也支持查询某个用户的所有试卷。

- **功能描述：**根据试卷 ID 或用户名查询对应的试卷信息，包括题目列表和试卷 ID。
- **参数：**
  - `String examId`: 试卷 ID，若为空则查询所有试卷。
  - `String username`: 用户名。
- **返回值：**
  - `JSONArray`: 返回试卷列表，包括试卷 ID、标题和题目列表等信息。

## 4. submitExam 方法

`submitExam` 方法用于处理学生提交的试卷答案，并将用户答案与试卷进行关联。

- **功能描述：**将学生的作答数据保存到数据库中，并将其与试卷节点关联。
- **参数：**
  - `String examId`: 试卷 ID。
  - `JSONArray answers`: 学生提交的答案列表。
  - `String username`: 用户名。
- **返回值：**
  - `boolean`: 返回是否提交成功。

# 接口设计

## 1. GET /api/knowledge/exercises

请求参数:

```
export interface Request {  
    /**  
     * 知识点 ID 编号（可选）  
     */  
    id?: string;  
    [property: string]: any;  
}
```

返回响应:

```
export interface Response {  
    /**  
     * http状态码, 200为成功, 其他为不成功  
     */  
    code: number;  
  
    /**  
     * 题目列表  
     */  
    quesList: QuesList[];  
  
    /**  
     * 成功与否, 状态  
     */  
    success: boolean;  
  
    [property: string]: any;  
}  
  
export interface QuesList {  
    /**  
     * 标准答案  
     */  
    standardAnswer: string;  
  
    /**  
     * 题目内容  
     */  
    titleContent: string;  
  
    [property: string]: any;  
}
```

## 功能描述：

- 获取知识点相关的题目列表。可以通过 `id` 参数筛选特定知识点的题目。

## 2. POST /api/exam/generate

### 请求参数：

```
export interface Request {  
    /**  
     * 要包含的知识点ID列表  
     */  
    knowledgeIds: string[];  
  
    /**  
     * 用户名  
     */  
    username: string;  
  
    [property: string]: any;  
}
```

### 返回响应：

```
export interface Response {  
    /**  
     * http状态码，200为成功，其他为不成功  
     */  
    code: string;  
  
    /**  
     * 试卷ID  
     */  
    examId: string;  
  
    /**  
     * 试卷标题  
     */  
    examTitle: string;  
  
    /**  
     * 题目列表  
     */  
    quesList: QuesList[];  
  
    /**  
     * 成功与否，true为成功，false为不成功  
     */  
    success: string;  
  
    [property: string]: any;  
}
```

```
export interface QuesList {  
    /**  
     * 标准答案  
     */  
    standardAnswer: string;  
  
    /**  
     * 题目内容  
     */  
    titleContent: string;  
  
    [property: string]: any;  
}
```

### 功能描述：

- 用户通过传递知识点 ID 列表生成包含相关习题的试卷。

## 3. GET /api/exam/save

### 请求参数：

```
export interface Request {  
    /**  
     * 试卷ID（可选）  
     */  
    id?: string;  
  
    [property: string]: any;  
}
```

### 返回响应：

```
export interface Response {  
    /**  
     * http状态码，200为成功，其他为不成功  
     */  
    code: string;  
  
    /**  
     * 试卷ID  
     */  
    examId: string;  
  
    /**  
     * 试卷标题  
     */  
    examTitle: string;  
  
    /**  
     * 题目列表  
     */  
    quesList: QuesList[];
```

```

    /**
     * 成功与否，true为成功，false为不成功
     */
    success: string;

    [property: string]: any;
}

export interface QuesList {
    /**
     * 标准答案
     */
    standardAnswer: string;

    /**
     * 题目内容
     */
    titleContent: string;

    [property: string]: any;
}

```

### 功能描述：

- 保存试卷数据，完成试卷保存后的状态标记。

## 4. POST /api/exam/submit

### 请求参数：

```

export interface Request {
    /**
     * 用户答案列表
     */
    answers: string[];

    /**
     * 试卷ID
     */
    examId: string;

    /**
     * 用户名
     */
    username: string;
}

```

## 返回响应:

```
export interface Response {  
    /**  
     * http状态码, 200为成功, 其他为不成功  
     */  
    code: number;  
  
    [property: string]: any;  
}
```

## 功能描述:

- 提交用户的试卷答案, 并关联到试卷数据。

## 5. GET /api/exam/getExam

### 请求参数:

```
export interface Request {  
    /**  
     * 试卷ID (可选)  
     */  
    examId?: string;  
  
    /**  
     * 用户名 (可选)  
     */  
    username?: string;  
  
    [property: string]: any;  
}
```

## 返回响应:

```
export interface Response {  
    /**  
     * http状态码, 200为成功, 其他为不成功  
     */  
    code: string;  
  
    /**  
     * 试卷ID  
     */  
    examId: string;  
  
    /**  
     * 试卷标题  
     */  
    examTitle: string;  
  
    /**  
     * 题目id列表  
     */  
}
```

```

    */
    quesIds: string[];

    /**
     * 题目详细列表
     */
    quesList: QuesList[];

    /**
     * 成功与否, true为成功, false为不成功
     */
    success: string;

    [property: string]: any;
}

export interface QuesList {
    /**
     * 标准答案
     */
    standardAnswer: string;

    /**
     * 题目内容
     */
    titleContent: string;

    /**
     * 题目类型
     */
    type: string;

    [property: string]: any;
}

```

### 功能描述：

- 获取试卷内容，包括试卷的标题、题目列表和题目 ID 列表。

## 6. GET /api/exam/getAnswer

### 请求参数：

```

export interface Request {
    /**
     * 试卷ID（可选）
     */
    examId?: string;

    /**
     * 用户名（可选）
     */
    username?: string;
}

```

```
[property: string]: any;
}
```

## 返回响应:

```
export interface Response {
  /**
   * http状态码, 200为成功, 其他为不成功
   */
  code: number;

  /**
   * 用户作答数据
   */
  userAnswers: UserAnswer[];

  [property: string]: any;
}

export interface UserAnswer {
  /**
   * 答案列表
   */
  answers: string[];

  /**
   * 用户作答 ID
   */
  id: string;
}
```

## 功能描述:

- 获取用户作答数据, 返回用户的作答答案。

## 7. DELETE /api/exam/del

### 请求参数:

```
export interface Request {
  /**
   * 试卷ID
   */
  examId: string;

  [property: string]: any;
}
```

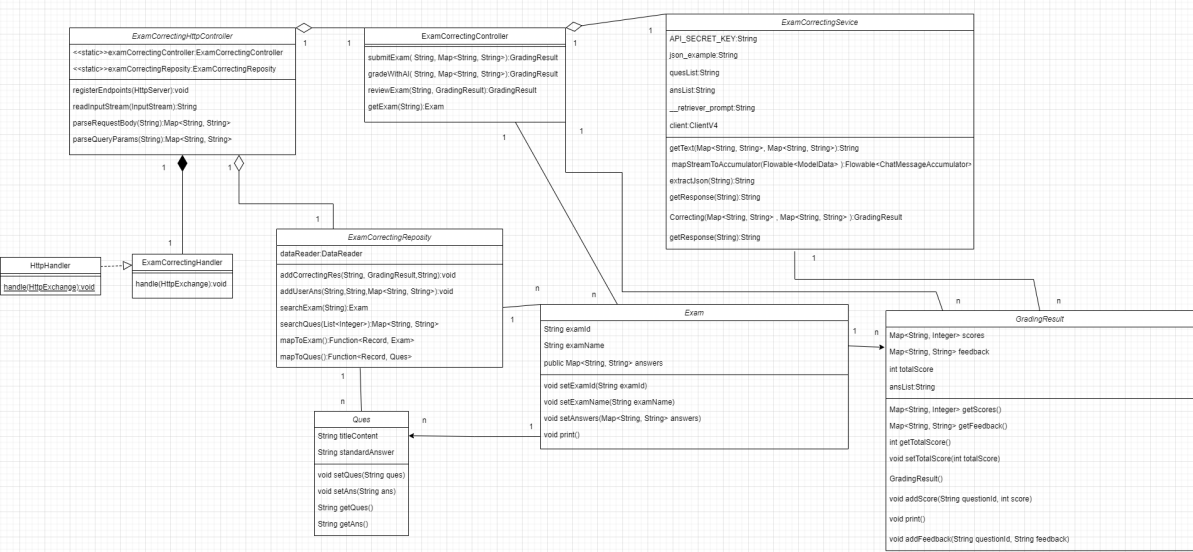


返回响应：

```
export interface Response {  
  /**  
   * http状态码，200为成功，其他为不成功  
   */  
  code: string;  
  
  [property: string]: any;  
}
```

# 试卷批改模块

试卷批改类图：



实现细节：

- **试卷提交批改后端实现：**  
该模块负责处理学生提交的试卷答案，并进行批改。学生通过前端提交答案，后端接收并存储答案，同时调用大模型接口进行批改。批改结果包含每道题的得分和反馈，并会存储在数据库中。在此过程中，`submitExam` 方法负责接收答案，并调用 `gradewithAI` 方法与大模型交互。
- **大模型批改试卷后端实现：**  
此模块的核心是与大模型进行交互，利用其强大的智能批改功能。通过调用大模型的批改接口，后端将学生提交的答案与标准答案进行比对，返回批改结果。这一过程不仅生成每道题的得分，还包括针对错误的详细反馈。
- **试卷复核功能后端实现：**  
教师可以对已批改的试卷进行复核，修改批改结果或提供额外的反馈。复核功能通过 `/api/exam/{userAnsId}/review` 接口实现，允许教师在后台进行修改。

## 数据设计

1. Exam 类设计：

- 表示一份试卷，包含试卷的基本信息以及该试卷的所有题目和答案。
- **数据字段：**
  - `examId`：试卷的唯一标识符。

- `examName`：试卷名称。
- `answers`：存储题目与标准答案的映射关系，使用 `Map<String, String>` 类型。
- **核心方法：**
  - `setExamId(String examId)`：设置试卷的 ID。
  - `setExamName(String examName)`：设置试卷名称。
  - `getExample()`：生成试卷示例。
  - `print()`：打印试卷信息，包括题目和答案。

## 2. GradingResult 类设计：

- 用于存储批改结果，包括得分和反馈。
- **数据字段：**
  - `scores`：存储每道题目得分的映射。
  - `feedback`：存储每道题目的批改反馈。
  - `totalScore`：总得分。
- **核心方法：**
  - `addScore(String questionId, int score)`：为指定题目添加得分。
  - `addFeedback(String questionId, String feedback)`：为指定题目添加反馈。

## 3. Ques 类设计：

- 表示一个单独的题目，包含题目内容和标准答案。
- **数据字段：**
  - `titleContent`：题目内容。
  - `standardAnswer`：标准答案。
- **核心方法：**
  - `setQues(String ques)`：设置题目内容。
  - `setAns(String ans)`：设置标准答案。

# 接口设计

## 1. 试卷提交批改接口：

- **请求方法：** POST
- **请求路径：** `/api/exam/{examId}/submit`
- **功能描述：** 提交学生的试卷答案，系统批改后返回得分和反馈。

请求参数：

请求参数

Path 参数	
<code>examId</code> string	必需
Query 参数 <span>生成代码</span>	
<code>userAnsId</code> string	必需
<code>username</code> string	必需
Body 参数 <code>application/json</code>	示例
<div><div>array of:</div><div><div><code>id</code> string 题目ID ID 编号</div><div><code>userAnswer</code> string 用户答案 用户传来的答案</div></div></div>	<pre>[   {     "id": "string",     "userAnswer": "string"   } ]</pre>

响应参数：

<div><div>▼ <code>correctionResults</code> array [object {2}] 批改结果</div><div><div><code>score</code> integer 得分</div><div><code>feedback</code> string 反馈</div></div></div>	必需
<div><div><code>totalScore</code> integer 总分</div></div>	必需
<div><div><code>success</code> boolean 成功与否 true为成功， false为不成功</div></div>	必需
<div><div><code>code</code> integer http状态码 200为成功， 其他为不成功</div></div>	必需

2. 大模型批改试卷接口：

- 请求方法：POST
- 请求路径：`/api/exam/{examId}/grade`
- 功能描述：通过大模型对试卷进行批改，返回得分和智能反馈。

请求参数：

请求参数		
Path 参数		
examId	string	必需
Query 参数		
userAnsId	string	可选
username	string	可选
Body 参数 application/json		示例
array of:		
id	string	可选
userAnswer	string	可选
		<pre>[   {     "id": "string",     "userAnswer": "string"   } ]</pre>

响应参数：

▼	correctionResults	array [object {2}]	批改结果	必需
	score	integer	得分	可选
	feedback	string	反馈	可选
	totalScore	integer	总分	必需
	success	boolean	成功与否 true为成功， false为不成功	必需
	code	integer	http状态码 200为成功， 其他为不成功	必需

3. 试卷复核接口：

- 请求方法：GET
- 请求路径： /api/exam/{userAnsId}/review
- 功能描述：教师对试卷进行复核，修改批改结果并提供反馈。

请求参数：

≡ 请求参数

Path 参数

userAnsId

string

必需

Query 参数

username

string

可选

<> 生成代码

Body 参数

application/json

correctionResults

array [object (2)]

必需

score

integer

可选

feedback

string

可选

totalScore

integer

必需

示例

```
{
  "correctionResults": [
    {
      "score": 0,
      "feedback": "string"
    }
  ],
  "totalScore": 0
}
```

响应参数：

feedback

array[string] 最终批改结果

必需

success

boolean 成功与否

必需

true为成功，false为不成功

code

integer http状态码

必需

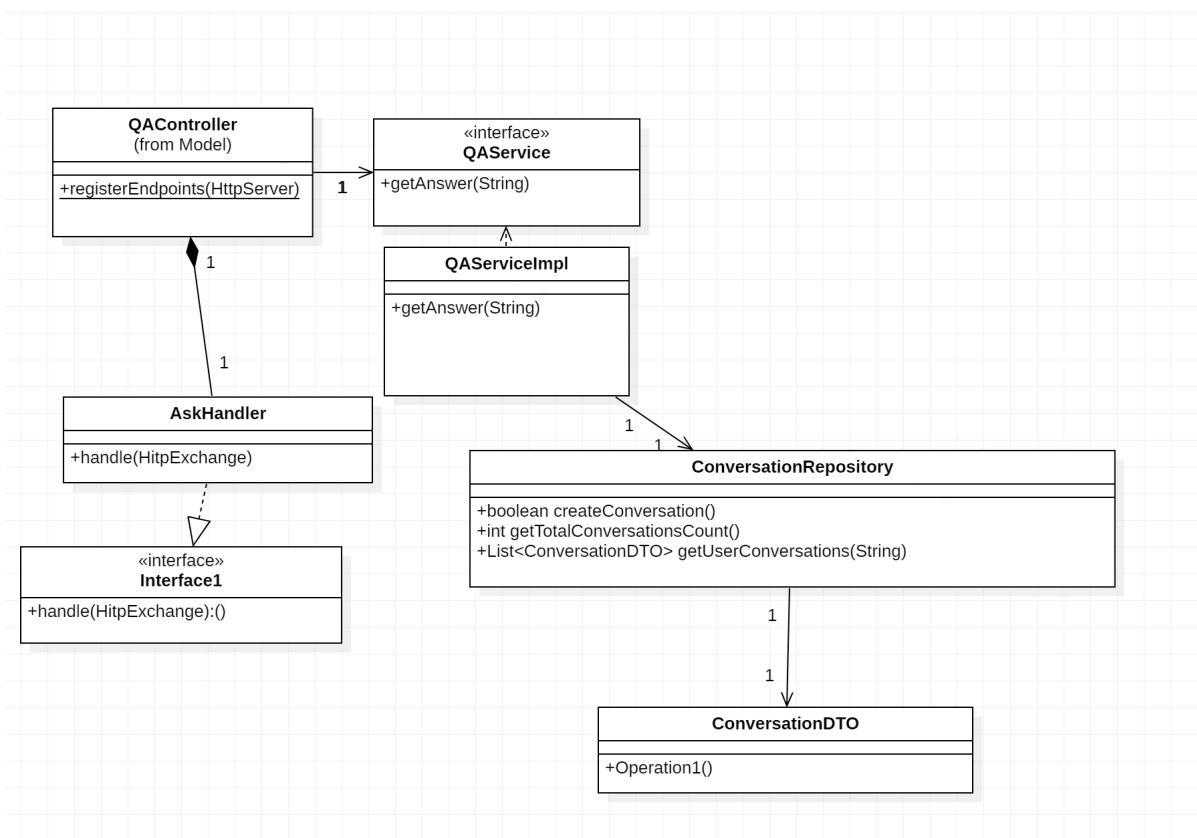
200为成功，其他为不成功

功能描述：

- 删除指定 ID 的试卷，删除操作会移除相关节点及其关系。

问答模块

问答类图：



#### 实现细节：

- **问答接口后端实现：**

该模块负责处理学生提出的问题并返回答案。学生通过前端提交问题后，后端接收问题并调用大模型接口进行回答。回答结果将通过 RESTful API 返回给前端。

核心功能包括接收问题的 `submitQuestion` 方法和调用大模型的 `getAnswerFromAI` 方法。

- **用户历史问答管理后端实现：**

该模块主要用于记录用户的历史提问和答案，支持查询和管理用户历史问答数据。通过 `/api/qa/history` 接口，用户可以获取自己的问答记录，并基于历史数据生成个性化反馈。

- **知识点关系生成后端实现：**

根据用户的问答记录，系统生成知识点与问题的关系图，帮助学生掌握未熟练的知识点。通过调用知识点关系生成器，分析历史数据并自动生成反馈。

## 数据设计

### 1. ConversationDTO 类设计：

- 表示一个对话的详细信息，包含对话 ID、用户名、问题和答案。

- **数据字段：**

- `conversationId`：对话的唯一标识符。
- `username`：用户名称。
- `question`：用户提出的问题。
- `answer`：系统提供的答案。

- **核心方法：**

- `getConversationId()`：获取对话 ID。
- `setConversationId(int conversationId)`：设置对话 ID。
- `getUsername()`：获取用户名。

- setUsername(String username)：设置用户名。
- getQuestion()：获取问题内容。
- setQuestion(String question)：设置问题内容。
- getAnswer()：获取答案内容。
- setAnswer(String answer)：设置答案内容。

## 接口设计

### 1. 提交问题接口：

- 请求方法：GET
- 请求路径：/api/qa/ask
- 功能描述：用户提交问题，系统调用大模型生成答案并返回。

请求参数：

请求参数

Query 参数

questionstring

用户的提问内容

示例值:"你好"

usernamestring

响应参数：

成功(200)

HTTP 状态码: 200

内容格式: JSON

application/json

数据结构

successboolean

成功与否

必需

true为成功, false为不成功

codeinteger

http状态码

必需

200为成功, 其他为不成功

answerstring | null

回答内容

必需

回答内容

示例

```
{
  "success": true,
  "code": 0,
  "answer": "string"
}
```

### 2. 查询历史问答接口：

- 请求方法：GET

- **请求路径：** `/api/qa/history`
- **功能描述：** 通过用户标识获取历史问答记录，支持个性化反馈。

请求参数：

### Query 参数

username

string

用户名

示例值：

“213223205”

响应参数：

成功(200)

运行

</>

HTTP 状态码: 200

内容格式: JSON

application/json

数据结构	示例
<div><div>result</div><div>array [object {3}]</div><div>结果</div><div>必需</div><div><div>id</div><div>integer</div><div>对话的id</div><div>必需</div></div><div><div>question</div><div>string</div><div>对话数组</div><div>必需</div></div><div><div>每段对话</div></div><div><div>answer</div><div>string</div><div>必需</div></div></div>	<pre>{  "result": [    {      "id": 0,      "question": "string",      "answer": "string"    }  ],  "code": "string",  "success": true}</pre>
<div><div>code</div><div>string</div><div>http状态码</div><div>必需</div><div>200为成功，其他为不成功</div></div>	
<div><div>success</div><div>boolean</div><div>是否成功</div><div>必需</div></div>	

## 功能描述

- **生成知识点关系：**  
系统根据用户的历史问答记录，自动生成知识点之间的关系图，为学生提供知识盲点反馈。

## 知识点查询与搜索模块

### 描述

用户可以通过模糊搜索的方式查找知识点，并查看与之相关的详细内容、关联知识点、前置和后置知识点等信息。

### 接口设计

- 1. 知识点搜索接口
  - **请求方法：** GET
  - **请求路径：** `/api/knowledge/search`



- **功能描述**：输入知识点关键词返回相应内容，支持模糊搜索。

<b>success</b>	boolean	成功与否	必需
		true为成功、false为失败	
<b>code</b>	integer	Http状态码	必需
		200为成功，其他为失败	
	默认值:	200	
▼ <b>result</b>	array [object {3}]   null	返回结果	必需
		包含搜索的各项内容	
<b>id</b>	string	知识点id	必需
		前端无需展示此字段	
<b>name</b>	string	知识点名称	必需
		名称	
<b>content</b>	string	内容	必需
		知识点内容	

- 2. 知识点详情接口
  - **请求方法**：GET
  - **请求路径**：/api/knowledge/search?keyword=
  - **功能描述**：展示某个知识点的详细内容

根节点	object	Mock	中文名	说明	⊕ ⊖
success	boolean *	Mock	成功与否	true为成功, false为不成功	⊕ ⊖
code	integer *	Mock	http状态码	200为成功, 其他为不成功	⊕ ⊖
▼ result	object *	Mock	结果对象	包含知识点的详细内容	⊕ ⊕ ⊖
name	string * 需 N	Mock	知识点名称	名称	⊕ ⊖
▼ prePoint	array * N	Mock	前置知识点	包含名称和id	⊕ ⊖
ITEMS	string	Mock	中文名	说明	
▼ postPoint	array * N	Mock	后置知识点	包含名称和id	⊕ ⊖
ITEMS	string	Mock	中文名	说明	
content	string *	Mock	内容	内容	⊕ ⊖
cognition	string *	Mock	认知维度	记忆/理解运用/了解	⊕ ⊖
tag	string * N	Mock	标签	难点/重点	⊕ ⊖
▼ relatedPoint	array * N	Mock	关联知识点	关联知识点	⊕ ⊖
ITEMS	string	Mock	中文名	说明	
note	string * N	Mock	补充说明	说明	⊕ ⊖
id	string *	Mock	id编号	ID 编号	⊕ ⊖

# 前端用户友好页面的设计

## 注册与登录页面



## 1. 功能分析

`Login.vue` 是一个用户登录和注册界面，提供了以下主要功能：

- **用户登录**：支持输入角色、用户名和密码进行验证。
- **用户注册**：切换至注册模式，支持新用户注册。
- **用户友好提示**：错误信息提示、状态反馈（加载状态和按钮禁用）。
- **切换登录/注册模式**：通过交互动态切换页面状态。

## 2. 用户界面友好设计

### 2.1 页面布局

- **居中设计**：整个页面内容居中（`justify-content: center; align-items: center`），为用户提供一个直观且集中的操作体验。
- **表单容器**：容器采用卡片式设计，增加了阴影（`box-shadow`），突出内容的层次感，同时背景白色增加清晰度。
- **响应式布局**：设置最大宽度 `400px`，保证页面在不同设备上显示一致且舒适。

### 2.2 输入表单

- 使用了 `Element UI` 提供的组件（`el-form`、`el-input`、`el-button`），保持了表单设计的一致性与专业外观。
- 角色、用户名和密码输入框
  - 支持动态绑定（`v-model`）。
  - 使用占位符（`placeholder`）引导用户输入正确的信息。
  - 提供了验证规则（`required`）和错误提示。
- 提交按钮
  - 按钮样式清晰，通过 `:loading` 属性动态切换加载状态，防止重复提交。
  - 禁用功能（`disabled`），增强用户体验。

## 2.3 动态切换

- 提供了“已有账号？点此登录”和“还没有账号？点此注册”切换按钮，用户可在登录和注册模式之间快速切换。
- 标题（`h2`）随状态动态更新，直观显示当前操作（登录或注册）。

## 2.4 错误提示

- 在表单下方提供了错误提示区域（`class="error-message"`），通过 `error` 字段绑定错误信息。
- 使用红色文字突出提示信息，帮助用户快速定位并修正输入错误。

# 3. 功能逻辑

## 3.1 登录/注册切换

- 点击切换按钮时，通过 `isRegistering` 标志控制模式切换。
- 动态更新标题和表单按钮文本，提供一致且直观的用户体验。

## 3.2 提交表单

- `handleSubmit`

方法处理登录/注册的提交逻辑：

- 动态选择 API 端点：根据 `isRegistering` 决定使用注册接口还是登录接口。
- 注册时，发送 `POST` 请求，包含用户信息。
- 登录时，发送 `GET` 请求，带有查询参数。
- 提供了加载状态（`loading`），并禁用按钮，防止重复提交。
- 根据服务器响应，提示用户成功或失败信息。

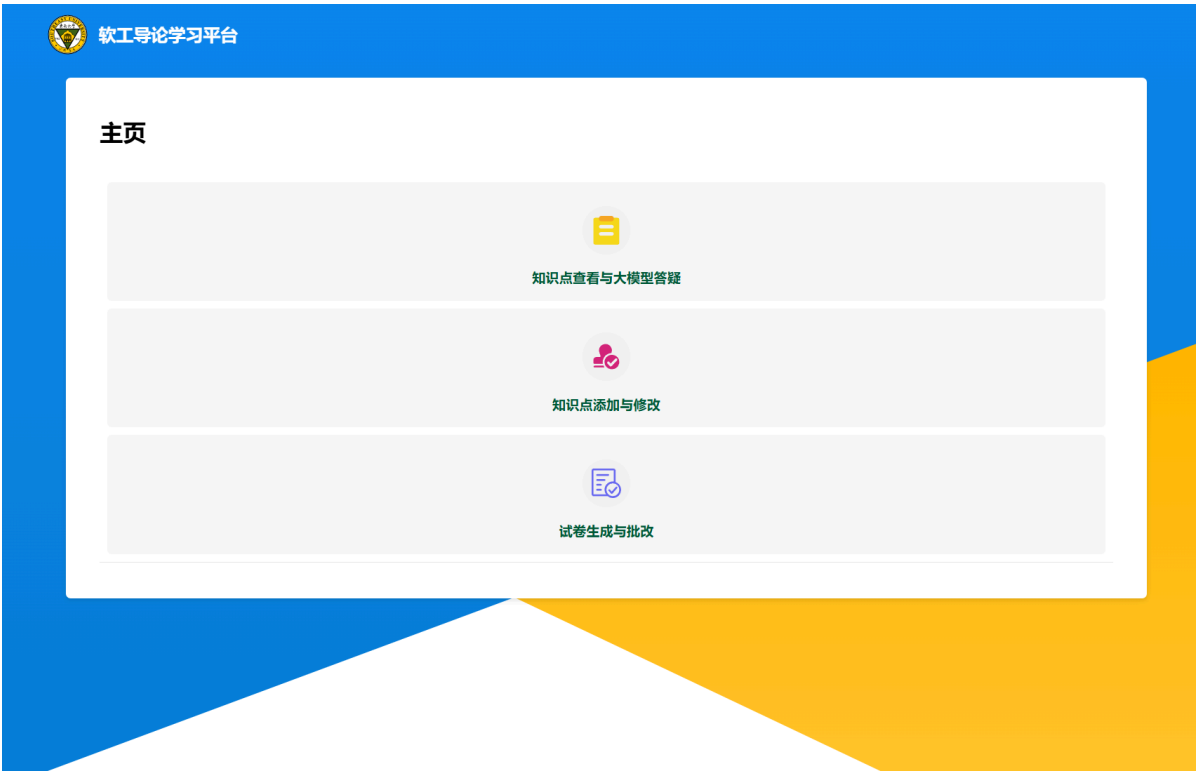
## 3.3 本地存储与事件触发

- 登录成功后，将用户名存储在 `localStorage`，便于后续页面使用。
- 触发 `login-success` 事件，便于父组件监听登录成功并执行后续操作。

# 4. 样式设计

- **简洁明了：**避免复杂装饰，使用统一的白色背景和蓝色强调文字，保持清晰简洁。
- **清晰的错误提示：**红色错误信息醒目且位置合理，便于用户快速修正错误。
- **视觉引导：**通过按钮和切换链接的颜色与交互效果，吸引用户关注操作。

# 平台主页



## 1. 功能分析

`HomePage.vue` 是平台的主页面，提供以下功能：

- **导航菜单：**通过水平菜单导航到不同功能页面（知识点查看、知识点修改、试卷生成）。
- **用户登录弹窗：**检测用户是否已登录，未登录时弹出登录对话框。
- **自定义背景：**提供简洁的背景组件，增强用户体验。

## 2. 用户界面友好设计

### 2.1 页面结构

- 整体布局采用 `Element UI` 的 `el-container` 组件，实现标准化的页面结构：
  - `el-header`：顶部导航标题（如“主页”）。
  - `el-main`：主要内容区域，包括导航菜单和登录弹窗。
- **背景组件：**通过 `Background` 组件为页面提供统一的背景设计，标题信息直观简洁。

### 2.2 导航菜单

- **水平导航菜单：**使用 `el-menu` 组件实现水平菜单导航，增强页面简洁性。
- 菜单项
  - 图标（`icon`）与标题结合，提升菜单的视觉吸引力。
  - 每个菜单项链接到对应的路由（`route`），点击后自动跳转。
- **模块化设计：**`MenuItem` 组件化菜单项，便于维护与扩展。

## 2.3 登录弹窗

- 弹窗触发条件
  - 页面加载时检查本地存储是否有 `username`。
  - 未登录时自动弹出登录对话框 (`el-dialog`)。
- 自适应弹窗宽度：根据屏幕大小动态调整弹窗宽度 (`min-width` 和 `max-width`)。
- 动态关闭
  - 登录成功后关闭弹窗。
  - 提供钩子函数 `handleBeforeClose`，支持关闭前执行自定义逻辑。

## 3. 功能逻辑

### 3.1 登录状态检测

- 自动弹窗：在 `mounted` 生命周期中检查 `localStorage`，如果未检测到 `username`，则设置 `notHaveUsername` 为 `true`，触发登录弹窗。
- 关闭弹窗
  - 登录成功后触发 `login-success` 事件，调用 `closeLoginModal` 方法关闭弹窗。

### 3.2 导航功能

- 通过 `el-menu` 的路由模式 (`router: true`)，实现菜单与路由的自动关联。
- 每个菜单项定义一个独立路由地址，点击后页面自动跳转到对应模块。

### 3.3 模块化与复用

- 背景组件：通过 `BackGround` 组件统一页面背景设计，提高一致性。
- 菜单项组件： `MenuItem` 组件实现每个菜单项的图标和路由绑定，便于扩展其他菜单。

## 4. 样式设计

### 4.1 弹窗设计

- 弹窗大小：通过 `width` 设置适中的宽度 (60%)，确保内容显示清晰。
- 响应式支持：定义 `min-width` 和 `max-width`，兼容不同屏幕尺寸。
- 滚动支持：为弹窗内容设置最大高度 (`max-height: 70vh`) 并启用滚动条，防止内容超出视窗。

### 4.2 层次感增强

- 导航菜单：使用水平导航布局和图标，突出功能模块。
- 弹窗居中：通过 `Element UI` 的 `center` 属性，使弹窗内容垂直居中，符合用户视觉习惯。

## 知识点查看与大模型答疑页面

知识点查看与大模型答疑页面由以下模块组成，每个模块均以用户友好和功能实用为核心设计理念。

# 1. 知识点搜索页面



## 功能描述：

- 提供知识点搜索功能，用户可以通过关键词查找知识点。
- 展示搜索结果，包含知识点的名称、内容、ID等详细信息。
- 支持分页显示，用户可以切换页面浏览大量知识点。
- 提供快捷跳转功能，点击知识点名称即可查看详情。

## 用户友好性设计：

- 搜索框和按钮均位于显眼位置，输入关键词即可触发搜索。
- 搜索结果以表格形式展示，直观明了。
- 使用分页组件管理结果，提升性能的同时避免信息过载。

## 关键技术：

- 使用 `Element UI` 的 `el-input` 和 `el-table` 实现交互式搜索和展示。
- 通过 `el-pagination` 实现分页功能，保证流畅用户体验。

## 2. 知识点详情页面

← 软件质量属性 | 知识点详情

前置知识点	[ "6.4" ]
后续知识点	[ "6.6" ]
内容详情	<p>质量属性 (quality attribute) 反映软件产品某一方面质量的特征或特性，如可靠性、安全性、易用性等。</p> <p>(1) 性能 (Performance) 效率指标，是指系统的响应能力，处理任务所需时间或单位时间内的处理量。</p> <p>(2) 可靠性 (Reliability) 是指软件系统在实际或错误使用情况下维持软件系统功能特性的基本能力。</p> <p>(2.1) 容错 (Fault-tolerant) 出现错误后仍能保证系统系统继续运行，且自行修正错误。</p> <p>(2.2) 健壮性 (Robustness) 是指在处理或环境中，系统能够承受压力或变更的能力，错误不对系统产生影响，按既定程序忽略错误。</p> <p>(3) 可用性 (Availability) 是系统能够正常运行的时间比例。</p> <p>(4) 安全性 (Security) 是指系统向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。</p> <p>(5) 可修改性 (Modification) 是指能够快速地对较高的性能价格比对系统进行变更的能力。</p> <p>(5.1) 可维护性 (Maintainability) 局部修复使故障对架构的负面影响最小化。</p> <p>(5.2) 可拓展性 (Extensibility) 因松散耦合更易实现新特征功能，不影响架构。</p> <p>(5.3) 可移植性 (Portability) 适用于多样的环境 (硬件平台、语言、操作系统)。</p> <p>(5.4) 结构重组 (Reconstructability) 不影响主体进行的灵活配置。</p> <p>(6) 可变性 (Changeability) 总体架构可变，体系结构经扩充或变更成为新体系结构的能力。</p> <p>(7) 功能性 (Functionality) 需求的满足程度，是系统所能完成所期望工作的能力。</p> <p>(8) 互操作性 (Inter-operation) 是指系统与外界或系统与系统之间的相互作用能力，通过可视化或接口方式提供更好的交互操作体验。</p> <p>(9) 易用性 (Usability) 是衡量用户使用一个软件产品完成指定任务的难易程度。</p> <p>(10) 可测试性 (Testability) 是指软件发现故障并隔离、定位其故障的能力特性，以及在一定的时间和成本前提下，进行测试设计、测试执行的能力。</p>
认知点	理解运用
标签	难点
相关知识	[ "6.4" ]
备注	学习软件质量的各项属性，理解如何评估和优化软件的质量特性。
知识点ID	6.5

### 功能描述：

- 展示单个知识点的详细信息，包括前置知识点、后续知识点、内容详情、认知点、标签、备注等。
- 提供“返回”按钮，便于用户导航回搜索结果页。

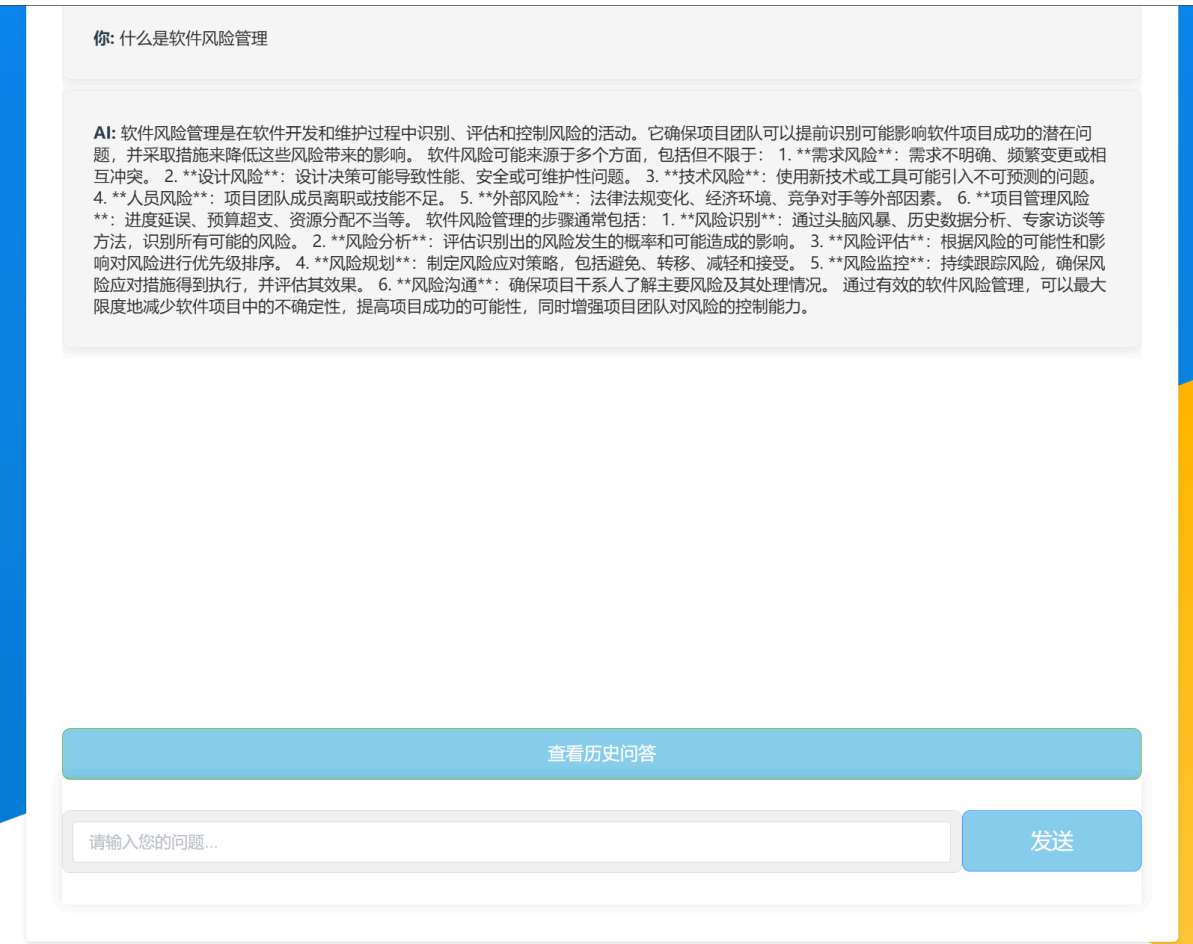
### 用户友好性设计：

- 以卡片形式清晰展示知识点详情，每个信息点均有对应的标题和内容。
- 在信息缺失时显示“无”而非留空，避免用户困惑。
- 提供“重新加载”功能以应对数据加载失败的场景。

### 关键技术：

- 使用 `el-card` 美化展示详情内容。
- 通过 `el-empty` 处理无数据或加载失败的情况。
- 支持动态路由跳转，用户点击返回按钮后可直接回到搜索页面。

### 3. 知识点搜索与问答页面



#### 功能描述：

- 综合知识点搜索和智能问答功能，为用户提供双重支持。
- 用户可以通过搜索查找知识点，也可以直接提问获取答案。
- 支持历史问答查看，用户可回顾以往的提问和答案。

#### 用户友好性设计：

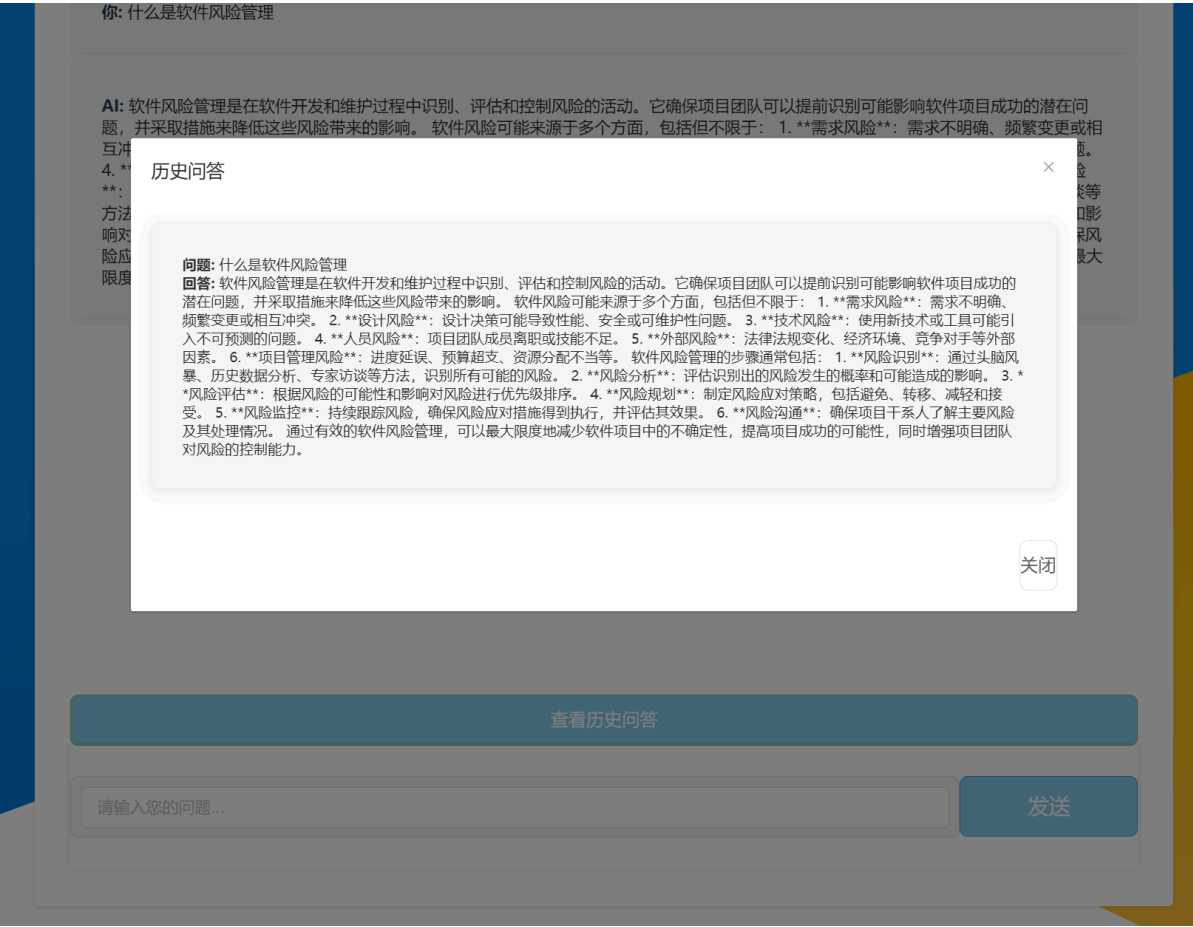
- 使用标签页区分搜索与问答功能，界面简洁、逻辑清晰。
- 智能问答部分包含输入框和发送按钮，操作直观。
- 问答历史以弹窗形式展示，不打扰当前操作。

#### 关键技术：

- 使用 `el-tabs` 实现搜索与问答的功能切换。
- 通过 `axios` 与后端交互，实时获取问答内容。
- 使用 `el-dialog` 优化历史记录显示方式。



## 4. 智能问答模块



### 功能描述：

- 用户可直接输入问题，系统调用大模型返回答案。
- 支持用户提问的历史记录查看，包括问题和模型回答。

### 用户友好性设计：

- 聊天消息分为两种（用户与AI），不同角色显示不同的背景和样式。
- 输入框自动对焦并支持按回车键发送，减少用户操作步骤。
- 历史问答记录以卡片形式展示，简洁清晰。

### 关键技术：

- 使用 `axios` 与后端 `API` 进行实时交互。
- 聊天记录采用虚拟滚动方式，保证大规模数据处理时流畅。
- 历史记录存储于后端，前端通过请求加载数据，减少初始加载的资源消耗。

# 知识点添加与修改页面

软工导论学习平台

知识点管理

添加知识点

修改知识点

\* 知识点名称

请输入名称

\* ID 编号

请输入 ID 编号

前置知识点

请输入前置知识点ID, 用逗号分隔

后置知识点

请输入后置知识点ID, 用逗号分隔

\* 内容

请输入内容

标签

请输入标签(重点/难点)

\* 认知维度


请输入认知维度(了解/识记/应用)

备注

请输入备注

提交

重置

软工导论学习平台

知识点管理

添加知识点

修改知识点

\* 搜索知识点 ID

请输入知识点 ID

搜索知识点

\* 知识点名称

请输入名称

\* ID 编号

请输入 ID 编号

前置知识点

请输入前置知识点ID, 用逗号分隔

后置知识点

请输入后置知识点ID, 用逗号分隔

\* 内容

请输入内容

标签

请输入标签(重点/难点)

\* 认知维度

请输入认知维度(了解/识记/应用)

备注

请输入备注

保存修改

重置

## 页面功能

### 1. 知识点录入：

- 用户可以输入知识点的名称、ID、内容等信息。
- 支持填写前置和后置知识点，并以逗号分隔的形式录入，方便形成知识点的逻辑关联。
- 提供认知维度、标签等详细信息的填写字段，方便知识点分类和管理。

### 2. 知识点搜索与修改：

- 用户可通过知识点 ID 搜索现有知识点信息。
- 支持对搜索到的知识点进行修改并提交保存。

### 3. 表单管理：

- 提供“重置”按钮，清空当前填写内容，便于用户快速重新录入。

### 4. 状态反馈：

- 提交成功或失败后，通过消息框实时反馈操作状态。
- 搜索知识点失败时，提示用户检查输入或稍后重试。

## 组件设计

### 1. 表单设计：

- 表单使用 Element UI 的 `el-form` 和 `el-form-item` 组件，结构清晰，具有自动对齐标签和输入框的功能。
- 每个字段均提供占位符提示，便于用户理解需要输入的信息类型。
- 必填字段如“知识点名称”、“ID 编号”等设置为必填项，并通过属性提示用户。

### 2. 布局优化：

- 页面采用 `el-row` 和 `el-col` 进行栅格布局，每个输入框宽度设置为 100%，确保不同屏幕设备上的响应式显示。
- 字段间设置适当间隔，避免信息密集。

### 3. 加载状态：

- 使用 `v-loading` 指令，在加载数据或提交操作时显示加载动画，提升用户体验。

### 4. 输入数据处理：

- `prePointString` 和 `postPointString` 将输入的字符串转化为数组后提交，便于后台数据处理。
- 提交时剔除无用的字符串字段，优化数据传输。

### 5. 交互优化：

- 搜索功能通过按钮触发，避免用户误操作。
- 搜索失败时提供明确的错误消息，减少用户困惑。

## 后端交互

#### • 提交接口：

- 使用 `axios.put` 方法将表单数据发送至后端 API。
- 提交时携带内容类型为 `application/json` 的请求头，确保数据格式正确。

#### • 搜索接口：

- 使用 `axios.get` 方法调用知识点查询 API，附加用户输入的 `ID` 参数。
- 根据返回数据填充表单，同时处理搜索失败的情况。

## 样式设计

### 1. 视觉优化：

- 表单整体白色背景，简洁明了，避免干扰用户输入。
- 使用适当的边距和内边距，增强整体视觉平衡。

### 2. 用户体验：

- 输入框的宽度和弹窗布局经过优化，适配大部分屏幕尺寸。
- 提供滚动条处理，保证内容过长时依然可读性良好。

## 试卷生成与批改页面

 软工导论学习平台

试卷生成与批改

[生成试卷](#) [批改试卷](#) [答题试卷](#)

试卷生成页面

☐ 全选

知识点名称	是否包含
项目管理概述	<input type="checkbox"/> 包含
度量	<input type="checkbox"/> 包含
估算与项目进度安排	<input type="checkbox"/> 包含
软件风险管理	<input type="checkbox"/> 包含
软件质量属性	<input type="checkbox"/> 包含
软件质量控制过程	<input type="checkbox"/> 包含
软件质量评审	<input type="checkbox"/> 包含

预览试卷

共 37 条 < 1 2 3 4 5 6 > 前往 1 页



试卷生成与批改

生成试卷 批改试卷 答题试卷

试卷批改页面

项目管理概述, 度量, 等 - 2024-12-03 20:1 用户作答 - cafd7cd9-00a9-40f1-99a5-5fe

批改

批改结果

总分: 80

反馈	得分
回答正确	10
回答不正确, 需要更详细准确的描述	0
回答不正确, 需要更详细准确的描述	0
回答正确	10
回答正确	10
回答不正确, 需要选择正确的选项	0
回答不正确, 正确答案为False	0
回答不正确, 正确答案为True	0
回答不正确, 需要更详细准确的描述	0



试卷生成与批改

生成试卷 批改试卷 答题试卷

试卷选择

试卷名称	操作
项目管理概述, 度量, 等 - 2024-12-03 20:07:54	<a href="#">开始答题</a> <a href="#">删除</a>
项目管理概述, 度量, 等 - 2024-12-03 20:14:43	<a href="#">开始答题</a> <a href="#">删除</a>
项目管理概述, 度量 - 2024-12-03 20:33:25	<a href="#">开始答题</a> <a href="#">删除</a>

共 3 条 < 1 > 前往 1 页

题目内容	作答
度量是对软件开发项目、过程及其产品进行数据定义、收集以及分析的持续性定量化过程。	<input checked="" type="radio"/> 正确 <input type="radio"/> 错误
估算与项目进度安排只涉及项目的成本评估，不包括工作量和时间的预测。	<input checked="" type="radio"/> 正确 <input type="radio"/> 错误
健壮性是指系统能够承受压力或变更的能力，错误不对系统产生影响。	<input type="radio"/> 正确 <input checked="" type="radio"/> 错误
性能是软件系统的响应能力、处理任务所需时间或单位时间内的处理量。	<input type="radio"/> 正确 <input checked="" type="radio"/> 错误
可用性是指系统能够正常运行的时间比例，而不是指系统的性能。	<input checked="" type="radio"/> 正确 <input type="radio"/> 错误
软件评审仅由开发团队内部进行，不涉及其他跨部门团队。	<input type="radio"/> 正确 <input checked="" type="radio"/> 错误
软件生命周期内的软件评审是什么活动，涉及哪些内容，由哪些部门组织进行，结果如何呈现？	<div>我不会</div>

提交答案

## 模块功能说明

### 1. 试卷生成 (ExamGeneration)

- 用户可以选择多个知识点，通过知识点生成试卷。
- 提供预览功能，用户可以在提交生成前查看试卷内容。
- 支持全选和分页显示，优化大批量知识点的选择操作。
- 支持实时加载知识点，并通过复选框灵活选择。

### 2. 试卷批改 (ExamCorrection)

- 教师可以选择学生的作答内容进行批改。
- 批改后提供详细结果，包括总分和每道题的得分与反馈信息。
- 自动匹配试卷与用户答案，确保数据的一致性。
- 支持实时加载学生答案和试卷列表。

### 3. 试卷答题 (ExamAnswer)

- 学生可在线完成试卷作答，包括简答题、选择题和判断题。
- 支持自动解析选择题的选项，并优化界面交互。
- 答题后将答案提交至后端，支持实时反馈提交状态。

### 4. 试卷选择 (ExamSelect)

- 学生可以浏览自己的试卷列表，并选择进入答题或删除试卷。
- 提供分页功能，优化大批量试卷管理。

## 模块设计亮点

### 用户友好交互

- **动态加载数据**: 各模块数据如知识点、试卷、答案等, 均在页面加载时动态从后端获取, 减少前端冗余存储。
- **分页与全选**: 试卷生成和选择模块采用分页与全选功能, 优化大数据量下的选择体验。
- **实时状态反馈**: 用户在提交试卷、批改等操作后, 页面通过实时提示 (如 `this.$message`) 告知用户操作结果。

### 响应式布局

- 使用 `Element UI` 的 `Row` 和 `Col` 栅格系统, 确保模块在不同设备屏幕上的显示效果。
- 自定义样式, 优化表格、按钮组和弹窗的布局, 提升整体视觉一致性。

### 模块化组件

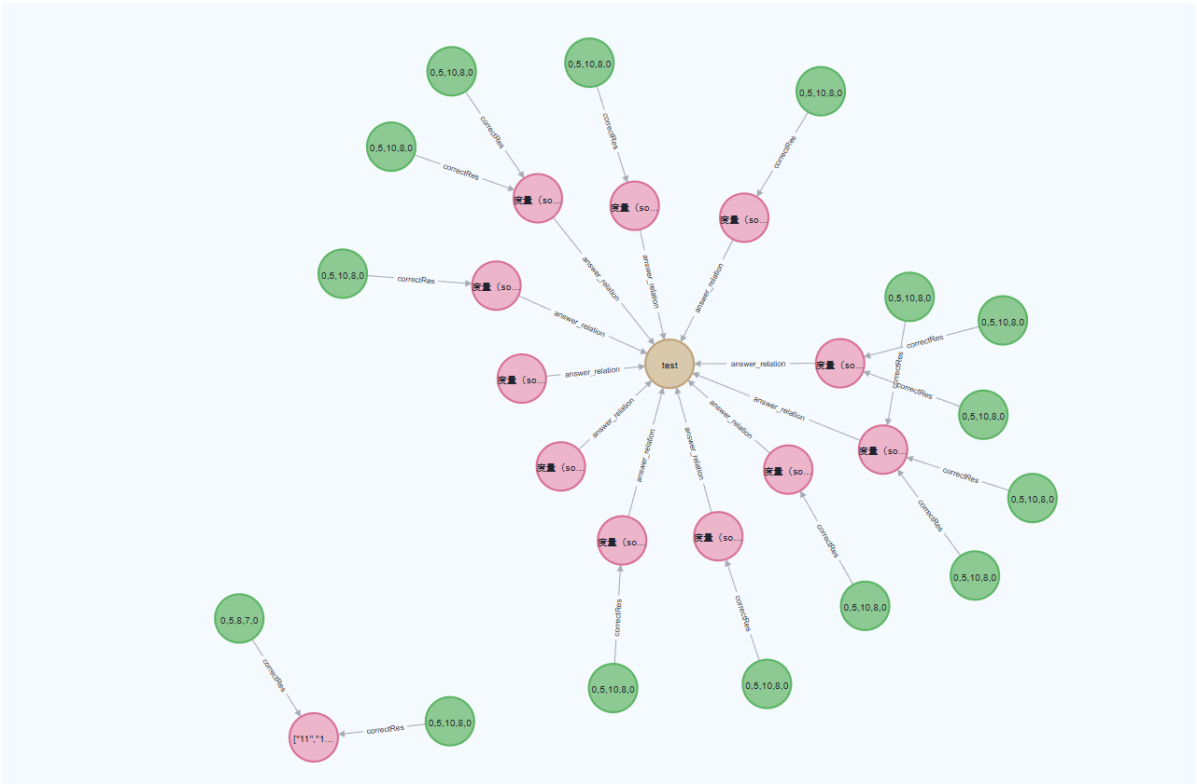
- 各模块组件化开发, 例如 `ExamGeneration`、`ExamCorrection` 等独立组件, 便于维护与复用。
- 使用 `KeepAlive` 保持子页面的状态, 减少重复加载数据的时间。

### 后端接口整合

- 所有数据交互通过 `axios` 完成, 接口设计统一, 支持 `GET` 和 `POST` 请求。
- 接口设计示例:
  - **试卷生成接口**: `POST /api/exam/generate`, 传递知识点ID列表并生成试卷。
  - **答案提交接口**: `POST /api/exam/submit`, 传递用户作答内容和试卷ID。
  - **批改接口**: `POST /api/exam/{examId}/grade`, 传递批改内容并返回结果。

The figure displays two network visualizations. The top visualization is a large, complex network with numerous nodes and edges. Nodes are colored in purple, orange, and blue. The bottom visualization is a smaller, more structured network with nodes colored in green, pink, and blue. Both networks show a high degree of connectivity and clustering.





## 1. 概述

本节描述了用于软件工程知识问答系统的数据库设计与开发方法。该系统基于 Neo4j 图数据库，旨在高效存储与管理知识点、习题、用户信息等数据。系统的核心需求是支持灵活查询不同节点间的关系，同时具备高效的查询性能、可靠的扩展性和良好的可维护性。

## 2. 数据库架构

### 2.1 架构模式

数据库设计采用图数据库架构，通过图节点与关系的灵活设计，实现数据的高效存储与查询。图数据库特别适用于表现复杂的多对多关系、依赖关系及路径查询，能够为软件工程知识问答系统提供强大的查询能力和高效的数据处理。

在架构设计中，图数据库的节点类型与关系类型之间构成了清晰的图结构，使得我们能够通过图数据库的深度查询，轻松获取某一知识点的相关习题、用户作答情况以及前后关系等信息。这种架构模式充分体现了数据库设计中的面向对象思想（Object-Oriented Design）和关系建模（Relational Modeling）原则。

### 2.2 数据库维护方法

数据库维护的方法包括但不限于以下几个方面：

- **数据一致性与完整性检查**：确保数据在修改和更新时不会丢失或损坏，特别是在复杂的图关系中，保证每个节点和关系都符合业务逻辑。
- **性能优化**：通过图数据库的索引机制、查询优化与并行处理策略，提升对复杂查询的响应速度。
- **备份与恢复策略**：制定定期备份方案，确保数据的安全性。特别是在数据库升级或大规模数据导入时，保持系统的高可用性。

## 3. 数据库设计

本节主要描述数据库中各类节点及其关系的设计。节点代表了系统中的各个实体，而关系则体现了实体之间的依赖和连接。通过精心设计节点和关系，可以高效地组织和存储数据，并为后续查询提供灵活性和高效性。

### 3.1 节点设计

在数据库中，存在多个类型的节点，它们分别代表了系统中的不同实体，主要包括以下几类：

#### 1. 主题节点（一级知识点）

主题节点是数据库中的一级知识点，代表了一个知识领域或一个核心概念。例如，软件工程、需求分析等。主题节点通常位于知识体系的顶层，用来归纳和组织相关的二级和三级知识点。

#### 2. 知识点节点（二、三级知识点）

知识点节点包括二级知识点和三级知识点。每个知识点节点代表了软件工程中某个具体的子模块或概念。知识点节点是系统的核心元素，它们详细描述了与特定主题相关的具体内容，可能包括子模块、算法、方法、流程等。

- **二级知识点**：这些知识点通常是较为广泛的子领域。
- **三级知识点**：这些知识点通常是更为具体和深入的内容。

#### 3. 习题节点

习题节点代表数据库中的各类题目。每个习题节点包括题目内容以及题型。习题节点有三种类型：

- **问答题 (type=1)**：要求用户自由回答问题。
- **选择题 (type=2)**：用户从若干选项中选择正确答案。
- **判断题 (type=3)**：用户选择“正确”或“错误”。

#### 4. 批改结果节点

批改结果节点表示用户作答后，系统对其答案进行的批改结果。每个批改结果节点包含答案是否正确、得分等信息，帮助系统评估用户的学习进度和掌握情况。

#### 5. 用户作答节点

用户作答节点记录用户对习题的作答情况。每个用户作答节点包括用户提交的答案、作答时间戳以及与习题之间的关联。该节点帮助系统追踪用户的作答历史。

#### 6. 用户信息节点

用户信息节点保存用户的基本资料，包括用户ID、姓名、邮箱等。此节点用于管理用户的注册信息以及与其他节点（如作答记录、习题等）的关联。

#### 7. 试卷节点

试卷节点代表系统中的一张试卷。每个试卷包含若干习题，试卷节点也记录了试卷的基本信息，如标题、创建日期等。

#### 8. Conversation节点（对话信息记录节点）

对话记录节点保存用户与系统的对话内容。它能够帮助系统跟踪用户在学习过程中的问题，并记录系统的回答。

### 3.2 关系设计

数据库中的关系是节点之间相互联系的纽带。以下是不同节点之间的主要关系类型：

#### 1. include\_relation（包含关系）

该关系表示主题节点与知识点节点之间的包含关系，或者二级知识点与三级知识点之间的包含关系。主题节点包含多个知识点节点，二级知识点节点包含多个三级知识点节点。该关系反映了知识体系中的层级结构和归属关系。

## 2. `post_relation` (后续关系)

该关系表示知识点之间的顺序或依赖关系。具体而言，一个知识点的学习通常需要在掌握了其他知识点之后进行。后续关系连接的是一个知识点与它后续需要学习的知识点。

## 3. `pre_relation` (前置关系)

该关系表示知识点之间的前置依赖关系。一个知识点的掌握通常需要在学习了其他知识点之后进行。前置关系则连接的是一个知识点与其需要依赖的先前知识点。

## 4. `exercise_relation` (习题关联关系)

该关系表示知识点与习题之间的联系。一个知识点可能会与多个习题相关联，习题则帮助用户理解并巩固相关知识点。

## 5. `answer_relation` (作答关联关系)

该关系表示用户作答与习题之间的联系。每个用户作答节点都与相应的习题节点相关联，用于追踪用户的作答情况。

# 3.3 节点与关系的交互

通过节点与关系的结合，可以实现对软件工程知识问答系统中数据的灵活管理和查询。以下是一些典型的数据查询和交互场景：

- **查询某个主题下的所有知识点**

通过 `include_relation` 关系，能够获取某一主题节点下所有的二级和三级知识点。

- **查询某个知识点的前置知识点**

通过 `pre_relation` 关系，可以快速查询某个知识点所依赖的前置知识点。

- **查询某个知识点下的相关习题**

通过 `exercise_relation` 关系，可以查找与某个知识点相关的所有习题，帮助用户进行针对性的练习。

- **追踪用户的作答记录**

通过 `answer_relation` 关系，可以追踪用户对特定习题的作答记录，评估其学习进度和知识掌握情况。

# 3.4 设计原则

在设计数据库节点和关系时，遵循以下几个原则，以确保数据库的高效性、可维护性和扩展性：

## 1. 清晰的层级结构

通过 `include_relation`、`post_relation` 和 `pre_relation` 等关系，清晰地表示出知识点之间的层次关系及依赖顺序，帮助用户更好地理解知识点的构成与学习路径。

## 2. 关系的灵活性

通过合理设计关系类型，使得系统可以灵活地管理节点间的复杂关联。例如，习题和知识点之间的 `exercise_relation` 关系能够帮助建立高效的学习路径。

## 3. 易于扩展

系统能够轻松增加新的节点类型或关系类型，支持不断扩展的知识体系和习题库。例如，新的习题类型或新的知识点类别可以轻松集成到现有架构中。

## 4. 高效的数据查询

通过图数据库的优势，使得系统能够快速执行复杂的多级查询。节点间的关系设计保证了查询的高效性，避免了关系型数据库中常见的表连接操作所带来的性能瓶颈。

# 四、质量保证

## 试卷生成与用户作答

### 测试策略：

为了确保试卷生成与用户作答模块的可靠性、稳定性以及用户满意度，我们实施了全面的多层次测试策略。此策略涵盖了单元测试、模块测试、前后端联调测试、系统测试及交付测试，确保每个模块和功能在实际环境中能够顺利运行。以下是基于试卷生成与用户作答功能的具体测试内容和执行策略：

#### 1. 单元测试（基础逻辑、函数等）

单元测试是对系统中最小功能单元（如函数和方法）进行的独立测试，确保每个功能的基本逻辑正确性。我们通过单元测试验证核心数据处理类的功能，并确保在单一功能模块下没有潜在的逻辑错误。

##### 测试内容：

- **ExamService 类：**
  - 测试 `generateExam` 方法，确保根据知识点生成的试卷包含正确的题目和知识点。
  - 测试 `saveExam` 方法，确保保存试卷时，试卷的题目和相关数据能够正确更新，并且在数据库中标记为已保存。
- **提交与作答相关方法：**
  - 测试 `submitExam` 方法，确保提交的答案能够正确关联到用户，并且能够正确存储到数据库中。
  - 测试 `getAnswersByExamId` 方法，确保用户提交的答案能够通过试卷 ID 和用户名进行准确查询。
  - 测试 `deleteExam` 方法，确保删除试卷操作能够在数据库中成功执行。
- **数据模型测试：**
  - 确保 `Exam` 类、`UserAnswer` 类和其他数据结构能够正确地映射到数据库中的记录，并能够正确处理存储和查询。

##### 示例测试：

- 测试 `generateExam` 方法是否能根据传入的知识点 ID 正确生成试卷，并包含指定数量的题目。
- 测试 `submitExam` 方法是否能成功将用户提交的答案存储并与试卷关联。

#### 2. 模块测试（每个接口、页面）

模块测试验证每个独立模块和功能是否能够按照预期工作，确保不同部分的业务逻辑实现无误。通过模拟用户请求，测试试卷生成、用户作答提交和试卷查询等接口的功能。

##### 测试内容：

- **试卷生成接口：**
  - 测试 `POST /api/exam/generate` 接口，确保传入知识点 ID 列表后，生成的试卷包含正确的题目，并且返回的试卷 ID 和标题符合预期。
- **试卷保存接口：**
  - 测试 `GET /api/exam/save` 接口，确保保存试卷后，数据库中保存的试卷状态正确更新。

- **用户作答提交接口：**

- 测试 `POST /api/exam/submit` 接口，确保用户提交的答案能够正确与试卷进行关联，并且答案能够正确存储。

- **试卷内容获取接口：**

- 测试 `GET /api/exam/getExam` 接口，确保通过传入的试卷 ID 和用户名能够获取到正确的试卷内容和题目列表。

**示例测试：**

- 测试 `generateExam` 方法生成试卷后，调用 `saveExam` 接口是否能够成功保存该试卷。
- 测试 `submitExam` 接口是否能够正确保存用户的作答并返回状态。

---

### 3. 前后端联调测试（每个页面和对应的接口）

前后端联调测试确保前端页面能够正确调用后端接口，并且数据交互顺畅，UI呈现的数据准确无误。确保用户从试卷生成、作答提交到查看试卷结果的整个过程能够顺利执行。

**测试内容：**

- **试卷生成页面：**

- 模拟用户选择多个知识点并点击“生成试卷”按钮，前端通过调用 `/api/exam/generate` 接口生成试卷。
- 测试生成后的试卷是否正确展示在页面上，题目是否完整。

- **用户作答提交页面：**

- 测试学生在提交试卷时，是否能够将用户的答案成功提交到 `/api/exam/submit` 接口，并且返回正确的批改结果。

- **试卷批改结果页面：**

- 测试学生提交试卷后，批改结果是否正确显示，得分和详细反馈信息是否完整。

**示例测试：**

- 模拟用户选择多个知识点并点击“生成试卷”按钮，前端调用接口生成试卷，检查返回的数据是否在页面上正确展示。
- 模拟学生提交作答，前端调用 `submitExam` 接口，检查后端返回的状态码和批改结果是否准确显示在页面上。

# 结果分析

## 功能性质量评估

### 需求满足度：

在项目需求分析阶段，我们确保了所有功能模块的需求都被准确捕捉并得到了实现。通过详细的接口设计和数据结构设计，每个功能模块，如试卷提交、批改、复核等，均得到了充分的验证，确保符合最初的需求规格说明书。特别是对于大模型批改和复核功能，我们进行了反复的验证与测试，以确保每个环节的数据流和功能操作正确。

## 功能正确性：

通过多轮的单元测试和集成测试，我们确保了每个功能模块都能够按预期工作。例如，试卷提交批改接口能够正确接收学生提交的答案并生成批改结果；大模型批改接口能够正确调用外部AI接口并返回准确的批改反馈。在大模型批改中，我们也特别关注了接口的调用和反馈的准确性，避免了信息丢失或数据误差。

## 功能完整性

基本实现了需求文档中的所有功能

# 非功能性质量评估

## 核心模块的可靠性

### 试卷生成模块

- 可靠性保障：  
使用 usedQuestionIds 集合，确保每道题只出现一次，减少重复性。  
每个试卷的存储操作（如调用 saveExam 方法）保证了试卷生成后的数据一致性。通过 Apifox 对接口的逻辑进行多轮验证，确保从知识点到题目的映射没有偏差。
- 潜在问题：  
数据库查询性能可能在知识点和题目数量大幅增长时出现瓶颈。

### 试卷批改模块

- 可靠性保障：  
提前对大模型API调用设置超时和重试机制，避免接口调用失败影响整个流程。  
批改结果存储在数据库中，与试卷和用户作答进行关联，保证数据不会丢失。  
教师复核功能（reviewExam）作为兜底手段，确保即使大模型结果存在偏差，也能通过人工方式校正。
- 潜在问题：  
大模型在高并发情况下可能导致响应延迟，影响批改的实时性。  
批改结果依赖模型的表现，对于复杂主观题，可能会存在误判。

### 知识点查询模块

- 可靠性保障：  
数据库索引支持模糊查询，确保关键词搜索的响应时间在需求内（1秒）。  
在前端实现了关键词输入防抖机制，避免重复查询浪费资源。
- 潜在问题：  
知识点的前后置关系如果维护不当，可能导致错误的关联结果。

# 核心模块可维护性

---

## 后端模块的分层设计

-模块分工明确：

controller：处理用户请求，调用业务逻辑。

service：负责具体的业务逻辑实现。

repository：与数据库交互。

model：封装数据结构。

## 接口和数据设计

- 标准化接口： 每个接口的请求参数和返回值都有详细说明。例如，`/api/exam/submit` 的参数中明确定义了试卷ID和答案格式。这样的设计降低了接口调用的出错率。
- 扩展性强： 在增加试卷复核功能时，我们只需要在现有的试卷批改数据模型上新增一个复核字段，而不需要修改原有的接口和逻辑。

## 知识点图谱模块

模块化设计： 知识点的增删改查功能完全独立，不会影响到试卷模块。这种解耦设计便于后期在知识点模块中增加标签分类或难度级别等新特性。

实际维护体验： 在一次需求变更中，新增了知识点的“前置知识点”关系，仅需对后端的知识点查询逻辑进行小幅修改，其他模块完全无需调整。

## 可用性

---

### 界面设计

我们的界面设计简洁直观，便于用户使用

- 知识点查询页面：  
提供了清晰的搜索框，支持关键词模糊查询。  
搜索结果以列表形式呈现，点击即可查看详细信息，包括前后置关系、知识点标签等。
- 试卷生成页面：  
用户只需选择知识点并点击“生成试卷”按钮，系统自动完成试卷生成并预览，减少了用户的操作复杂度。  
提供实时反馈（如生成进度和错误提示），提高用户信心。
- 试卷答题页面：  
题目清晰分组，包含题目类型、题目内容及选项。
- 大模型问答页面: 按照ChatGPT网页版风格设计

## 代码质量

---

可读性：方法命名清晰，例如，`generateExam` 一目了然地描述了其生成试卷的功能。注释简洁但重点明确，对于复杂方法（如`submitExam`），注释详细描述了其主要逻辑。

在新增“多用户角色支持”时，代码只需扩展权限控制部分，而无需修改其他核心功能模块。

# 缺陷分析

## 缺陷分布：

- 在试卷生成与批改部分，大部分缺陷集中在大模型批改功能和接口交互部分，尤其是在大规模批改时，部分接口存在响应超时或返回数据不一致的情况。经过多次调试和优化后，相关问题已经得到有效解决。我们还加强了接口的容错机制，避免了因外部API调用问题导致的功能失效。

## 缺陷趋势：

- 随着开发和测试的深入，缺陷数量呈现出逐步减少的趋势。这表明缺陷管理流程逐渐成熟，团队对系统的理解和控制越来越好。后期的测试周期中，缺陷的修复效率也显著提升。