基于 SpringBoot 与大语言模型的在线考试系统“KnowledgeOunce”的设计与实现

姓　　名：　　夏玉杰

系　　所：　先进技术研究院

完成时间：　2026 年 1 月 6 日

摘要

随着互联网技术在教育领域的深度应用，传统的线下考试模式因其空间限制和人工评阅效率低下的问题，已难以满足现代教育对于灵活性与智能化的需求。本文设计并实现了一款基于 B/S 架构的全栈在线考试系统——KnowledgeOunce。系统后端采用 SpringBoot 框架，结合 MyBatis 进行数据持久化，利用 Redis 缓存技术实现验证码校验与会话管理；前端基于 Vue.js 构建响应式交互界面。

为解决传统考试练习系统缺乏即时反馈与深度解析的痛点，本系统创新性地接入了 DeepSeek 大语言模型，为考生提供智能化的错题分析与知识点讲解。本文重点阐述了系统的后端架构设计、数据库实体建模、JWT 鉴权机制、试题随机组卷算法以及 AI 辅助分析等核心模块的实现。测试结果表明，该系统运行稳定、功能完善，有效提升了用户的练习效率与学习体验。

关键词：SpringBoot；在线考试系统；大语言模型；Redis；前后端分离；JWT

目录

第一章 绪论

1.1 研究背景与意义

在信息化时代，在线教育平台已成为获取知识的重要途径。传统的考试形式受限于时间、地点及人工阅卷的效率低下，难以适应大规模、个性化的学习需求。构建一个轻量级、高可用且具备智能反馈功能的在线考试系统，对于提升教学评价效率、辅助学习者自我诊断具有重要意义。

本项目“KnowledgeOunce”旨在解决传统考试系统交互单一、缺乏深度解析的问题。通过引入大语言模型（LLM），系统不仅能自动评分，还能针对用户的错题提供 AI 智能解析，实现了从单纯的“考核”工具向“助学”工具的转变。

1.2 国内外研究现状

目前市场上的考试系统大多侧重于题库管理和自动评分（如 Moodle，问卷星等），但在用户体验的流畅度以及对试题的智能化分析方面仍有欠缺。多数开源项目架构较为陈旧，缺乏对高并发场景下的缓存处理及现代 AI 技术的应用。本文提出的系统结合了当前主流的 Java Web 技术栈（SpringBoot + MyBatis + Redis）与新兴的 AI 接口技术，具有较强的先进性与实用性。

1.3 论文结构安排

本文共分为六章。第一章介绍项目背景；第二章进行系统需求分析与技术选型；第三章详细阐述数据库设计；第四章重点论述后端核心功能的实现；第五章简述前端实现与系统测试；第六章总结全文。

第二章 系统分析与技术架构

2.1 需求分析

本系统主要面向两类用户：管理员与普通考生。

1. **考生端功能：**
    - **用户注册与登录**：支持邮箱验证码注册，保障账户真实性。
    - **在线练习**：用户可自定义选择科目（如高级数据库、工程伦理）、题型（选择题、填空题）及题目数量进行随机组卷。
    - **智能分析**：答题结束后，针对错题通过 AI 生成解析。
    - **历史记录与收藏**：查看过往练习成绩，收藏错题。
    - **数据统计**：可视化展示做题数量、正确率及超越人数。
2. **管理端功能：**
    - **试题管理**：支持 Excel 模板批量导入试题。

2.2 系统架构设计

系统采用标准的 B/S（Browser/Server）架构，基于 MVC 设计模式，实行前后端分离开发。

- **前端层**：使用 Vue.js + Element UI，负责页面渲染与用户交互，通过 Axios 与后端进行异步数据通信。
- **后端层**：基于 SpringBoot 框架，提供 RESTful API 接口，负责业务逻辑处理。
- **数据层**：MySQL 存储业务数据，Redis 作为缓存中间件处理验证码及临时会话信息。
- **AI 服务层**：后端通过反向代理接入 DeepSeek API，为系统提供大模型分析能力。

2.3 后端关键技术选型

- **SpringBoot**：利用其“约定优于配置”的特性，快速搭建后端服务，结合 AOP 实现全局异常处理与日志记录。
- **MyBatis**：作为 ORM 框架，通过 XML 配置文件实现 SQL 与 Java 代码的解耦，灵活处理复杂的多表关联查询。
- **JWT (JSON Web Token)**：实现无状态的分布式身份认证，配合 Redis 黑名单机制，有效防止 Token 滥用。
- **Redis**：用于存储有时效性的数据（如邮箱验证码）。
- **DeepSeek API**：通过 HTTP Client 调用大模型接口，实现自然语言处理功能。

第三章 数据库设计

数据库设计是后端开发的基础。根据系统业务逻辑，本系统设计了满足第三范式的关系型数据库模型。

3.1 实体关系分析 (ER 图)

系统主要实体关系如下：

- **用户(User)**与**考试记录(ExamHistory)**是一对多关系，一个用户可进行多次练习。
- **用户(User)**与**题目收藏(QuestionCollection)**是一对多关系。
- **科目(Subject)** 是题目的分类依据，与**题目(Question)**为一对多关系。
- **考试记录**包含具体的答题快照与得分情况。

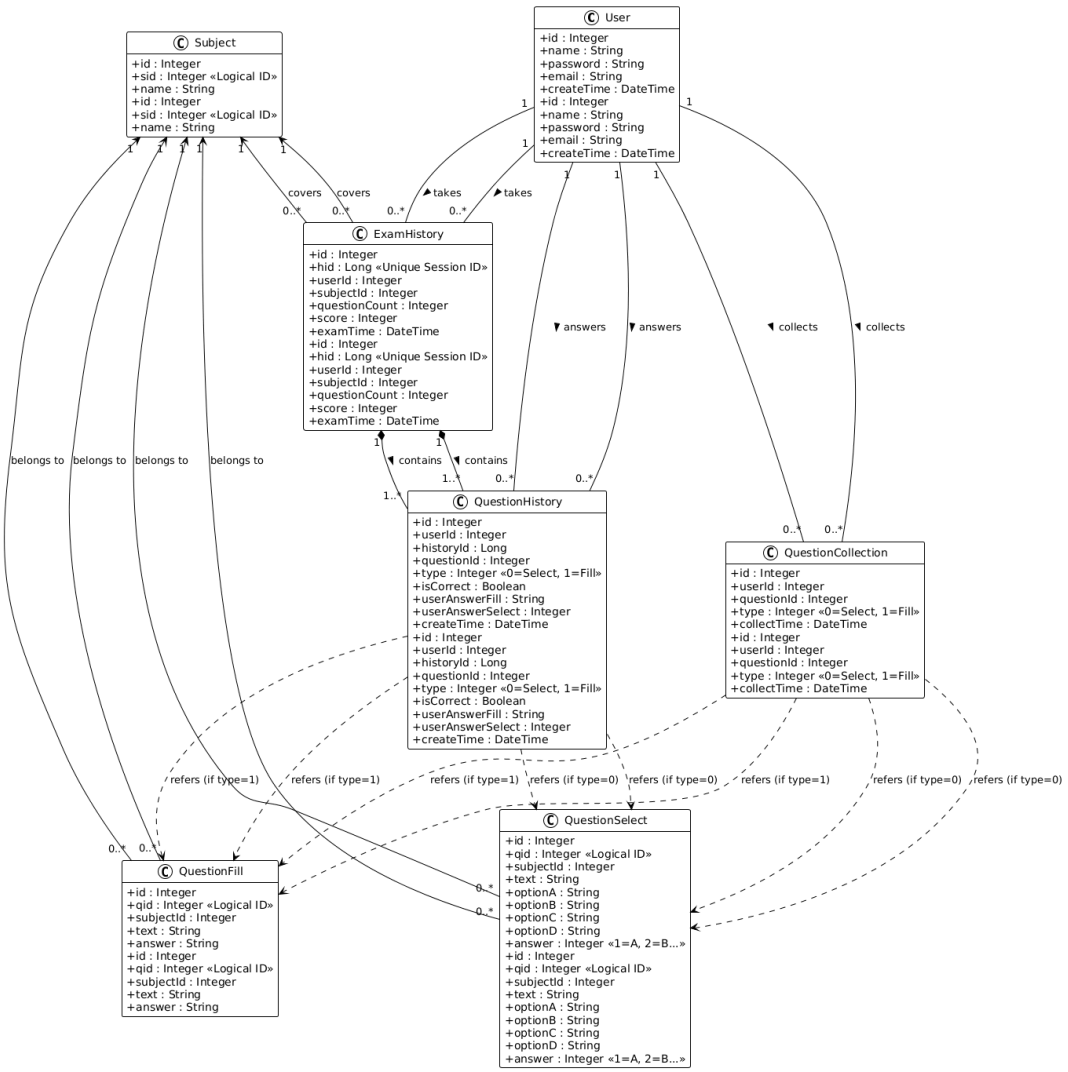

图 3-1 数据库 ER 关系图

3.2 核心数据表结构

3.2.1 用户表 (User)

字段名	类型	说明
id	Integer	主键/自增
name	String	用户昵称
password	String	加密后的密码
email	String	注册邮箱/唯一索引
createTime	DateTime	注册时间

表 3-1 用户表

3.2.2 试题表 (QuestionSelect / QuestionFill)

鉴于选择题和填空题结构差异，系统采用分表策略存储。以选择题为例：

字段名	类型	说明
id	Integer	主键
subjectId	Integer	关联科目表 ID
text	String	题干内容
optionA/B/C/D	String	选项内容
answer	String	标准答案
logicalId	Integer	逻辑编号

表 3-2 试题表

3.2.3 练习记录表 (ExamHistory)

字段名	类型	说明
id	Long	主键
userId	Integer	关联用户
subjectId	Integer	考试科目
score	Integer	本次得分
examTime	DateTime	考试时间

表 3-3 练习记录表

3.2.4 试题收藏表 (QuestionCollection)

字段名	类型	说明
id	Integer	主键/自增
userId	Integer	关联用户 ID
questionId	Integer	关联题目 ID
type	Integer	题目类型 (0=选择题 1=填空题)
collectTime	DateTime	收藏时间

表 3-4 试题收藏表

第四章 后端核心功能设计与实现

本章将详细阐述后端核心模块的设计思路与代码实现逻辑。

4.1 用户注册与鉴权模块

图 4-1 用户注册

4.1.1 邮箱验证码发送

为了防止恶意注册和机器人攻击，系统引入了邮箱验证码机制。

- **实现逻辑：**
  1. 用户请求发送验证码。
  2. 后端生成 6 位随机数字。
  3. 调用 JavaMailSender 发送邮件。
  4. **存入 Redis 缓存：**Key 为 code:email，设置 TTL（过期时间）为 5 分钟。
  5. 注册提交时，从 Redis 取出验证码进行比对，比对成功后删除 Key。

图 4-2 用户登录

4.1.2 JWT 登录认证

系统弃用了传统的 Session 模式，采用 JWT 进行鉴权，更适合前后端分离架构。

- **登录流程：**用户提交账号密码 -> 后端校验通过 -> 生成包含用户 ID 和角色的 JWT Token -> 返回给前端。
- **拦截器实现：**定义 LoginInterceptor，拦截所有非公开接口（如 /admin/**）。拦截器解析请求头中的 Authorization 字段，验证 Token 签名及有效期。若验证失败，直接返回 HTTP 401 状态码。

4.2 试题导入功能

为了方便管理员维护海量题库，后端实现了 Excel 批量导入功能。

- **技术实现：**后端接收前端上传的 MultipartFile，使用 Apache POI 组件解析 Excel 文件流。
- **事务控制：**解析过程中，系统会进行数据格式校验。采用 Spring 的 @Transactional 注解，若导入过程中出现任何异常（如数据格式错误），所有数据回滚，保证数据库的一致性。

4.3 在线练习与自动组卷

这是系统的核心业务功能，涉及复杂的业务逻辑。

4.3.1 随机组卷算法

前端传递参数：科目 ID、题型、题目数量（如 5 题）。

- **算法优化：**为避免大数据量下 SQL ORDER BY RAND() 带来的性能问题，后端采用“先查询 ID 列表 -> 真随机数算法 -> WHERE id IN (...)”的策略获取题目。这显著提升了组卷接口的响应速度。

4.3.2 答题交互与自动评分

用户在前端完成作答后，系统提供实时反馈。

1. **前端展示：**采用卡片式布局，支持上一题/下一题切换。
2. **提交判分：**后端接收用户答案，比对数据库中的标准答案。
3. **结果处理：**计算总分，将本次练习记录写入 ExamHistory 表，同时将答错的题目自动加入 QuestionCollection（错题集）。

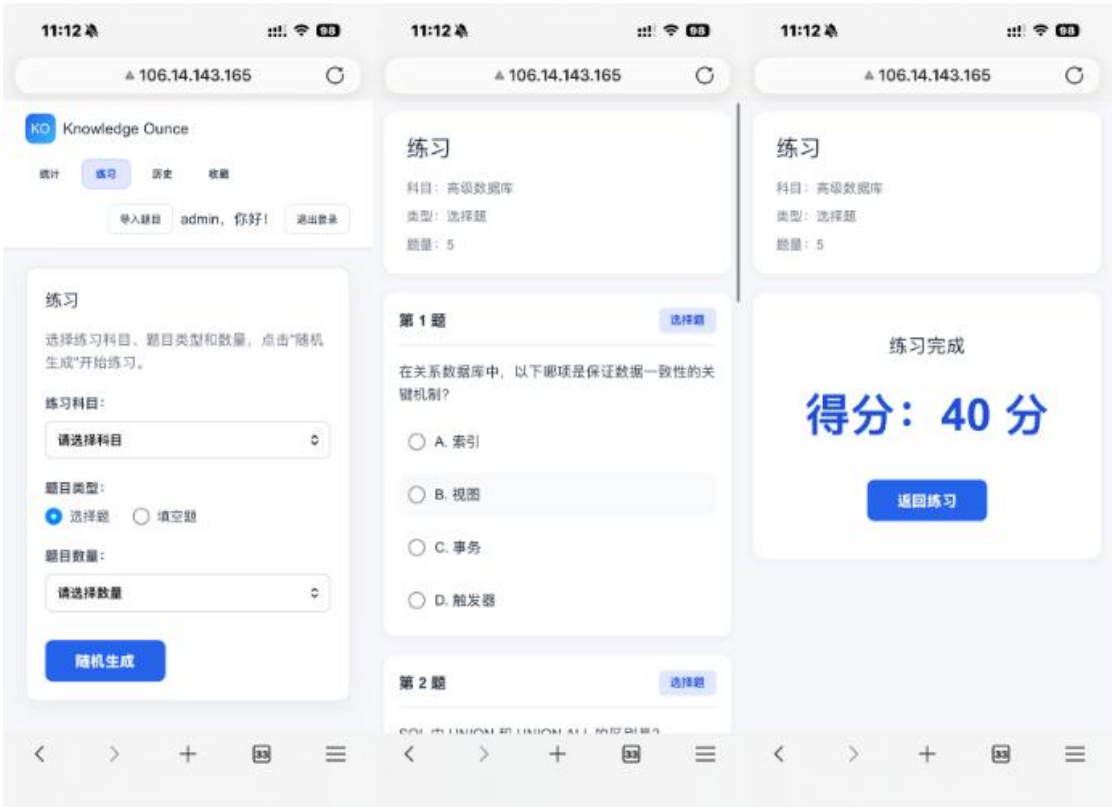

图 4-3 随机组卷与自动评分

4.4 AI 大模型分析接入

为了提升系统的教育价值，后端集成了 DeepSeek 大模型对题目进行深度解析。

- **架构设计**：由于网络环境限制，后端通过部署 DeepSeek2API 反向代理工具进行请求转发。
- **Prompt 工程**：后端构建特定的 Prompt 模板，例如：

“这是一道试题：{text}。考生的错误答案是：{userAnswer}。正确答案是{standardAnswer}。分析这道题出错的点，并给出对应的考点以便考生复习。简要回答，控制在两三句话左右，且不要给出“如果你需要的话”等扩展性建议。”

- **异步调用**：当用户点击“AI 分析”时，后端调用服务器部署的大模型接口，解析返回的文本并透传给前端展示。

图 4-4 练习历史与 AI 试题解析

4.5 错题积累与收藏管理模块

为了帮助用户巩固薄弱知识点，系统设计了错题自动积累与手动收藏功能。该模块连接了练习模块与题目实体，实现了个性化的学习路径。

4.5.1 错题自动判别与入库

系统采用"练测联动"的策略。在用户提交练习试卷时，后端不仅计算总分，还会遍历每一道题目的作答情况。

- **判别逻辑**：Service 层对比用户提交的答案 (userAnswer) 与数据库标准答案 (standardAnswer)。
- **入库操作**：一旦判定答案错误，系统会自动构建 QuestionCollection 对象，标记 type 字段（0 代表选择题，1 代表填空题），并调用 Mapper 接口将其插入数据库。

4.5.2 收藏列表的多态查询

由于系统中的题目数据分散在 QuestionSelect（选择题表）和 QuestionFill（填空题表）中，收藏列表的查询需要处理“多表聚合”的逻辑。

- **实现思路**：
    1. 首先根据 userId 查询 QuestionCollection 表，获取该用户所有收藏记录的列表（包含 questionId 和 type）。
    2. 后端根据 type 字段进行分流：
        - 若 type=0，将其 ID 放入选择题 ID 集合。
        - 若 type=1，将其 ID 放入填空题 ID 集合。
    3. 分别查询 QuestionSelect 和 QuestionFill 表，获取具体的题干与选项信息。
    4. 最后在内存中将数据组装成统一的 DTO（数据传输对象）返回给前端，从而在界面上无缝展示不同类型的题目。

4.6 数据统计与仪表盘

系统首页为用户提供了多维度的学习数据统计。

- **实现方式**：后端编写聚合查询 SQL（Aggregation），统计用户的总做题数、正确率（Correct Count / Total Count）、注册天数等指标，以及对所有注册用户进行排位和横向比较。
- **数据可视化**：前端接收 JSON 数据后，渲染出直观的数字仪表盘。

第五章 前端实现与系统测试

5.1 前端实现综述

前端采用 Vue.js 框架开发，配合 Element UI 组件库。

- **路由管理**：使用 Vue Router 管理 /login, /practice, /history 等页面跳转。
- **状态管理**：利用 Vuex 存储用户登录状态及 Token，实现组件间的数据共享。
- **收藏与历史**：前端复用了试题展示组件，方便用户查看历史错题。

5.2 系统环境与部署

本项目遵循严格的开发流程：

- **版本控制**：使用 Git 进行代码管理，托管于 Gitee 平台。
  - **分支管理策略**：
    - **主分支 (develop)**：保护分支，禁止直接推送。
    - **开发分支命名**：develop_user_yyyymmdd
  - **代码合入流程**：
    - 必须提交 Pull Request (PR)。
    - 经过代码评审 (Code Review) 后方可合入。
  - **问题追踪**：使用 Issue 跟踪 Bug 及前后端联调进度。
- **部署环境**：
  - **服务器**：
    - **Alpha 测试**：USTC Vlab (Ubuntu 24.04)
    - **Beta 测试**：阿里云 ECS (CentOS 7)。
  - **数据库**：阿里云 RDS MySQL。
  - **中间件**：阿里云 Tair (Redis)。
  - **Web 服务**：Nginx 反向代理，将 80 端口请求转发至后端 8080 端口。
  - **AI 模型**：使用反向代理，部署在阿里云 ECS。

第六章 总结与展望

本文设计并实现了一个基于 SpringBoot 的全栈在线考试系统“KnowledgeOunce”。系统通过合理的后端架构设计，实现了从用户注册、题库管理到在线考试、智能分析的完整业务闭环。

代码开源地址：https://gitee.com/charon_az/knowledge-ounce

主要工作总结：

1. **架构稳健：**采用 SpringBoot + MyBatis + Redis 及前后端分离的主流架构，保证了系统的可维护性与高性能。
2. **功能完备：**实现了 Excel 导入、自动组卷、自动评分等核心考试功能。
3. **智能化创新：**成功接入 DeepSeek 大模型，解决了传统考试系统缺乏个性化解析的痛点。

未来展望：

后续将考虑引入 Spring Cloud 微服务架构以支持更高并发；增加主观题及 AI 自动评分功能；后端采用集群部署以减轻 AI 大模型反向代理的压力；并开发移动端原生 App 以适配更多使用场景。

参考文献

[1] VMware Inc. Spring Boot Reference Guide[EB/OL]. (2024)[2025-12-30]. https://docs.spring.io/spring-boot/index.html.

[2] Vue.js Core Team. Vue.js: The Progressive JavaScript Framework[EB/OL]. [2025-12-30]. https://vuejs.org/.

[3] iidamie. DeepSeek2API: A Reverse Proxy for DeepSeek API[EB/OL]. (2025)[2025-12-30]. https://github.com/iidamie/deepseek2api.