

# 目录

<b>第一章 引言</b>	<b>4</b>
1.1 编写目的 . . . . .	4
1.2 背景 . . . . .	4
1.2.1 项目背景 . . . . .	4
1.3 参考资料 . . . . .	4
1.3.1 参考标准 . . . . .	4
1.3.2 参考文档 . . . . .	5
1.4 概述 . . . . .	5
1.4.1 文档结构 . . . . .	5
1.4.2 设计约束 . . . . .	5
<b>第二章 系统概述</b>	<b>6</b>
2.1 系统定位 . . . . .	6
2.2 主要功能 . . . . .	6
2.2.1 智能语音识别与评估 . . . . .	6
2.2.2 个性化训练任务 . . . . .	6
2.2.3 AI助手互动 . . . . .	6
2.2.4 学习进度管理 . . . . .	6
2.3 设计基础 . . . . .	6
2.3.1 教育学理论基础 . . . . .	6
2.3.2 技术支持 . . . . .	7
2.4 创新特色 . . . . .	7
2.4.1 三维度评估体系 . . . . .	7
2.4.2 智能AI助手 . . . . .	7
2.4.3 学习闭环设计 . . . . .	7
2.5 设计依据 . . . . .	7
2.5.1 数据/类设计 . . . . .	7
2.5.2 总体架构设计 . . . . .	8
2.5.3 接口设计 . . . . .	8
2.5.4 构件设计 . . . . .	8
2.5.5 设计原则 . . . . .	8
<b>第三章 系统架构</b>	<b>10</b>
3.1 架构设计 . . . . .	10
3.1.1 技术栈选择 . . . . .	10
3.1.2 模块间交互 . . . . .	10
3.1.3 设计理念 . . . . .	11
3.2 分层设计 . . . . .	11
3.3 表达小达人系统架构概述 . . . . .	12

<b>第四章 数据设计</b>	<b>14</b>
4.1 类设计 . . . . .	14
4.1.1 用户模块 . . . . .	14
4.1.2 语音识别模块 . . . . .	17
4.1.3 AI模块 . . . . .	19
4.1.4 任务管理模块 . . . . .	22
4.1.5 安全模块 . . . . .	24
4.2 数据库设计 . . . . .	26
4.2.1 数据库概述 . . . . .	26
4.2.2 表结构设计 . . . . .	27
4.2.3 关系约束 . . . . .	28
4.2.4 存储引擎 . . . . .	29
4.2.5 字符集 . . . . .	29
<b>第五章 构件设计</b>	<b>30</b>
5.1 AI辅助构件 . . . . .	30
5.1.1 组件构成 . . . . .	30
5.1.2 活动流程 . . . . .	31
5.1.3 组件交互 . . . . .	33
5.2 语音识别构件 . . . . .	33
5.2.1 核心构件组成 . . . . .	33
5.2.2 工作流程 . . . . .	34
5.2.3 关键技术实现 . . . . .	35
5.3 系统核心构件 . . . . .	35
5.3.1 任务管理构件 . . . . .	35
5.3.2 用户管理构件 . . . . .	36
5.3.3 安全管理构件 . . . . .	36
5.3.4 构件交互 . . . . .	37
<b>第六章 接口设计</b>	<b>38</b>
6.1 内部接口设计 . . . . .	38
6.1.1 用户模块接口设计 . . . . .	38
6.1.2 AI模块接口设计 . . . . .	40
6.1.3 语音识别模块接口设计 . . . . .	42
6.1.4 任务模块接口设计 . . . . .	43
6.2 外部接口设计 . . . . .	46
6.2.1 WebSocket接口与讯飞服务交互 . . . . .	46
6.2.2 智谱AI对话接口 . . . . .	47
<b>第七章 用户界面设计</b>	<b>50</b>
7.1 注册与登录 . . . . .	50
7.2 任务中心 . . . . .	51
7.3 任务学习 . . . . .	52
7.4 个人中心 . . . . .	53
7.5 历史记录 . . . . .	54
7.6 今日待办 . . . . .	55

<b>第八章 系统部署</b>	<b>56</b>
8.1 系统部署概述 . . . . .	56
8.2 部署环境说明 . . . . .	56
8.2.1 硬件环境 . . . . .	56
8.2.2 软件环境 . . . . .	56
8.3 部署流程 . . . . .	56
8.3.1 后端服务器部署 . . . . .	56
8.3.2 数据库服务器部署 . . . . .	57
8.3.3 前端服务器部署 . . . . .	58
8.4 系统部署图 . . . . .	58
8.5 安全配置 . . . . .	59
8.5.1 防火墙配置 . . . . .	59
8.6 部署检查清单 . . . . .	59

# 第一章 引言

## 1.1 编写目的

本设计说明书面向开发人员、测试人员及项目相关干系人，详细说明表达小达人系统的设计构架和实现细节。文档旨在：

- 明确系统的技术实现方案和架构设计
- 指导开发团队进行详细设计和编码实现
- 作为系统维护和升级的依据文档
- 便于项目相关人员理解系统结构和各模块的功能设计

## 1.2 背景

### 1.2.1 项目背景

在当今信息化时代，教育领域正经历着深刻的变革。”表达小达人”项目应运而生，旨在为低年级学生提供一个创新的语言表达学习平台。本项目的提出源于对当前语文教育现状的深入观察和思考。传统语文教育面临着诸多挑战。首先，课堂教学过分强调知识的单向灌输，忽视了学生主动探索和实践的重要性。特别是在语言表达和逻辑思维的培养方面，现有教学模式存在明显短板。其次，由于班级规模大、课时有限等客观因素，教师难以为每位学生提供充分的口语表达机会和个性化指导。再者，缺乏及时有效的反馈机制，使得学生难以准确认识自己的表达优劣，影响了学习效果的提升。这些问题在低年级阶段表现得尤为突出。研究表明，小学低年级是语言表达能力和逻辑思维发展的关键期，这一阶段的教育对学生未来的学习和发展具有重要影响。然而，现有的教育资源和方式难以满足学生在这一关键时期的学习需求。传统的练习方式往往枯燥单调，难以激发学生的学习兴趣；标准化的教学内容也无法适应不同学生的个性化需求。随着人工智能技术的快速发展，特别是大语言模型在自然语言处理领域取得的突破性进展，为解决这些教育难题提供了新的可能。大语言模型不仅具备强大的语言理解和生成能力，还能够进行智能化的评估和反馈，这为个性化教育的实现创造了技术基础。同时，教育信息化的深入推进也为新型教育平台的应用提供了良好的环境支持。本项目正是在这样的背景下，充分利用人工智能技术的优势，通过设计多样化的任务和智能反馈机制，为学生创造一个沉浸式的语言学习环境。系统能够实时分析学生的表达内容，提供个性化的学习建议，帮助学生循序渐进地提升表达能力。通过融合教育理论、人工智能技术和游戏化学习元素，项目致力于打造一个既能激发学生学习兴趣，又能确保学习效果的智能教育平台。

## 1.3 参考资料

### 1.3.1 参考标准

- GB/T 8567-2006 计算机软件文档编制规范
- IEEE 1016-2009 软件设计描述
- GB/T 9385-2008 计算机软件需求规格说明规范

### 1.3.2 参考文档

- Robertson, James Robertson, Suzanne-Mastering the requirements process-Appendix A.pdf
- SDD-IEEE-1016-2009+软件设计规约.pdf
- 《表达小达人项目需求规约说明书》V1.3版

## 1.4 概述

### 1.4.1 文档结构

本文档主要包含以下内容：

1. 第一章：引言
2. 第二章：系统概述
3. 第三章：系统架构
4. 第四章：数据设计
5. 第五章：构件设计
6. 第六章：接口设计
7. 第七章：用户界面设计
8. 第八章：系统部署

### 1.4.2 设计约束

在系统设计过程中需要考虑以下约束：

- 技术约束
  - 采用Web技术栈开发
  - 支持主流浏览器
  - 确保系统响应速度
- 开发约束
  - 开发周期：3个月
  - 开发人员：4人
  - 采用增量开发模型
- 运行环境约束
  - 支持Windows/macOS等主流操作系统
  - 需要稳定的网络连接
  - 设备需配备麦克风

## 第二章 系统概述

### 2.1 系统定位

表达小能手是一个基于人工智能技术的口语表达能力训练系统。该系统旨在通过语音识别、自然语言处理等技术，为用户提供个性化的口语表达训练服务。系统重点关注表达的整体性、逻辑性和情感性三个维度，通过AI助手的即时反馈和建议，帮助用户全面提升口语表达能力。

### 2.2 主要功能

#### 2.2.1 智能语音识别与评估

- 实时语音输入识别，准确将用户的语音转换为文本
- 基于完整性、逻辑性、情感性三个维度进行智能评估
- 提供改进建议

#### 2.2.2 个性化训练任务

- 提供多样化的口语表达训练任务

#### 2.2.3 AI助手互动

- 智能AI助手提供即时反馈和指导
- 针对表达中的问题给出具体的改进建议
- 提供相关的表达技巧和提示词

#### 2.2.4 学习进度管理

- 记录和追踪用户的训练历史
- 展示学习成果和进步情况
- 规划今日待办任务

### 2.3 设计基础

#### 2.3.1 教育学理论基础

- 建构主义学习理论：强调学习者通过主动实践和反思来构建知识
- 即时反馈原则：及时的反馈有助于纠正错误，强化正确的表达方式
- 渐进式学习：按照难度递进的方式安排训练内容

### 2.3.2 技术支持

- 语音识别技术：准确将语音转换为文本，支持实时识别
- 自然语言处理：分析语言表达的完整性、逻辑性和情感特征

## 2.4 创新特色

### 2.4.1 三维度评估体系

系统创新性地提出了完整性、逻辑性、情感性三个维度的评估体系，全面衡量用户的口语表达能力。这种多维度的评估方式有助于用户找到自己的优势和不足，有针对性地进行训练。

### 2.4.2 智能AI助手

系统配备的AI助手不仅能够提供即时反馈，还能根据用户的表达特点给出个性化的建议和提示词。这种智能交互方式使训练过程更加生动有效。

### 2.4.3 学习闭环设计

系统通过任务中心、训练过程、历史记录和今日待办等功能模块，构建了完整的学习闭环。用户可以清晰地了解自己的学习进度，并根据系统建议合理安排训练计划。

## 2.5 设计依据

系统设计基于需求分析，并能够追溯到相应的设计依据。系统的主要设计依据如图2.1所示。

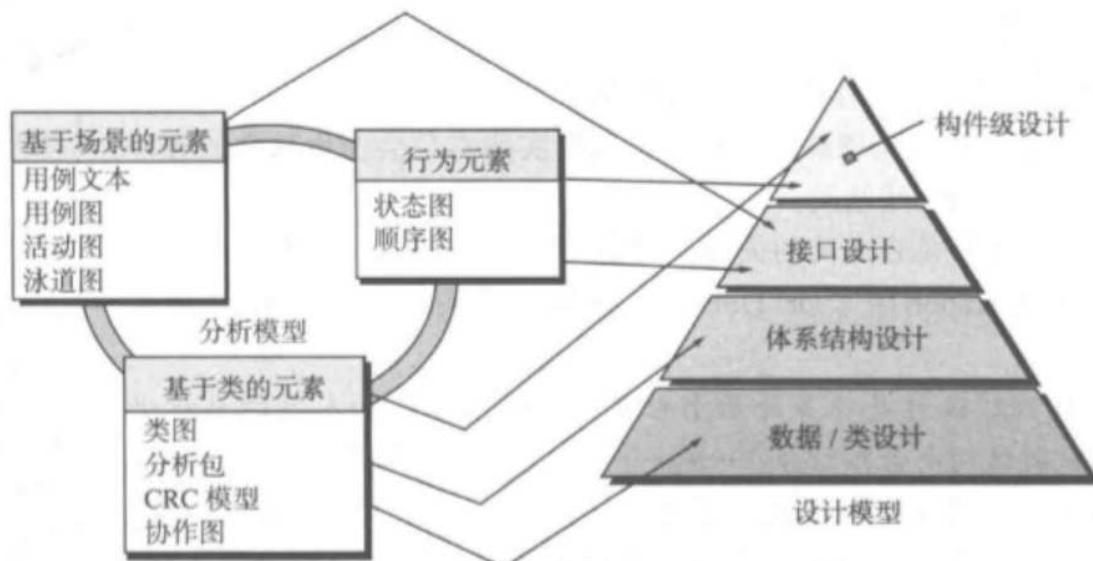


图 2.1: 从需求模型到设计模型的转换

图 2.1: 系统设计依据

### 2.5.1 数据/类设计

- **类图设计：**通过定义系统的主要类，指导代码实现的设计。其中包括：
  - 用户类（User）：包含用户的基本信息，如姓名、电子邮件、密码等
  - 任务类（Task）：定义训练任务的相关属性和方法

### 2.5.2 总体架构设计

- 系统架构图:
  - 综合系统的功能需求和性能目标，确定系统的块划分和整体架构
  - 系统架构设计规划各个模块及其交互
  - 对于用户管理模块，架构能够处理大量用户的注册、登录及权限分配
- 类图:
  - 展示不同类之间的交互和关系，特别是与用户、角色、权限相关的类
  - 展示对象之间如何进行交互及其数据流动
  - 明确类之间的依赖关系和通信方式

### 2.5.3 接口设计

- 用例图:
  - 分析系统中实际使用场景来设计系统间的交互接口
  - 确保接口能够满足用户的实际操作需求
  - 定义系统与外部实体的交互边界
- 顺序图:
  - 确保接口能够支持系统的核心功能
  - 具备相应的操作逻辑
  - 展示系统组件间的时序交互

### 2.5.4 构件设计

- 类图:
  - 关注类的职责、关系以及如何将它们组织成具有明确功能的构件
  - 定义构件的内部结构和对外接口
  - 确保构件的功能完整性和独立性
- 顺序图:
  - 明确系统在不同场景下的动态行为
  - 设计每个构件的执行流程和数据流
  - 展示构件间的交互序列和消息传递

### 2.5.5 设计原则

每个构件的设计都遵循以下原则：

- 高内聚:
  - 确保模块内部功能完整、独立
  - 相关功能集中在同一构件中
  - 提高代码的可维护性和复用性
- 低耦合:

- 模块之间的依赖关系尽量少
- 通过清晰的接口进行通信
- 降低系统各部分间的相互影响

# 第三章 系统架构

## 3.1 架构设计

### 3.1.1 技术栈选择

- **前端:** 使用Vue.js框架来实现前端页面和交互。
- **后端:** 后端使用SpringBoot，基于Java的SpringBoot框架能够快速构建生产级别的应用，简化了配置和开发流程，适合构建微服务架构。SpringBoot提供了强大的安全性控制以及高效的API开发支持。
- **数据库:** 使用MySQL作为关系型数据库，保证数据的一致性和完整性。MySQL适合存储用户信息、权限数据等关系型数据。

### 3.1.2 模块间交互

#### RESTful API通信

系统中的不同模块主要通过RESTful API进行交互。具体来说：

- 前端使用Vue.js通过REST API向后端发起请求
- 后端（SpringBoot）处理逻辑并通过RESTful API将响应返回给前端
- API将用户管理功能与其他模块（如教学活动、成绩管理等）解耦，确保了系统的灵活性和可扩展性

RESTful API是轻量级的通信方式，遵循HTTP协议的标准方法（GET、POST、PUT、DELETE）进行资源的操作。这使得用户管理模块能够与系统其他模块高效交互，同时提供简单清晰的接口进行功能扩展。

#### WebSocket实时通信

除了传统的HTTP请求响应模式，系统还使用WebSocket实现实时双向通信：

- **语音识别实时反馈**
  - 用户进行语音输入时，通过WebSocket将语音数据实时传输到服务器
  - 服务器进行实时语音识别处理
  - 将识别结果即时反馈给前端显示
- **AI实时互动**
  - AI系统通过WebSocket实时接收用户的语音输入
  - 进行实时分析和评估
  - 将评估结果和建议即时推送给用户
- **技术实现**
  - 前端：使用原生WebSocket API或Socket.io库
  - 后端：使用Spring WebSocket支持
  - 保持长连接，确保实时通信的稳定性

### 3.1.3 设计理念

#### 分层设计

系统采用分层架构，将功能模块按职责分为不同层次：

- 用户交互层
- 安全层
- 应用逻辑层
- 应用数据层
- 数据源

这种分层设计确保模块间低耦合并提高系统的可维护性。

#### JWT认证与授权

系统采用JWT（JSON Web Token）来实现用户身份验证和权限控制。JWT使得每个请求都包含安全令牌，确保每次请求都是经过授权的。

#### 可扩展性与模块化

- 用户管理模块与系统中的其他模块解耦
- 采用模块化设计，便于未来扩展新功能
- 通过标准化的接口实现模块间的通信
- 支持水平扩展以应对用户量增长

## 3.2 分层设计

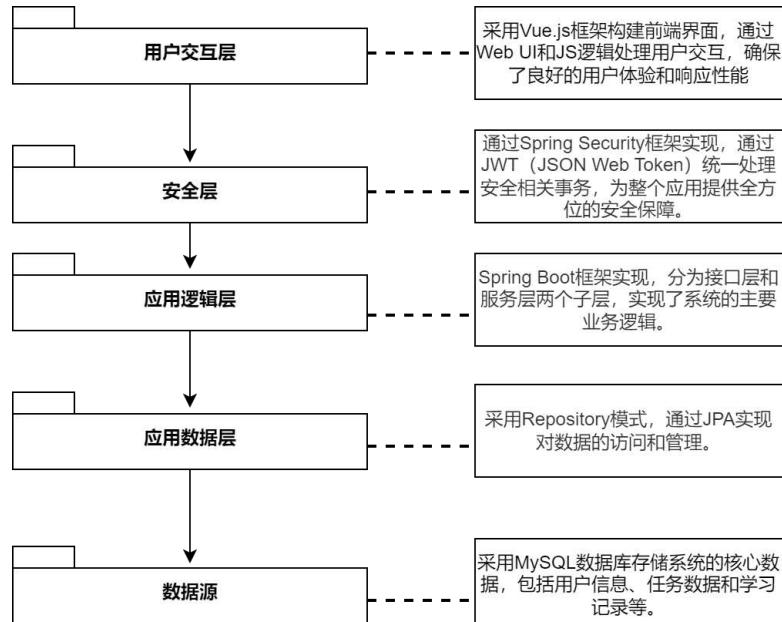


图 3.1: 逻辑视图

完成系统架构的分析之后，我们在传统的五层架构的基础上进行了拆分和优化，重新实现了更加详细的系统建模，我们的系统建模如下所示：

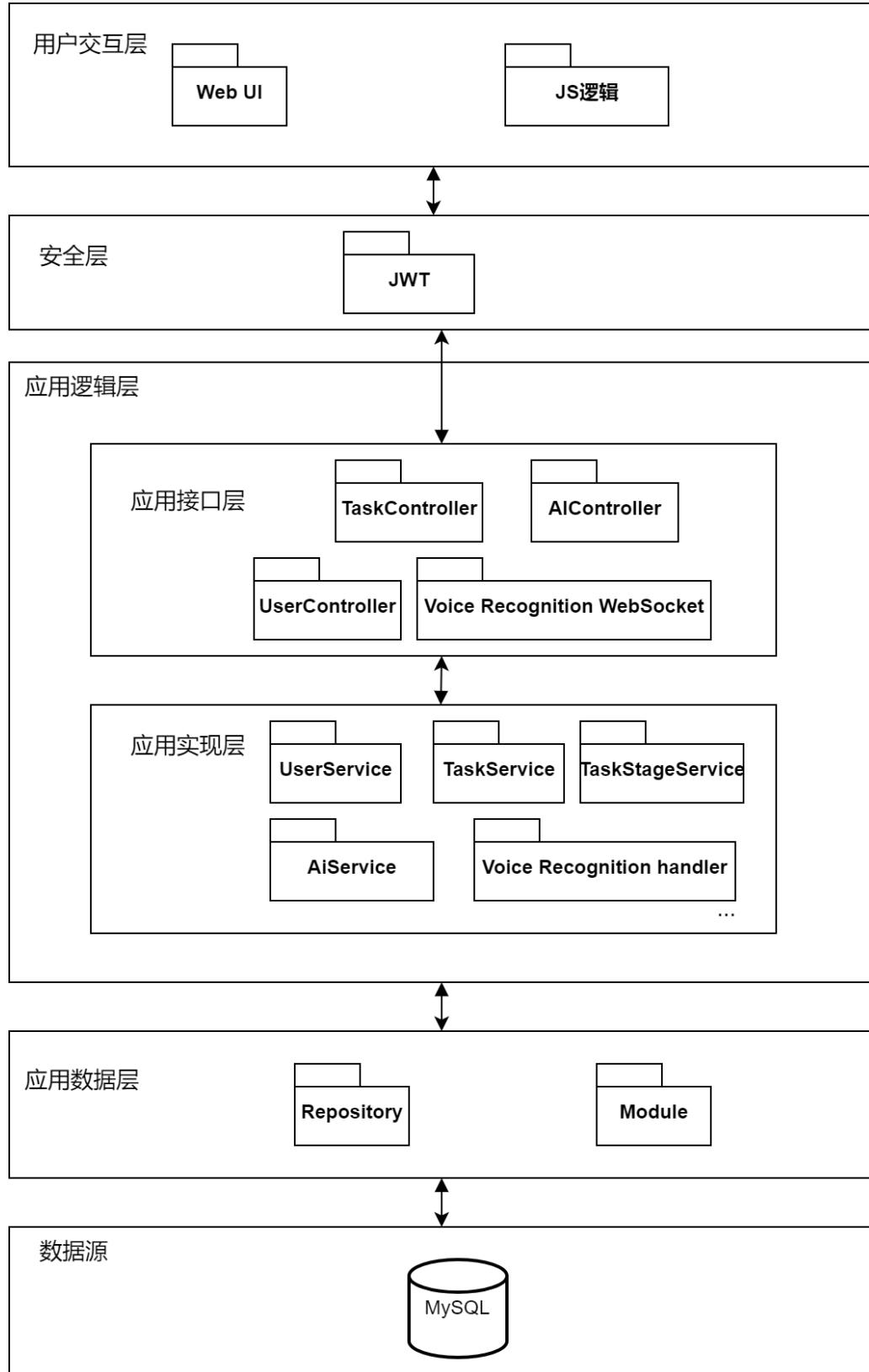


图 3.2: 系统架构图

### 3.3 表达小达人系统架构概述

表达小达人采用经典的多层架构设计，整体分为用户交互层、安全层、应用逻辑层、应用数据层和数据源层四个主要层次。在用户交互层，系统采用Vue.js框架构建前端界面，通过Web UI和JS逻辑处理用户交互，确保了良好的用户体验和响应性能。

安全层通过Spring Security框架实现，位于用户交互层与应用逻辑层之间。该层通过JWT（JSON Web To-

ken) 实现无状态的用户认证机制，配合细粒度的权限控制体系，确保系统的访问安全。安全层统一处理用户认证、授权验证、跨域请求控制(CORS)等安全相关事务，为整个应用提供全方位的安全保障。

应用逻辑层是系统的核心，采用Spring Boot框架实现，分为接口层和服务层两个子层。接口层包含TaskController、AIController、UserController等控制器组件，以及处理实时语音识别的WebSocket接口，负责接收和处理来自前端的各类请求。服务层则包含了UserService、TaskService、TaskStageService等核心业务处理单元，以及AI服务和语音识别处理器，实现了系统的主要业务逻辑。

应用数据层采用Repository模式，通过JPA实现对数据的访问和管理。系统的所有持久化数据最终存储在MySQL数据库中，确保了数据的可靠性和一致性。通过合理的数据模型设计，系统可以高效地存储和检索用户信息、任务数据、学习记录等各类数据。

这种多层架构设计不仅使系统具备了良好的功能性和可用性，同时也为未来功能扩展和性能优化预留了空间。通过合理的技术选型和架构设计，表达小达人系统能够有效支持小学生的表达能力培养，提供个性化的学习体验。

## 第四章 数据设计

## 4.1 类设计

总类图如下：

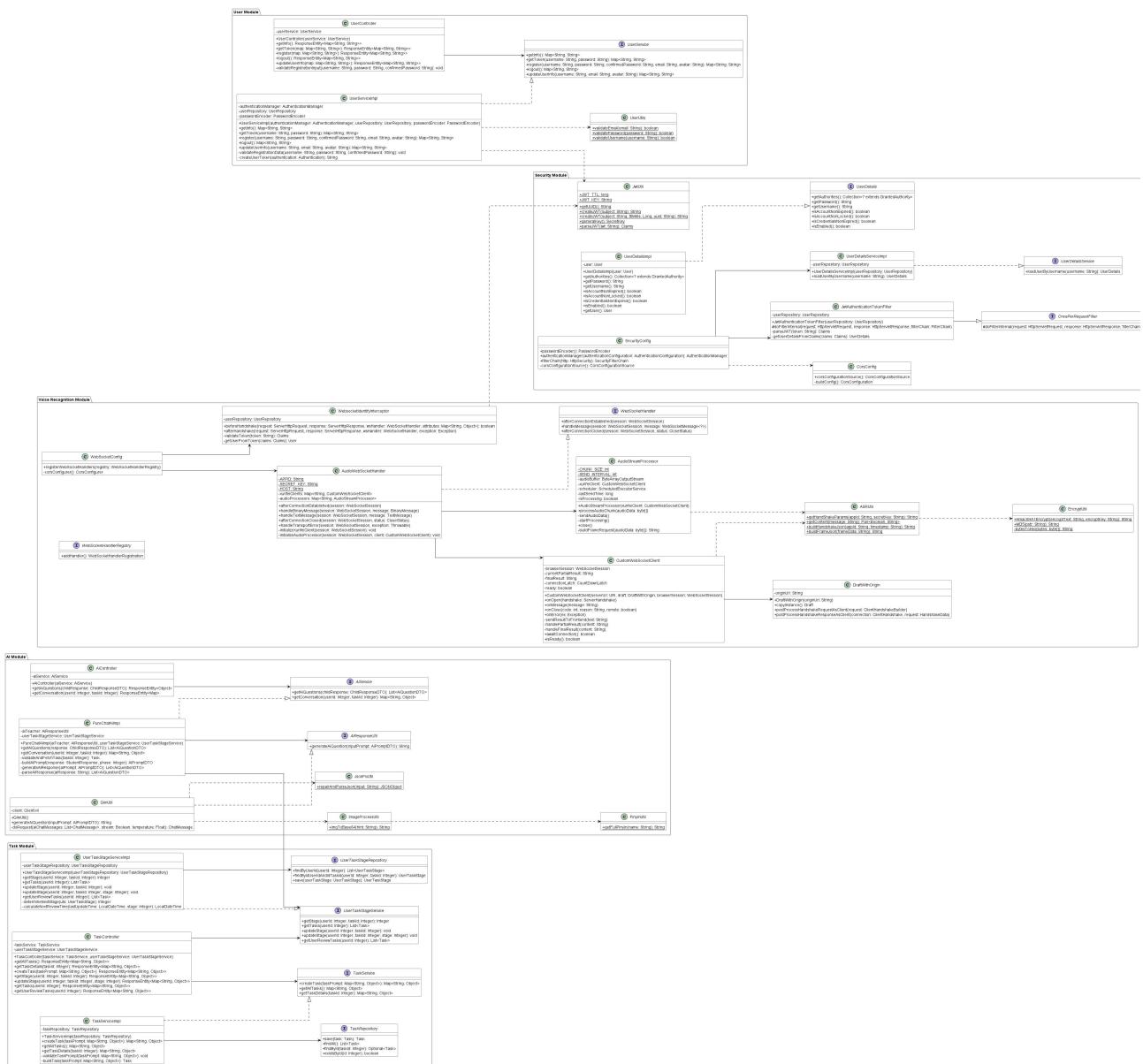


图 4.1: 系统总类图

### 4.1.1 用户模块

用户模块采用标准的分层架构，通过UserController处理HTTP请求，UserService提供业务逻辑，UserRepository实现数据访问，以User实体为核心。模块支持用户注册、登录等基础功能，采用DTO模式进行数据传输，实现了

用户信息与任务进度的关联管理。所有密码数据都经过加密处理，确保了用户数据的安全性。该模块为整个系统提供了完整的用户管理和认证支持。

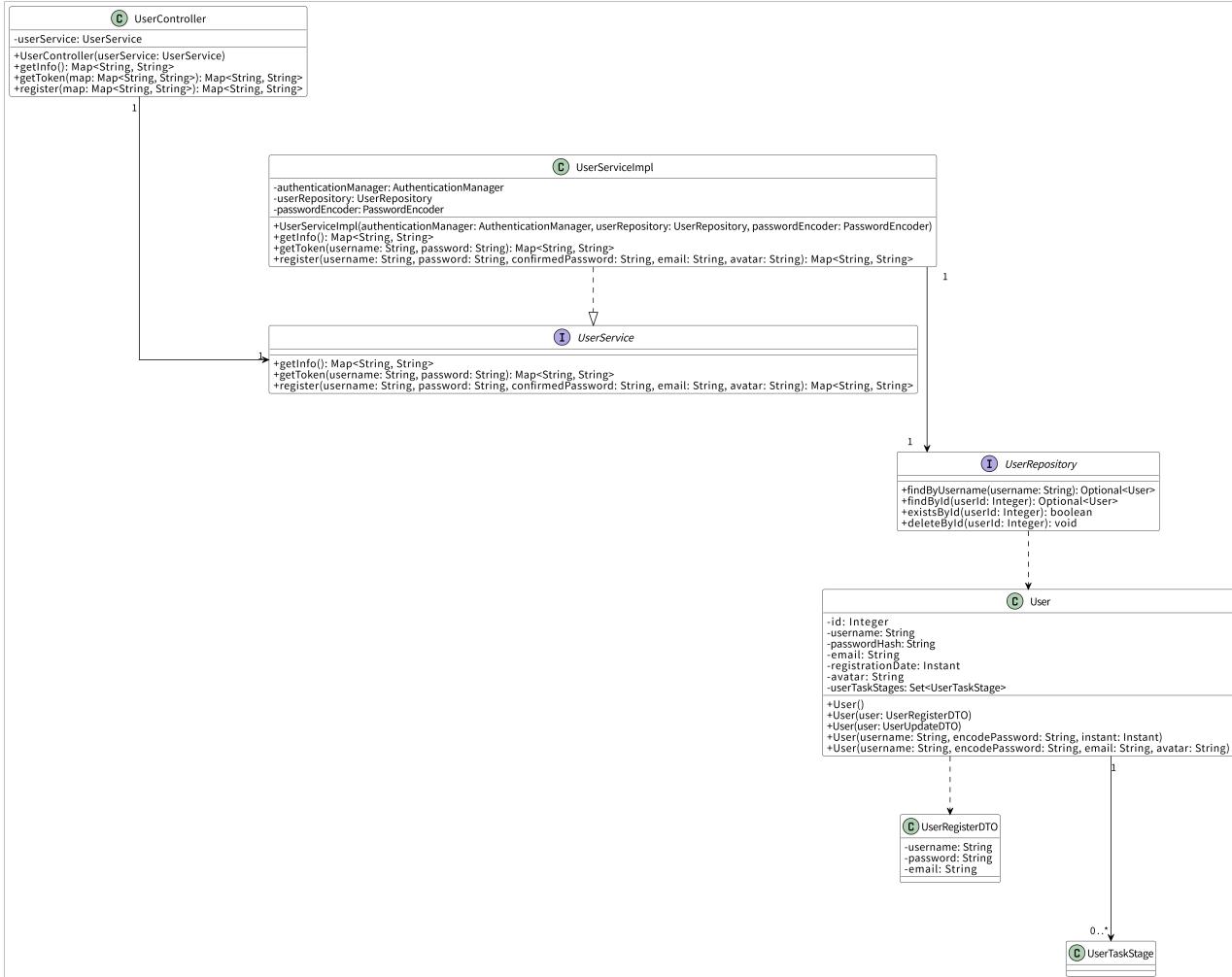


图 4.2: 用户模块类图

### UserController类

属性	
userService: UserService	用户服务接口的实例
方法	
getInfo(): Map<String, String>	获取当前用户信息
getToken(Map<String, String>): Map<String, String>	用户登录并获取令牌
register(Map<String, String>): Map<String, String>	处理用户注册请求

### UserService接口

方法	
getInfo():Map<String, String>	获取用户信息
getToken(username: String, password: String): Map<String, String>	用户登录认证

register(username,password, confirmedPassword,email, avatar):Map<String, String>	用户注册
--	------

### UserServiceImpl类

属性	
authenticationManager: AuthenticationManager	认证管理器
userRepository: UserRepository	用户数据访问接口
passwordEncoder:PasswordEncoder	密码加密工具

### UserRepository接口

方法	
findByUsername(username: String): Optional<User>	根据用户名查找用户
findById(userId:Integer): Optional<User>	根据ID查找用户
existsById(userId:Integer): boolean	检查用户ID是否存在
deleteById(userId:Integer): void	删除指定用户

### User类

属性	
id: Integer	用户唯一标识符
username: String	用户名
passwordHash: String	加密后的密码
email: String	电子邮件
registrationDate: Instant	注册时间
avatar: String	用户头像
userTaskStages:Set<UserTaskStage>	用户任务阶段集合
构造方法	
User()	默认构造函数
User(UserRegisterDTO)	基于注册DTO创建用户
User(UserUpdateDTO)	基于更新DTO创建用户
User(username, encodePassword, instant)	基本信息构造
User(username, encodePassword, email, avatar)	完整信息构造

### UserRegisterDTO类

属性	
username: String	用户名
password: String	密码

email: String	电子邮件
---------------	------

### 4.1.2 语音识别模块

语音识别模块采用WebSocket实现实时语音传输和识别，通过集成讯飞开放平台的语音识别服务，实现语音到文字的转换。模块包含WebSocket配置、连接管理、音频处理和工具类等组件，实现了用户认证、音频数据处理、实时识别结果返回等功能。整个模块采用异步处理机制，确保了实时语音识别的性能和可靠性。

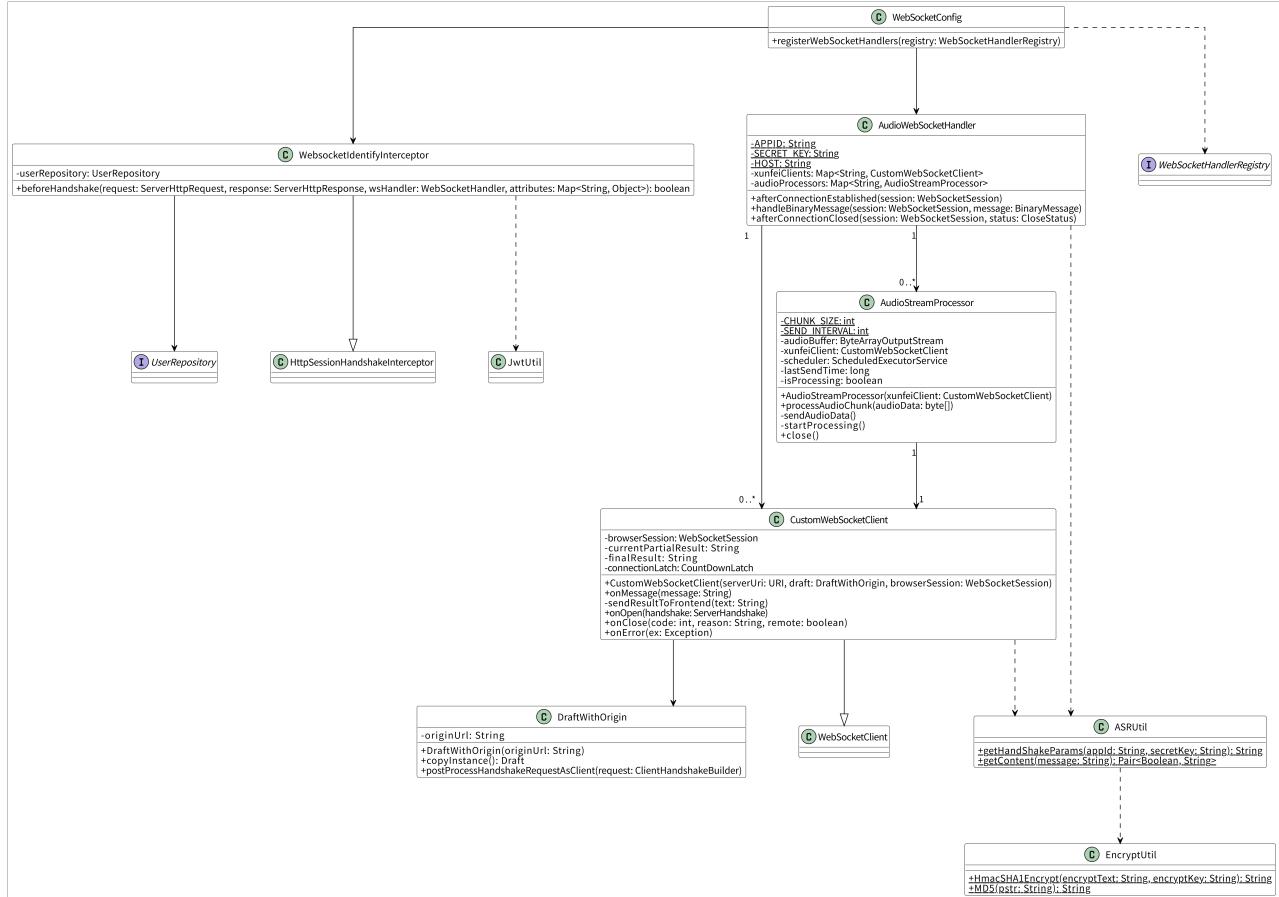


图 4.3: 语音识别模块类图

#### WebSocketConfig类

方法	
registerWebSocketHandlers(registry: WebSocketHandlerRegistry)	注册WebSocket处理器和拦截器

#### WebsocketIdentifyInterceptor类

属性	
userRepository: UserRepository	用户数据访问接口
方法	
beforeHandshake(request, response, wsHandler, attributes): boolean	在握手前进行身份验证

**AudioWebSocketHandler类**

属性	
APPID: String	讯飞平台应用ID
SECRET_KEY: String	讯飞平台密钥
HOST: String	讯飞服务器地址
xunfeiClients:Map<String, CustomWebSocketClient>	讯飞客户端映射
audioProcessors:Map<String, AudioStreamProcessor>	音频处理器映射
方法	
afterConnectionEstablished(session: WebSocketSession)	建立连接时的处理
handleBinaryMessage(session, message)	处理二进制音频数据
afterConnectionClosed(session, status)	关闭连接时的处理

**CustomWebSocketClient类**

属性	
browserSession: WebSocketSession	浏览器WebSocket会话
currentPartialResult: String	当前部分识别结果
finalResult: String	最终识别结果
connectionLatch: CountDownLatch	连接等待锁
方法	
onMessage(message: String)	处理服务器消息
sendResultToFrontend(text: String)	发送结果到前端
onOpen(handshake)	连接建立时的处理
onClose(code, reason, remote)	连接关闭时的处理
onError(ex)	错误处理

**AudioStreamProcessor类**

属性	
CHUNK_SIZE: int	音频数据块大小
SEND_INTERVAL: int	发送间隔
audioBuffer:ByteArrayOutputStream	音频数据缓冲区
xunfeiClient:CustomWebSocketClient	讯飞客户端实例
scheduler:ScheduledExecutorService	定时调度器
方法	
processAudioChunk(audioData: byte[])	处理音频数据块
sendAudioData()	发送音频数据
startProcessing()	开始处理
close()	关闭处理器

## ASRUtil类

方法	
getHandShakeParams(appId: String, secretKey: String): String	获取握手参数
getContent(message:String): Pair<Boolean, String>	解析识别结果内容

## EncryptUtil类

方法	
HmacSHA1Encrypt(encryptText: String, encryptKey: String): String	HMAC-SHA1加密
MD5(pstr: String):String	MD5加密

### DraftWithOrigin类

继承	
继承自Draft_17	
属性	
originUrl: String	WebSocket连接的源地址
构造方法	
DraftWithOrigin(originUrl: String)	使用指定的源地址初始化握手协议
方法	
copyInstance(): Draft	创建当前协议的副本实例
postProcessHandshakeRequest AsClient(request:ClientHandshakeBuilder): ClientHandshakeBuilder	在客户端握手请求中添加Origin头部

#### 4.1.3 AI模块

AI模块是系统的智能交互核心，负责处理学生响应并生成相应的AI问题与提示。该模块采用分层架构，通过与智谱AI大模型的集成，实现了智能问答和对话管理功能。模块支持多阶段对话流程，包含了完整的提示词构建、响应解析和图像处理等功能。

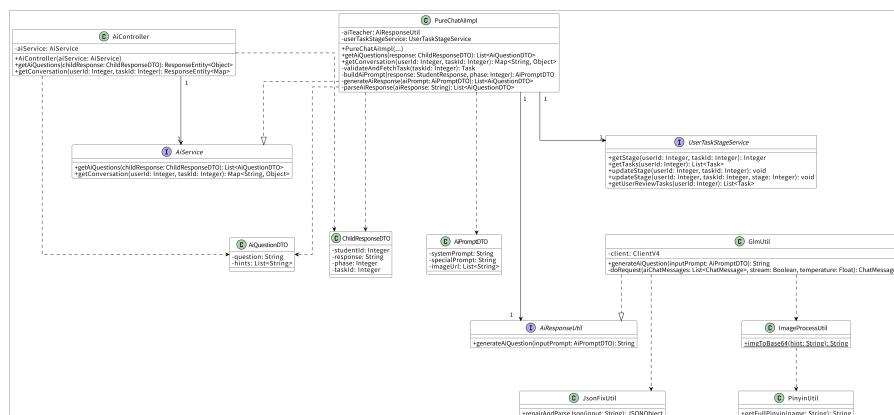


图 4.4: AI模块类图

**AiController类**

属性	
aiService: AiService	AI服务接口实例
方法	
getAiQuestions(childResponse: ChildResponseDTO): ResponseEntity<Object>	获取AI生成的问题
getConversation(userId: Integer, taskId: Integer): ResponseEntity<Map>	获取对话历史

**AiService接口**

方法	
getAiQuestions(childResponse: ChildResponseDTO): List<AiQuestionDTO>	获取AI问题列表
getConversation(userId:Integer, taskId:Integer):Map<String, Object>	获取对话记录

**PureChatAiImpl类**

属性	
aiTeacher: AiResponseUtil	AI响应工具
userTaskStageService: UserTaskStageService	用户任务阶段服务
方法	
getAiQuestions(response: ChildResponseDTO): List<AiQuestionDTO>	实现获取AI问题
getConversation(userId:Integer, taskId: Integer): Map<String, Object>	实现获取对话
validateAndFetchTask(taskId:Integer):Task	验证并获取任务
buildAiPrompt(response:StudentResponse, phase: Integer): AiPromptDTO	构建AI提示
generateAiResponse(aiPrompt: AiPromptDTO): List<AiQuestionDTO>	生成AI响应
parseAiResponse(aiResponse:String): List<AiQuestionDTO>	解析AI响应

**AiResponseUtil接口**

方法	
generateAiQuestion(inputPrompt: AiPromptDTO): String	生成AI问题

**GlmUtil类**

继承	
实现AiResponseUtil接口	
属性	
client: ClientV4	智谱AI客户端
方法	
generateAiQuestion(inputPrompt: AiPromptDTO): String	实现生成AI问题
doRequest(aiChatMessages: List<ChatMessage>, stream: Boolean, temperature: Float): ChatMessage	发送请求

**数据传输对象**

AiPromptDTO类属性	
systemPrompt: String	系统提示词
specialPrompt: String	特殊提示词
imageUrl: List<String>	图片URL列表
AiQuestionDTO类属性	
question: String	问题内容
hints: List<String>	提示列表
ChildResponseDTO类属性	
studentId: Integer	学生ID
response: String	响应内容
phase: Integer	阶段
taskId: Integer	任务ID

**工具类****JsonFixUtil类**

方法	
repairAndParseJson(input: String): JSONObject	修复并解析JSON

**ImageProcessUtil类**

方法	
imgToBase64(hint: String): String	图片转Base64

**PinyinUtil类**

方法	
getFullPinyin(name: String): String	获取完整拼音

#### 4.1.4 任务管理模块

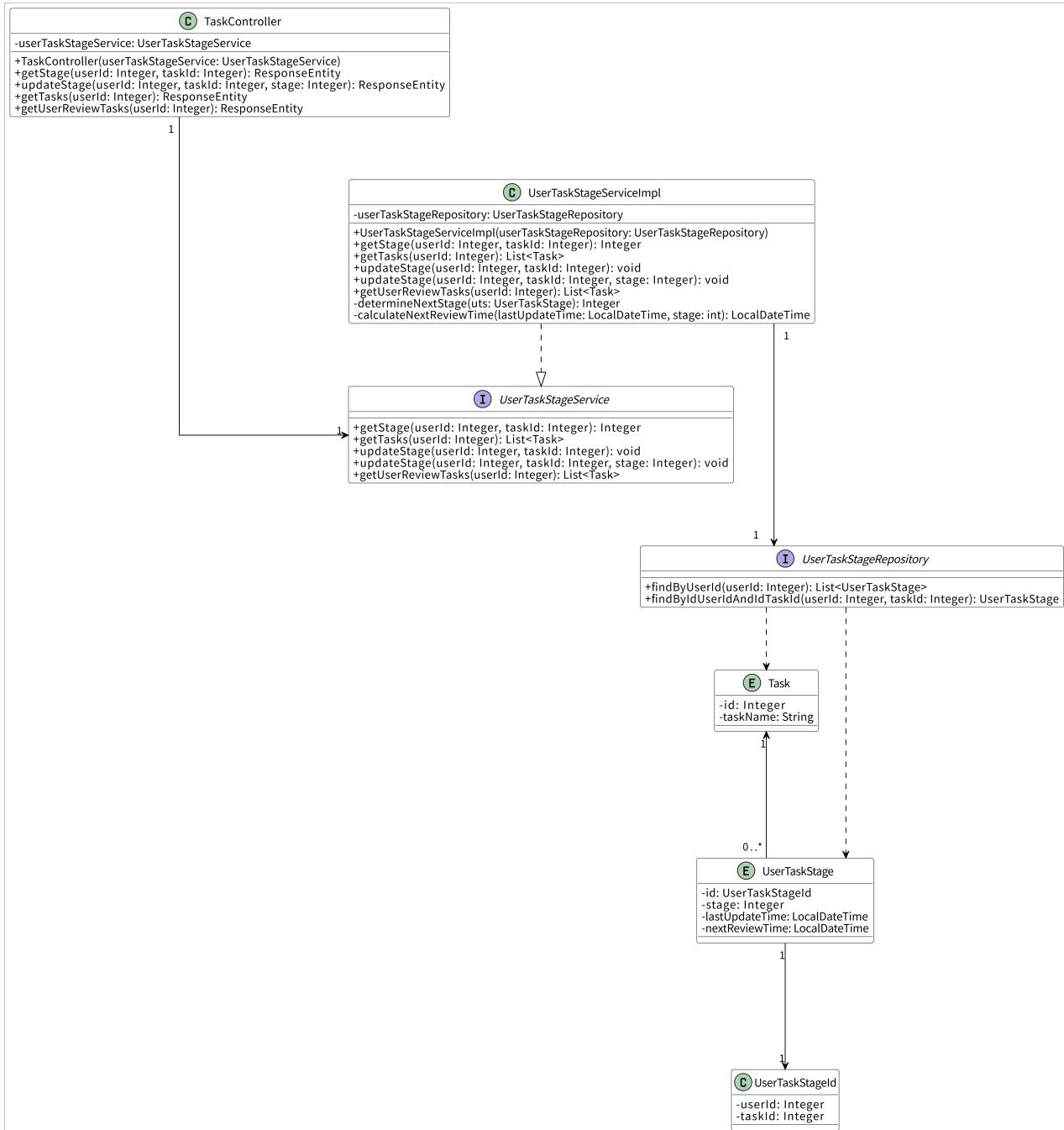


图 4.5: 任务阶段管理类图

#### TaskController类

属性	
userTaskStageService: User- TaskStageService	用户任务阶段服务接口实例

方法	
getStage(userId: Integer, taskId: Integer): ResponseEntity	获取任务阶段
updateStage(userId: Integer, taskId: Integer, stage: Integer): ResponseEntity	更新任务阶段
getTasks(userId: Integer): ResponseEntity	获取用户任务列表
getUserReviewTasks(userId: Integer): ResponseEntity	获取用户复习任务

**UserTaskStageService接口**

方法	
getStage(userId: Integer, taskId: Integer): Integer	获取任务阶段
getTasks(userId: Integer): List<Task>	获取用户任务列表
updateStage(userId: Integer, taskId: Integer): void	更新任务阶段
updateStage(userId: Integer, taskId: Integer, stage: Integer): void	更新指定任务阶段
getUserReviewTasks(userId: Integer): List<Task>	获取用户复习任务

**UserTaskStageServiceImpl类**

属性	
userTaskStageRepository: UserTaskStageRepository	用户任务阶段数据访问接口
方法	
determineNextStage(uts: UserTaskStage): Integer	确定下一阶段
calculateNextReviewTime (lastUpdateTime: LocalDateTime, stage: int): LocalDateTime	计算下次复习时间

**UserTaskStageRepository接口**

方法	
findByIdUserId(userId: Integer): List<UserTaskStage>	查找用户的所有任务阶段
findByIdUserIdAndIdTaskId(userId: Integer, taskId: Integer): UserTaskStage	查找特定用户任务阶段

### UserTaskStage实体类

属性	
id: UserTaskStageId	复合主键
stage: Integer	当前阶段
lastUpdateTime: LocalDateTime	最后更新时间
nextReviewTime: LocalDateTime	下次复习时间

### UserTaskStageId类

属性	
userId: Integer	用户ID
taskId: Integer	任务ID

### 4.1.5 安全模块

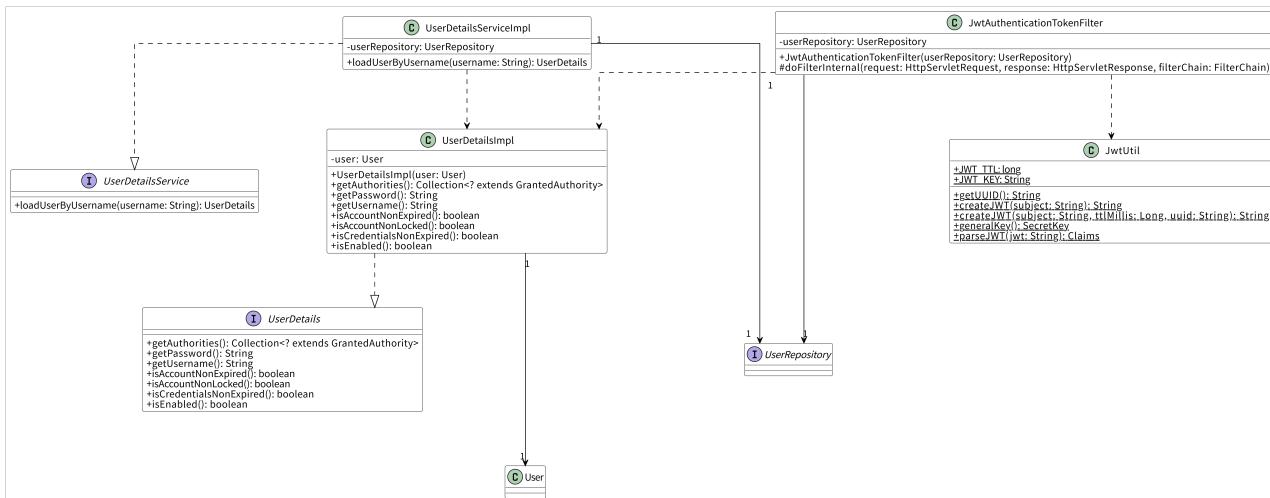


图 4.6: 安全模块类图

安全模块主要负责系统的认证和授权功能，采用JWT（JSON Web Token）实现无状态的用户认证，并通过Spring Security框架实现安全管理。该模块包含JWT处理、用户认证过滤器和用户详情服务等组件。

### SecurityConfig类

注解	
@Configuration	标识为配置类
@EnableWebSecurity	启用Web安全功能
属性	
jwtAuthenticationTokenFilter: JwtAuthenticationTokenFilter	JWT认证过滤器
构造方法	
SecurityConfig (jwtAuthenticationTokenFilter)	通过依赖注入初始化JWT过滤器
方法	

passwordEncoder(): PasswordEncoder	配置密码加密器，使用BCrypt加密算法
securityFilterChain(http: HttpSecurity): SecurityFilterChain	配置安全过滤链
authManager(http: HttpSecurity): AuthenticationManager	配置认证管理器

## JWT相关组件

### JwtUtil类

常量	
JWT_TTL: long	JWT令牌有效期
JWT_KEY: String	JWT密钥
方法	
getUUID(): String	获取UUID
createJWT(subject: String): String	创建JWT令牌
createJWT(subject: String, ttlMillis: Long, uuid: String): String	创建指定参数的JWT令牌
generalKey(): SecretKey	生成密钥
parseJWT(jwt: String): Claims	解析JWT令牌

### JwtAuthenticationTokenFilter类

属性	
userRepository: UserRepository	用户数据访问接口
方法	
doFilterInternal(request: HttpServletRequest, response: HttpServletResponse, filterChain: FilterChain)	内部过滤方法，处理token验证

## 用户认证组件

### UserDetailsImpl类

属性	
user: User	用户实体
方法	
getAuthorities(): Collection<? extends GrantedAuthority>	获取用户权限
getPassword(): String	获取用户密码
getUsername(): String	获取用户名
isAccountNonExpired(): boolean	账号是否未过期
isAccountNonLocked(): boolean	账号是否未锁定
isCredentialsNonExpired(): boolean	凭证是否未过期
isEnabled(): boolean	账号是否启用

### UserDetailsServiceImpl类

属性	
userRepository: UserRepository	用户数据访问接口
方法	
loadUserByUsername(username: String): UserDetails	根据用户名加载用户详情

核心接口

### UserDetails接口

方法	
getAuthorities(): Collection<? extends GrantedAuthority>	获取用户权限
getPassword(): String	获取密码
getUsername(): String	获取用户名
isAccountNonExpired(): boolean	检查账号是否未过期
isAccountNonLocked(): boolean	检查账号是否未锁定
isCredentialsNonExpired(): boolean	检查凭证是否未过期
isEnabled(): boolean	检查账号是否启用

### UserDetailsService接口

方法	
loadUserByUsername(username: String): UserDetails	根据用户名加载用户详情

## 4.2 数据库设计

### 4.2.1 数据库概述

表达小能手数据库设计主要支持任务管理、学生作答和用户管理等功能。系统支持图片任务的创建和管理，学生可以提交作答并获得评分和AI反馈。

### 4.2.2 表结构设计

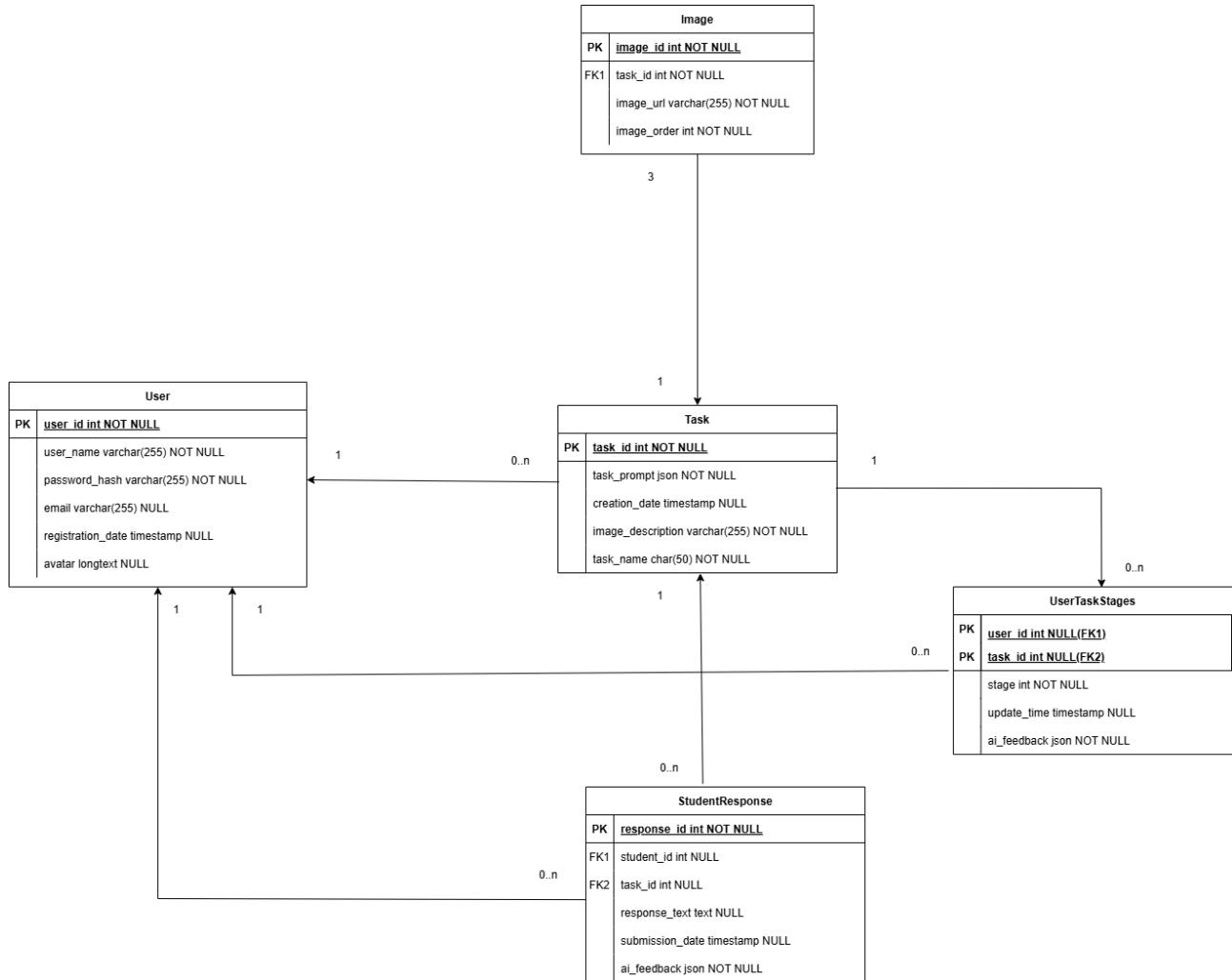


图 4.7: 数据库E-R图

#### user表

字段名	类型	是否为空	键	描述
user_id	int	NOT NULL	PK	用户ID, 自增
username	varchar(255)	NOT NULL	-	用户名
password_hash	varchar(255)	NOT NULL	-	密码哈希值
email	varchar(255)	NULL	-	电子邮箱
registration_date	timestamp	NULL	-	注册时间
avatar	text	NULL	-	用户头像URL

表4.37 用户表结构

#### Task表

字段名	类型	是否为空	键	描述
task_id	int	NOT NULL	PK	任务ID, 自增
task_prompt	json	NOT NULL	-	任务提示/要求

续下页

表 4.38 – 续

字段名	类型	是否为空	键	描述
creation_date	timestamp	NULL	-	创建时间
image_description	varchar(255)	NOT NULL	-	图片描述
task_name	varchar(50)	NOT NULL	-	任务名称

表4.38 任务表结构

**Image表**

字段名	类型	是否为空	键	描述
image_id	int	NOT NULL	PK	图片ID, 自增
task_id	int	NOT NULL	FK	关联的任务ID
image_url	varchar(255)	NOT NULL	-	图片URL
image_order	int	NOT NULL	-	图片顺序

表4.39 图片表结构

**StudentResponse表**

字段名	类型	是否为空	键	描述
response_id	int	NOT NULL	PK	作答ID, 自增
student_id	int	NULL	FK	学生ID
task_id	int	NULL	FK	任务ID
response_text	text	NULL	-	作答内容
score	decimal(5,2)	NULL	-	得分
submission_date	timestamp	NULL	-	提交时间
ai_feedback	json	NOT NULL	-	AI反馈内容

表4.40 学生作答表结构

**UserTaskStages表**

字段名	类型	是否为空	键	描述
user_id	int	NOT NULL	FK/PK	用户ID
task_id	int	NOT NULL	FK/PK	任务ID
stage	int	NOT NULL	-	当前阶段
update_time	timestamp	NULL	-	更新时间

表4.41 用户任务阶段表结构

**4.2.3 关系约束****外键约束**

- Image表: task\_id → Task(task\_id) ON DELETE CASCADE
- StudentResponse表:
  - student\_id → User(user\_id) ON DELETE CASCADE
  - task\_id → Task(task\_id) ON DELETE CASCADE
- UserTaskStages表:

- user\_id → User(user\_id) ON DELETE CASCADE
- task\_id → Task(task\_id) ON DELETE CASCADE

### 唯一约束

UserTaskStages表中(user\_id, task\_id)为组合唯一约束。

### 索引设计

- Image表: task\_id索引
- StudentResponse表:
  - student\_id索引
  - task\_id索引
- UserTaskStages表:
  - task\_id索引
  - (user\_id, task\_id)联合唯一索引

### 4.2.4 存储引擎

所有表均使用InnoDB存储引擎，支持事务处理和外键约束。

### 4.2.5 字符集

使用utf8mb4字符集和utf8mb4\_0900\_ai\_ci排序规则，支持完整的Unicode字符集。

# 第五章 构件设计

下面是系统的总构件图

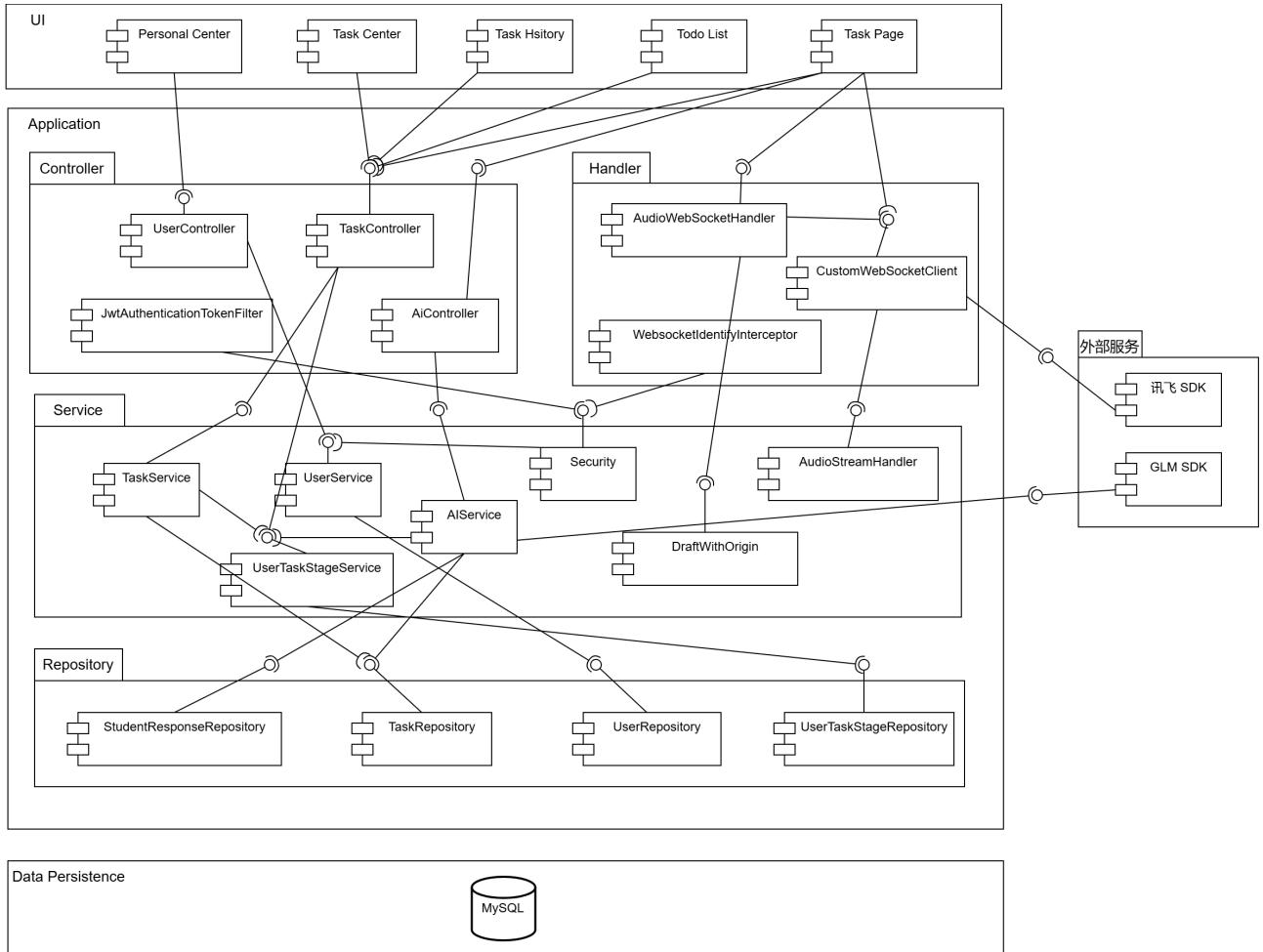


图 5.1: 系统总构件图

## 5.1 AI辅助构件

AI辅助构件是系统中负责智能问答和评估的核心组件，主要包含以下几个关键构件：

### 5.1.1 组件构成

- **AiController**

- 功能: 处理前端AI相关请求，协调AI服务的调用
- 实现: 基于SpringMVC实现RESTful接口，提供问题生成、答案评估等端点

- **AiService**

- 功能: 封装AI核心业务逻辑, 处理问答生成和评估
- 实现: 集成GLM模型, 实现问题生成、答案评估等业务逻辑

- **TaskRepository**

- 功能: 管理训练任务数据的持久化
- 实现: 基于Spring Data JPA实现数据访问层, 处理任务的CRUD操作

- **StudentResponseRepository**

- 功能: 管理学生答题记录的存储
- 实现: 使用Spring Data JPA实现, 负责答题记录的存取

- **GLM SDK**

- 功能: 提供AI模型的核心能力
- 实现: 封装GLM模型API, 提供模型调用接口

### 5.1.2 活动流程

如图5.2所示, AI辅助构件的主要活动流程包括:

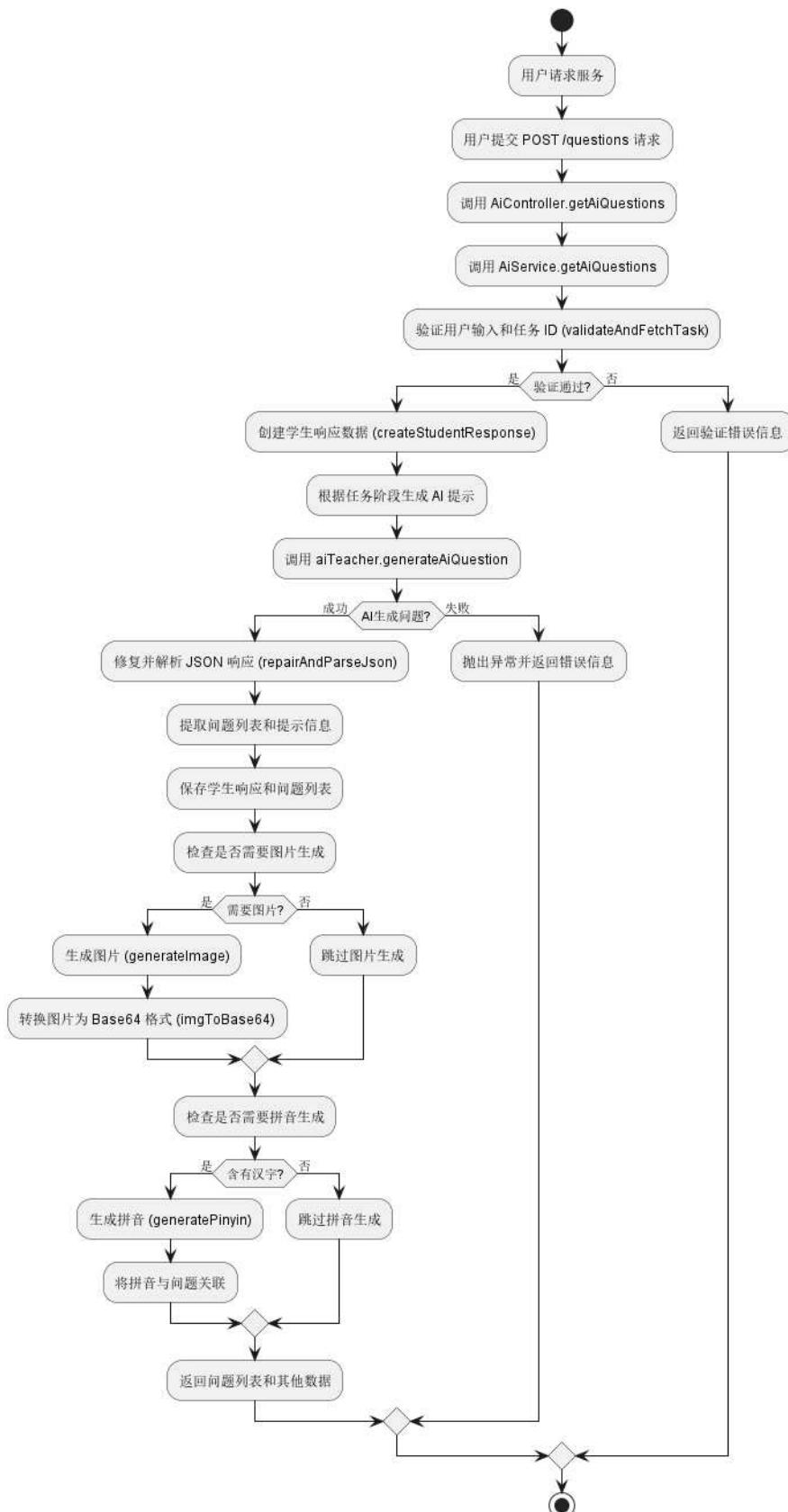


图 5.2: AI辅助构件活动图

1. 用户发起POST /questions请求
  2. AiController接收请求并调用getAiQuestions方法
  3. AiService处理请求，调用validateAndFetchTask验证用户身份
  4. 根据验证结果：

- 验证通过：创建学生响应数据，生成AI提示
- 验证失败：返回错误信息

5. 调用aiTeacher生成问题

6. 处理AI响应：

- 成功：解析JSON响应
- 失败：返回错误信息

7. 生成图片和拼音（如需要）

8. 返回问题列表和其他相关信息

### 5.1.3 组件交互

组件间通过标准接口进行交互：

- Controller层通过依赖注入调用Service层
- Service层调用Repository进行数据操作
- 通过SDK与AI模型进行交互
- 使用统一的数据传输对象（DTO）进行数据交换

## 5.2 语音识别构件

语音识别构件负责实时音频流的处理和语音转文字功能，是系统实现智能语音交互的核心组件。

### 5.2.1 核心构件组成

- **AudioWebSocketHandler**

- 功能：处理WebSocket连接的生命周期和音频数据传输
  - 实现：
    - \* 管理WebSocket会话建立与关闭
    - \* 配置SSL安全连接
    - \* 处理二进制音频消息

- **CustomWebSocketClient**

- 功能：自定义WebSocket客户端，处理与讯飞服务的通信
  - 实现：
    - \* 维护与讯飞服务的连接
    - \* 处理音频数据的发送
    - \* 解析识别结果

- **AudioStreamHandler**

- 功能：音频流处理器，负责音频数据的实时处理
  - 实现：
    - \* 音频数据分块处理
    - \* 实时数据流控制

\* 音频格式转换

- 讯飞SDK

- 功能: 提供语音识别的核心能力
- 实现:
  - \* ASRUTIL工具类封装
  - \* 语音识别参数配置
  - \* 识别结果解析

### 5.2.2 工作流程

如图5.3所示，语音识别构件的工作流程包括：

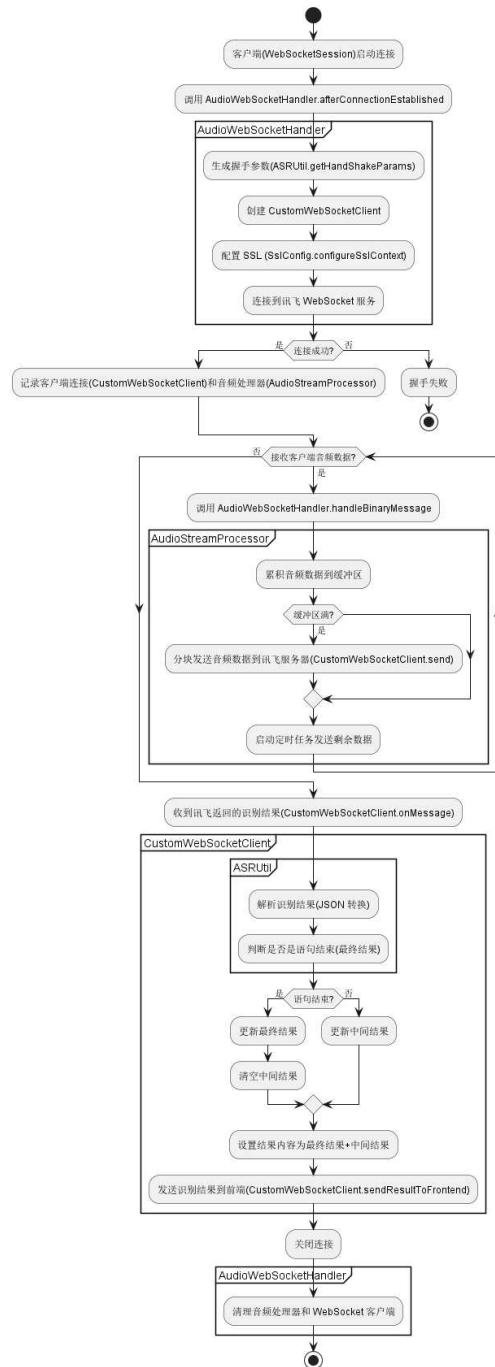


图 5.3: 语音识别构件活动图

### 1. 连接建立阶段

- 客户端发起WebSocket连接
- AudioWebSocketHandler处理连接建立
- 获取握手参数并创建CustomWebSocketClient
- 配置SSL连接

### 2. 音频处理阶段

- 注册音频流处理器(AudioStreamProcessor)
- 接收并处理二进制音频数据
- 累积音频数据进行分块处理
- 通过WebSocket发送处理后的数据

### 3. 识别结果处理阶段

- 接收识别结果
- 解析JSON格式的识别结果
- 判断是否为最终结果
- 更新识别状态
- 发送结果到前端

### 4. 连接关闭阶段

- 清理音频处理资源
- 关闭WebSocket连接

## 5.2.3 关键技术实现

- 实时音频流处理
  - 采用分块处理策略
  - 实现缓冲区管理
  - 保证数据传输实时性
- WebSocket通信
  - SSL安全连接配置
  - 二进制消息处理
  - 心跳维护机制
- 讯飞服务集成
  - 参数动态配置
  - 结果实时解析
  - 错误处理机制

## 5.3 系统核心构件

### 5.3.1 任务管理构件

任务管理构件负责系统中学习任务的创建、分配、追踪和管理。

## 构件组成

- **TaskController**
  - 功能: 处理任务相关的HTTP请求
  - 实现: 提供RESTful API接口, 处理任务的CRUD操作
- **TaskService**
  - 功能: 实现任务管理的核心业务逻辑
  - 实现: 任务创建、更新、查询等业务处理
- **UserTaskStageService**
  - 功能: 管理用户任务阶段
  - 实现: 跟踪任务进度, 状态更新
- 相关存储层
  - **TaskRepository**: 任务数据持久化
  - **UserTaskStageRepository**: 用户任务阶段数据管理
  - **StudentResponseRepository**: 学生答题记录存储

### 5.3.2 用户管理构件

用户管理构件负责系统用户的身份管理和权限控制。

## 构件组成

- **UserController**
  - 功能: 处理用户相关的HTTP请求
  - 实现: 提供用户注册、登录、信息管理等接口
- **UserService**
  - 功能: 实现用户管理的核心业务逻辑
  - 实现: 用户认证、授权、信息管理等
- **UserRepository**
  - 功能: 用户数据的持久化管理
  - 实现: 基于JPA的用户数据访问层

### 5.3.3 安全管理构件

安全管理构件确保系统的安全性和可靠性。

## 构件组成

- **JwtAuthenticationTokenFilter**
  - 功能: JWT令牌认证过滤器
  - 实现: 验证请求中的JWT令牌
- **WebsocketIdentifyInterceptor**

- 功能: WebSocket连接认证拦截器
- 实现: WebSocket握手阶段的身份验证
- **Security**
  - 功能: 安全配置和管理
  - 实现:
    - \* 配置安全策略
    - \* 管理认证流程
    - \* 实现授权控制
- 认证服务
  - **UserService**: 用户认证和授权服务
  - **UserRepository**: 用户数据访问

#### 5.3.4 构件交互

这三个核心构件通过以下方式进行交互:

- **数据流向**
  - Controller层接收请求并委托给Service层
  - Service层实现业务逻辑并通过Repository访问数据
  - 安全构件对所有请求进行拦截和认证
- **安全集成**
  - 所有请求经过安全过滤器验证
  - WebSocket连接通过专门的认证拦截器
  - 统一的用户认证和授权机制
- **数据一致性**
  - 统一的事务管理
  - 数据访问权限控制
  - 并发控制机制

# 第六章 接口设计

## 6.1 内部接口设计

### 6.1.1 用户模块接口设计

接口总览

基础信息

- 基础路径: /api/user/account
- 响应格式: JSON
- 编码格式: UTF-8

接口列表

1. 获取用户信息: GET /api/user/account/info/
2. 用户登录: POST /api/user/account/token/
3. 用户注册: POST /api/user/account/register/

详细接口说明

获取用户信息 基本信息

- 接口URL: /api/user/account/info/
- 请求方式: GET
- 认证要求: 需要JWT认证

请求参数

无需请求参数, 通过JWT Token获取用户信息

响应格式

---

```
1  {
2      "error_msg": "success",
3      "id": "string",
4      "username": "string",
5      "avatar": "string"
6 }
```

---

错误响应

如果Token无效或过期, 将返回401状态码

## 用户登录 基本信息

- 接口URL: /api/user/account/token/
- 请求方式: POST
- 认证要求: 无需认证

### 请求参数

Content-Type: application/x-www-form-urlencoded

参数名	类型	必填	说明
username	string	是	用户名
password	string	是	密码

### 响应格式

```

1  {
2      "error_msg": "success",
3      "token": "string"          // JWT Token
4 }
```

### 错误响应

```

1  {
2      "error_msg": "error"
3 }
```

## 用户注册 基本信息

- 接口URL: /api/user/account/register/
- 请求方式: POST
- 认证要求: 无需认证

### 请求参数

Content-Type: application/x-www-form-urlencoded

参数名	类型	必填	说明	验证规则
username	string	是	用户名	1. 不能为空 2. 长度≤100 3. 不能重复
password	string	是	密码	1. 不能为空 2. 长度≤100
confirmedPassword	string	是	确认密码	必须与password相同
email	string	是	邮箱地址	1. 不能为空 2. 必须符合邮箱格式
avatar	string	否	头像URL	可为空

### 错误信息列表

错误信息	说明
用户名不能为空	username参数为空或全为空格
密码不能为空	password或confirmedPassword为空

用户名长度不能超过100	username超过长度限制
密码长度不能超过100	password超过长度限制
两次输入的密码不同	password与confirmedPassword不匹配
邮箱不能为空	email参数为空或全为空格
邮箱格式不正确	email不符合标准格式
用户名已存在	该username已被注册

## 安全说明

### 认证机制

- 系统使用JWT (JSON Web Token) 进行身份认证
- Token通过登录接口获取
- 需要认证的接口在请求头中需要携带Token:

```
Authorization: Bearer <token>
```

### 密码安全

- 密码使用BCrypt算法进行加密存储
- 密码长度限制在100字符以内

### 数据验证

- 所有输入数据在服务端进行验证
- 特殊字符会被过滤或转义
- 邮箱格式使用正则表达式验证

## 6.1.2 AI模块接口设计

### 接口总览

#### 基础信息

- 基础路径: /api/ai
- 响应格式: JSON
- 编码格式: UTF-8

#### 接口列表

- 获取AI问题反馈: POST /api/ai/questions
- 获取对话历史: GET /api/ai/conversation

## 详细接口说明

### 获取AI问题反馈 基本信息

- 接口URL: /api/ai/questions
- 请求方式: POST
- 认证要求: 需要JWT认证

#### 请求参数

Content-Type: application/json

---

```

1  {
2      "studentId": "integer",      /* 学生ID */
3      "response": "string",       /* 学生回答内容 */
4      "phase": "integer",         /* 当前阶段 (1-3) */
5      "taskId": "integer"        /* 任务ID */
6  }

```

---

#### 响应格式

---

```

1  [
2      {
3          "question": "string",     /* AI提出的问题 */
4          "hints": [                /* 问题相关的提示词列表 */
5              "string"             /* Base64编码的提示图片 */
6          ]
7      }
8  ]

```

---

#### 错误响应

---

```

1  {
2      "message": "错误信息"      /* 具体错误描述 */
3  }

```

---

#### 错误信息列表

错误信息	说明
Response is empty	学生回答内容为空
Task not found	任务ID不存在
Task image not found	任务图片未找到
Invalid phase	无效的阶段值

### 获取对话历史 基本信息

- 接口URL: /api/ai/conversation
- 请求方式: GET
- 认证要求: 需要JWT认证

#### 请求参数

参数名	类型	必填	说明
userId	integer	是	用户ID

taskId	integer	是	任务ID
--------	---------	---	------

### 响应格式

```

1  {
2      "StudentResponses": [
3          {
4              "responseText": "string",      /* 学生回答内容 */
5              "aiFeedback": {                /* AI反馈内容 */
6                  "questions": [
7                      {
8                          "question": "string",
9                          "hints": ["string"]
10                     }
11                 ]
12             }
13         }
14     ],
15     "succeeded": true
16 }
```

### 失败响应

```

1  {
2      "succeeded": false,
3      "message": "错误信息"
4 }
```

## 6.1.3 语音识别模块接口设计

### WebSocket连接接口

#### 基本信息

- **接口URL:** ws://localhost:3001/ws
- 协议: WebSocket
- 认证方式: URL参数传递token

#### 连接参数

参数名	类型	必填	说明
token	string	是	JWT认证令牌

#### 交互格式 前端发送

- 数据格式: Binary
- 数据内容: PCM音频数据流
- 音频要求:
  - 采样率: 16000Hz
  - 位深度: 16bit

- 声道数: 1 (单声道)
- 缓冲区大小: 16384字节

#### 后端响应

---

```

1  {
2      "text": "string",      /* 识别的文本内容 */
3      "isEnd": boolean      /* 是否是句子结束 */
4 }
```

---

### 6.1.4 任务模块接口设计

#### 接口列表

1. 获取所有任务: GET /api/tasks
2. 获取任务详情: GET /api/tasks/{taskId}
3. 创建新任务: POST /api/tasks
4. 获取用户任务阶段: GET /api/tasks/{taskId}/stage
5. 更新用户任务阶段: PUT /api/tasks/{taskId}/stage
6. 获取用户所有任务: GET /api/tasks/user/{userId}
7. 获取用户复习任务: GET /api/tasks/user/{userId}/review

#### 详细接口说明

##### 获取所有任务 基本信息

- 接口URL: /api/tasks
- 请求方式: GET
- 认证要求: 需要JWT认证

#### 响应格式

---

```

1  {
2      "success": boolean,
3      "data": [
4          {
5              "id": "integer",
6              "title": "string",           /* 任务标题 */
7              "description": "string",    /* 任务描述 */
8              "imageDescription": "string", /* 图片描述 */
9              "creationDate": "timestamp" /* 创建时间 */
10         }
11     ],
12     "message": "string"        /* 当success为false时返回错误信息 */
13 }
```

---

## 获取任务详情 基本信息

- **接口URL:** /api/tasks/{taskId}
- **请求方式:** GET
- **认证要求:** 需要JWT认证

### 路径参数

参数名	类型	必填	说明
taskId	integer	是	任务ID

### 响应格式

```

1  {
2      "success": boolean,
3      "task": {
4          "id": "integer",
5          "title": "string",           /* 任务标题 */
6          "description": "string",    /* 任务描述 */
7          "imageDescription": "string", /* 图片描述 */
8          "creationDate": "timestamp" /* 创建时间 */
9      },
10     "message": "string"        /* 当success为false时返回错误信息 */
11 }

```

## 创建新任务 基本信息

- **接口URL:** /api/tasks
- **请求方式:** POST
- **认证要求:** 需要JWT认证

### 请求参数

Content-Type: application/json

```

1  {
2      "title": "string",           /* 任务标题 */
3      "description": "string",    /* 任务描述 */
4      "imageDescription": "string" /* 图片描述 */
5  }

```

### 响应格式

```

1  {
2      "success": boolean,
3      "data": {
4          "id": "integer",
5          "title": "string",           /* 任务标题 */
6          "description": "string",    /* 任务描述 */
7          "imageDescription": "string", /* 图片描述 */
8          "creationDate": "timestamp" /* 创建时间 */
9      },
10     "message": "string"        /* 当success为false时返回错误信息 */
11 }

```

## 获取用户任务阶段 基本信息

- 接口URL: /api/tasks/{taskId}/stage
- 请求方式: GET
- 认证要求: 需要JWT认证

### 请求参数

Query Parameters:

参数名	类型	必填	说明
userId	integer	是	用户ID

### 响应格式

---

```

1  {
2      "stage": "integer"      /* 当前阶段, 0表示未开始 */
3

```

---

## 更新用户任务阶段 基本信息

- 接口URL: /api/tasks/{taskId}/stage
- 请求方式: PUT
- 认证要求: 需要JWT认证

### 请求参数

Query Parameters:

参数名	类型	必填	说明
userId	integer	是	用户ID
stage	integer	是	要更新的阶段

### 响应格式

---

```

1  {
2      "message": "Stage updated successfully"
3

```

---

## 获取用户所有任务 基本信息

- 接口URL: /api/tasks/user/{userId}
- 请求方式: GET
- 认证要求: 需要JWT认证

### 路径参数

参数名	类型	必填	说明
userId	integer	是	用户ID

### 响应格式

---

```

1  [
2  {
3      "id": "integer",
4      "title": "string",          /* 任务标题 */
5      "description": "string",    /* 任务描述 */
6      "imageDescription": "string", /* 图片描述 */
7      "creationDate": "timestamp" /* 创建时间 */
8  }
9 ]

```

---

### 获取用户复习任务 基本信息

- **接口URL:** /api/tasks/user/{userId}/review
- **请求方式:** GET
- **认证要求:** 需要JWT认证

#### 路径参数

参数名	类型	必填	说明
userId	integer	是	用户ID

#### 响应格式

---

```

1  [
2  {
3      "id": "integer",
4      "title": "string",          /* 任务标题 */
5      "description": "string",    /* 任务描述 */
6      "imageDescription": "string", /* 图片描述 */
7      "creationDate": "timestamp" /* 创建时间 */
8  }
9 ]

```

---

## 6.2 外部接口设计

### 6.2.1 WebSocket接口与讯飞服务交互

#### 基本信息

- **接口URL:** wss://rtasr.xfyun.cn/v1/ws
- **认证参数:**
  - APPID: "e071cad7"
  - SECRET\_KEY: "\*\*\*\*\*"
- **协议:** WebSocket with SSL

#### 请求头要求

1	Origin: "https://rtasr.xfyun.cn/v1/ws"
---	--

## 交互格式 发送数据

- 格式: Binary
- 数据块大小: 1280 bytes
- 发送间隔: 40ms

## 接收数据

---

```

1   {
2     "action": "result",
3     "data": {
4       "cn": {
5         "st": {
6           "rt": [
7             {
8               "ws": [
9                 {
10                  "cw": [
11                    {
12                      "w": "识别的文本",
13                      "wp": "n"
14                    }
15                  ]
16                }
17              ]
18            }
19          ]
20        }
21      }
22    }
23  }

```

---

## 6.2.2 智谱AI对话接口

### 基本信息

- 接口URL: <https://open.bigmodel.cn/api/paas/v4/chat/completions>
- 认证参数:
  - API\_KEY: 从智谱AI开放平台获取
  - API\_SECRET: 从智谱AI开放平台获取
- 协议: HTTPS

### 请求头要求

---

```

1   {
2     "Authorization": "Bearer {API_KEY}",
3     "Content-Type": "application/json",
4     "Accept": "application/json"
5   }

```

---

### 请求参数

```

1  {
2      "model": "glm-4v-flash",
3      "stream": boolean,
4      "invoke_method": string,
5      "temperature": float,
6      "messages": [
7          {
8              "role": "system",
9              "content": string
10         },
11         {
12             "role": "user",
13             "content": [
14                 {
15                     "type": "text",
16                     "text": string
17                 },
18                 {
19                     "type": "image_url",
20                     "image_url": [
21                         {
22                             "url": string
23                         }
24                     ]
25                 }
26             ]
27         }

```

### 配置说明

参数	说明	默认值
temperature	响应随机性	稳定模式: 0.05, 随机模式: 0.99
stream	是否启用流式响应	false
model	使用的模型版本	glm-4v-flash

### 响应格式

```

1  {
2      "data": {
3          "choices": [
4              {
5                  "message": {
6                      "role": "assistant",
7                      "content": string
8                  }
9              }
10         ]
11     }
12 }

```

### 响应格式中系统对content要求

```

1  {
2      "questions": [

```

```
3  {
4      "question": "问题内容",
5      "hints": ["提示词1", "提示词2", "提示词3"]
6  }
7 ]
8 }
```

---

## 错误处理

错误类型	处理方式
API调用失败	抛出RuntimeException
响应解析失败	JSON修复或抛出异常
空响应	抛出"Failed to generate AI question"

# 第七章 用户界面设计

## 7.1 注册与登录

注册与登录页面，用户输入信息，如果有错误进行提示，登录成功会自动跳转。

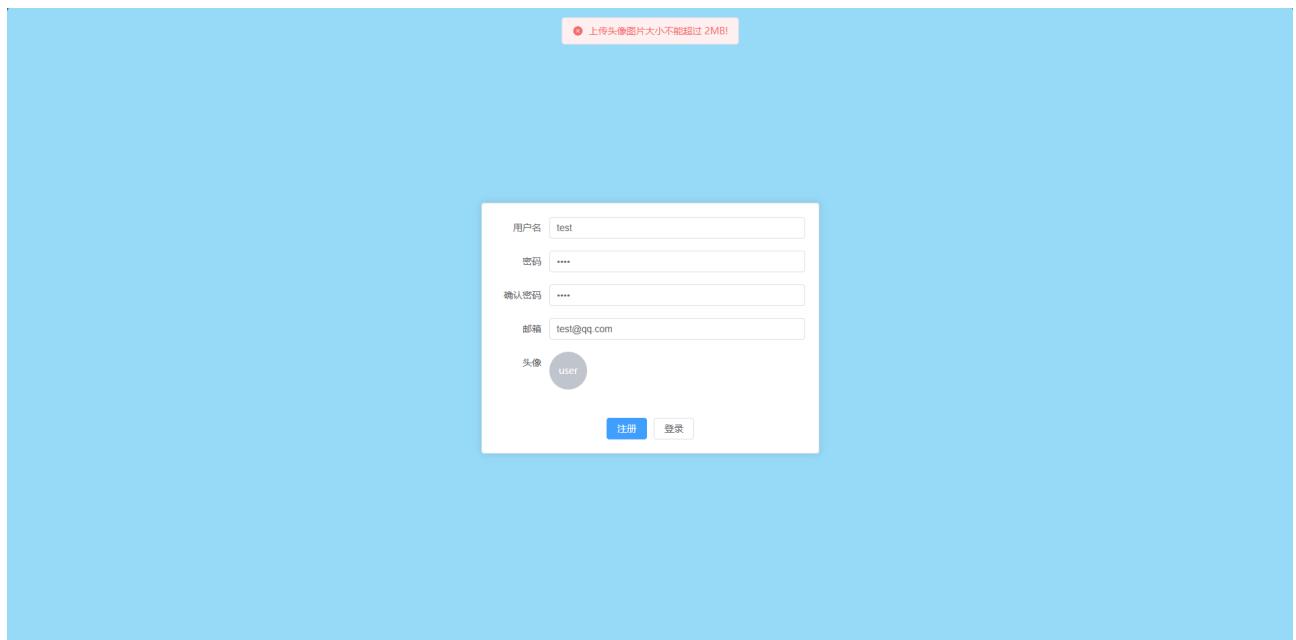


图 7.1: 注册错误-头像太大

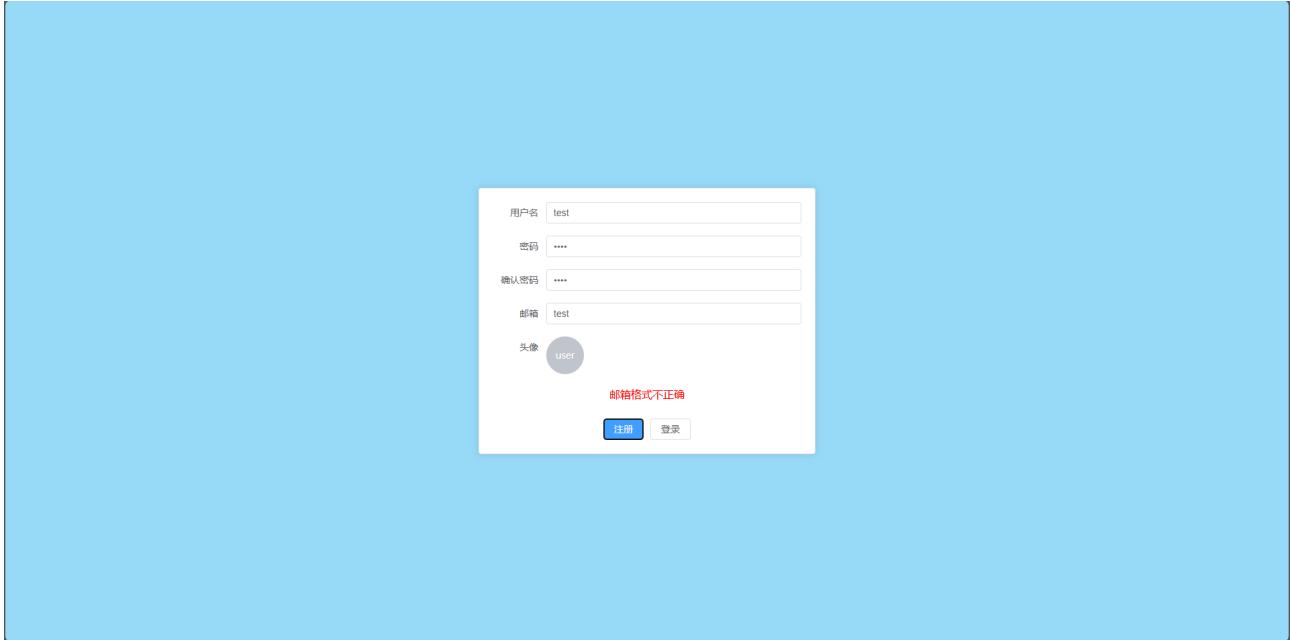


图 7.2: 注册错误-邮箱错误

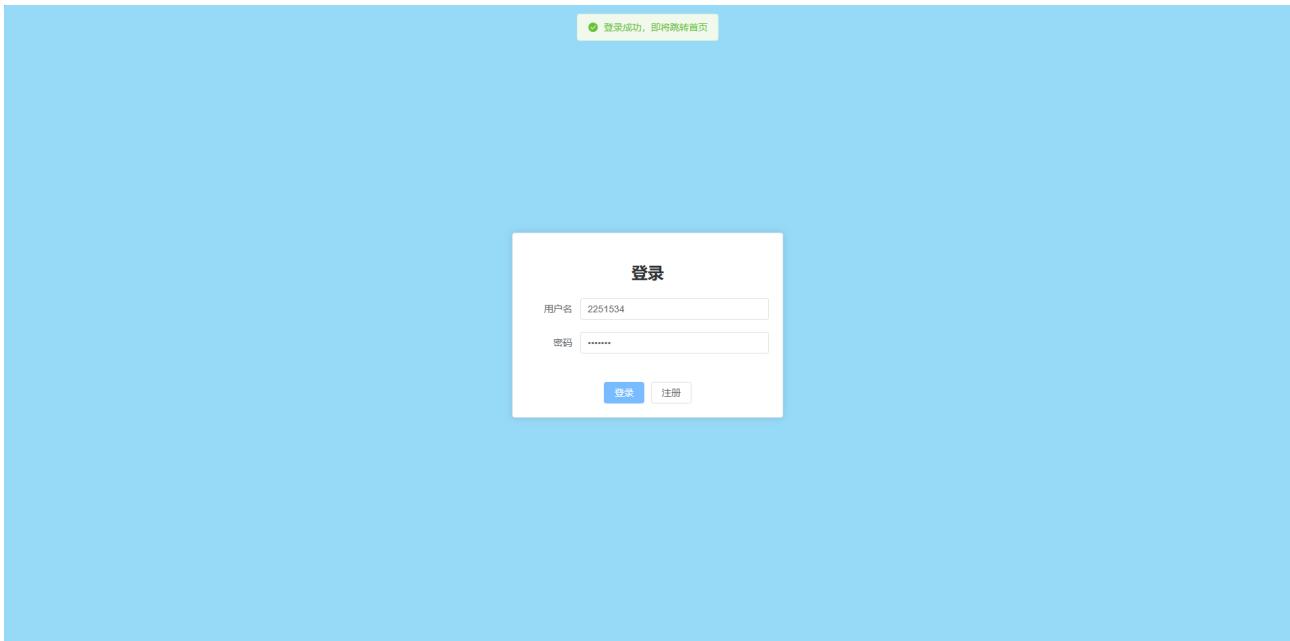


图 7.3: 登录成功

## 7.2 任务中心

进入首页-任务中心，展示全部的任务列表-用户点击任务后可以跳转到具体的任务学习界面，同时可以通过左侧导航栏跳转到个人中心。

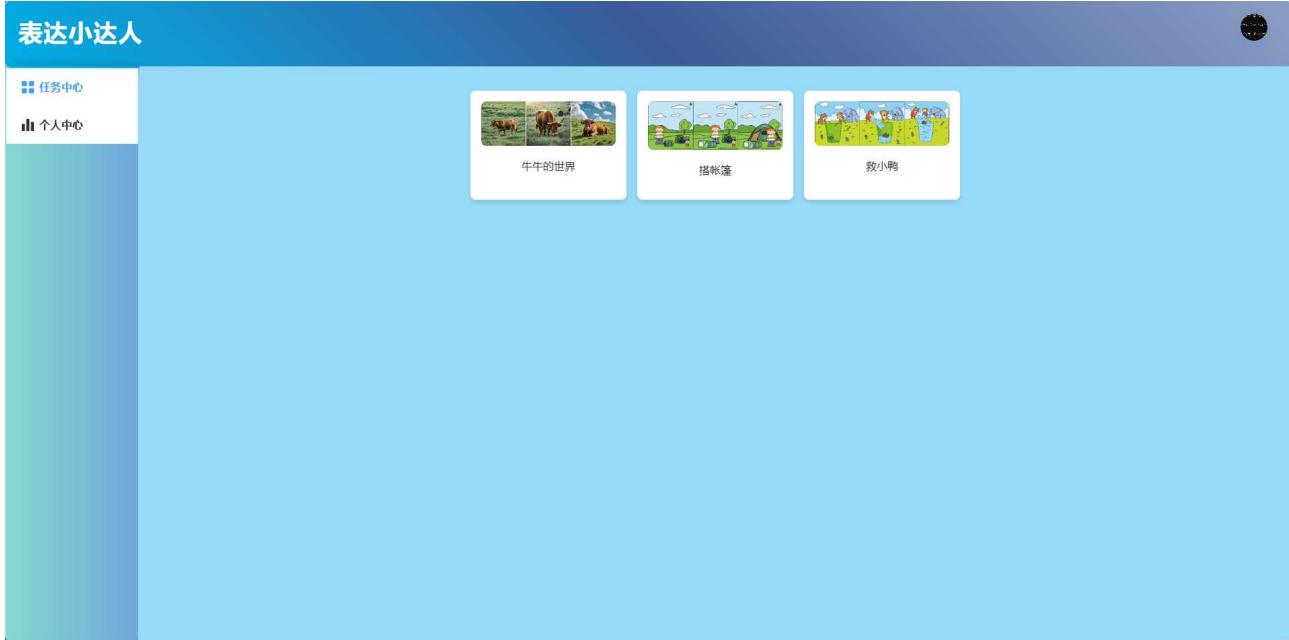


图 7.4: 首页-任务中心

### 7.3 任务学习

可以在任务学习页面通过按钮选择要进行的考验（完整性、逻辑性、情感性），然后点击话筒进行语音输入，再点击话筒停止语音输入，并获取AI的反馈，点击AI反馈对话条，出现AI提供的提示词。

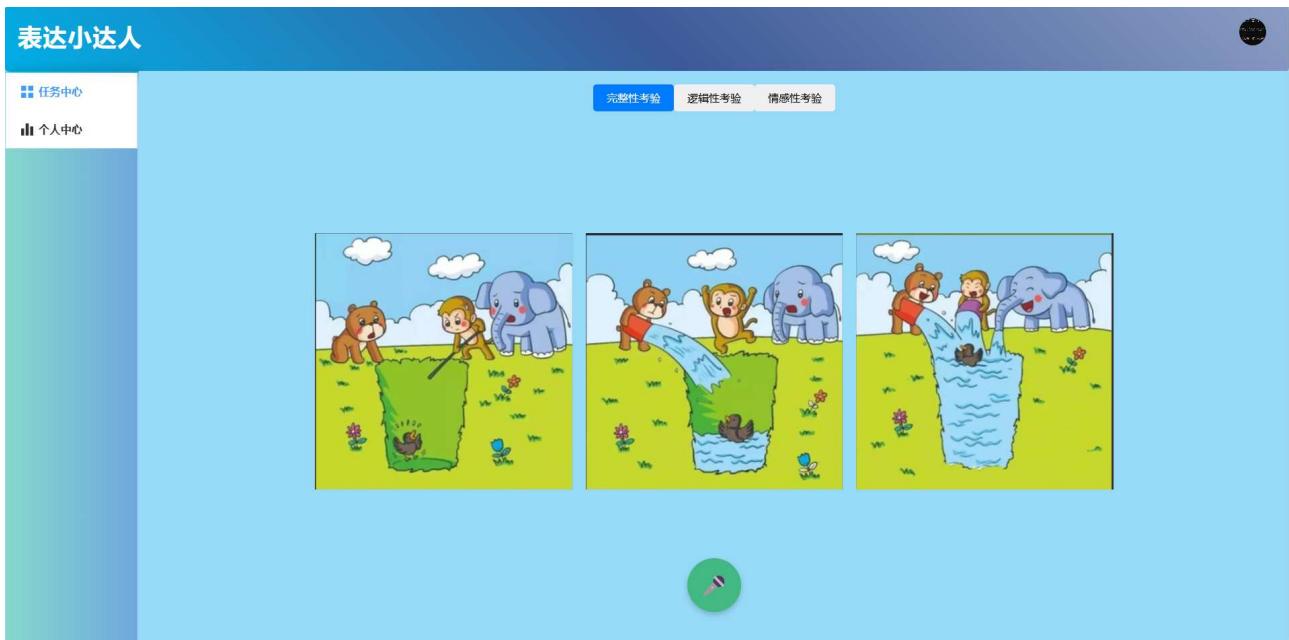


图 7.5: 任务学习界面

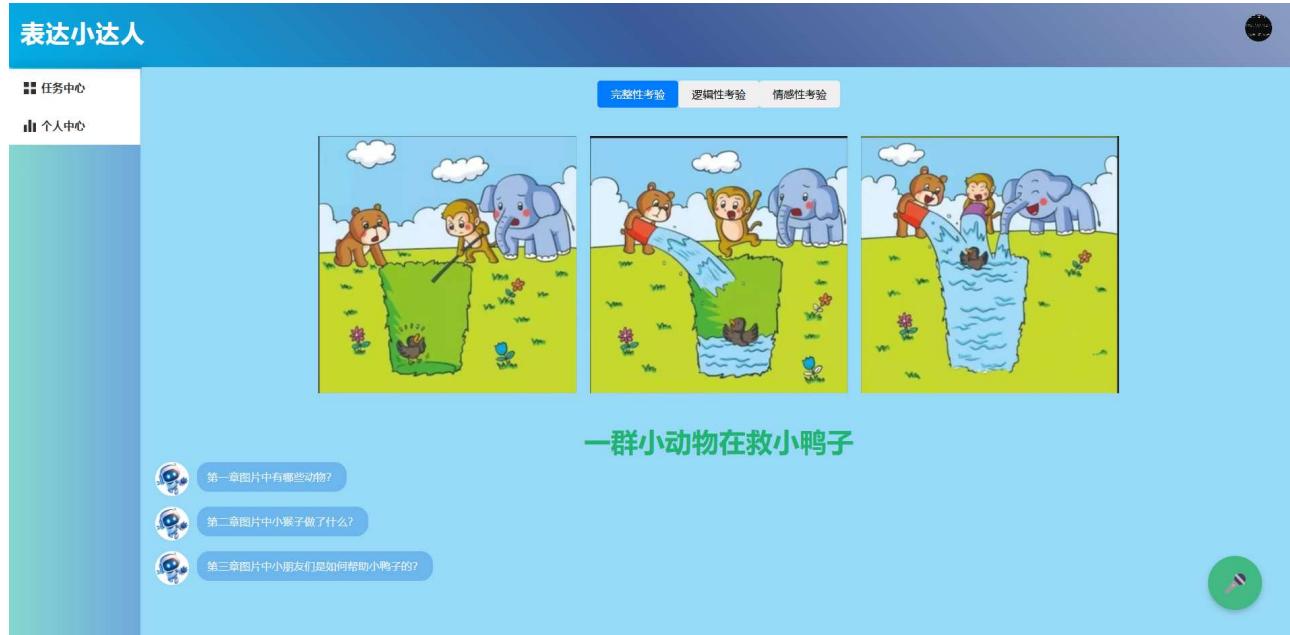


图 7.6: 任务学习-语音识别+AI回复



图 7.7: 任务学习-AI回复展开

## 7.4 个人中心

个人中心展示头像和姓名，首页展示我们的已做任务列表，点击可以跳转到具体任务的历史记录页面，查看过去的历史记录；也可以跳转通过左侧导航到今日待办任务界面或者返回任务中心。

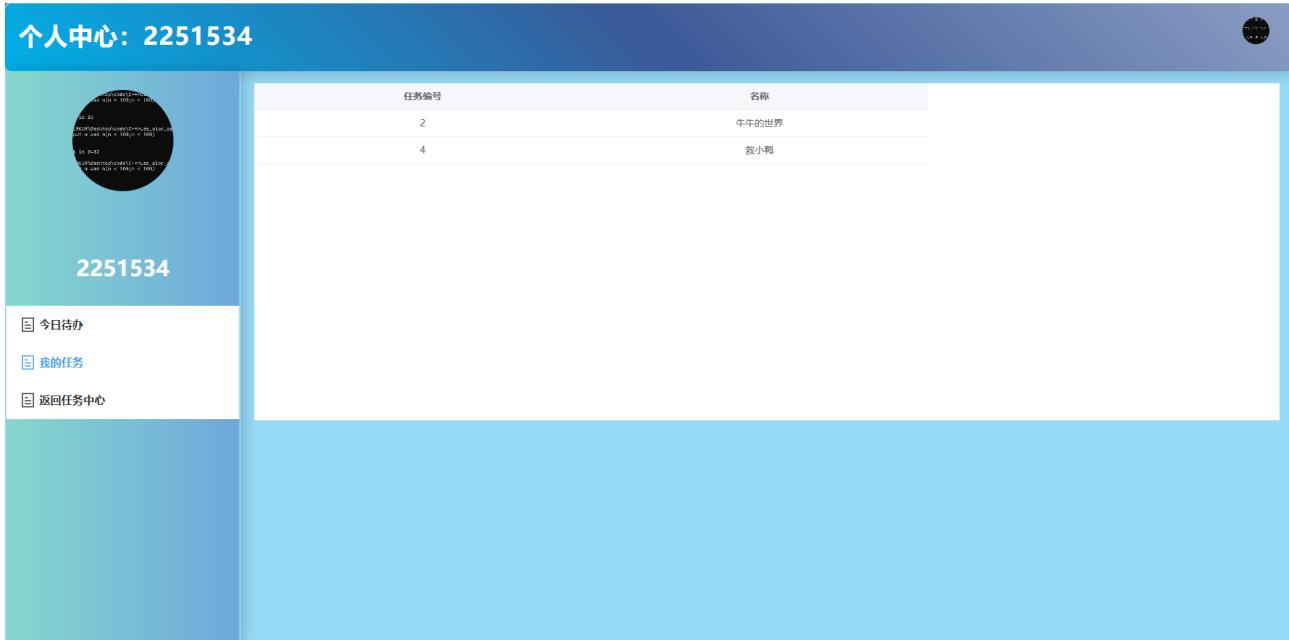


图 7.8: 个人中心界面

## 7.5 历史记录

在历史记录界面可以查看过去对于该任务的学习情况存储所有的自己的学习记录和AI反馈的记录，也可以通过进入学习按键跳转到学习界面。



图 7.9: 历史记录界面-1



图 7.10: 历史记录界面-2

## 7.6 今日待办

在今日待办界面可以看到今日需要复习的任务，点击进入学习页面可以跳转到具体的学习界面。

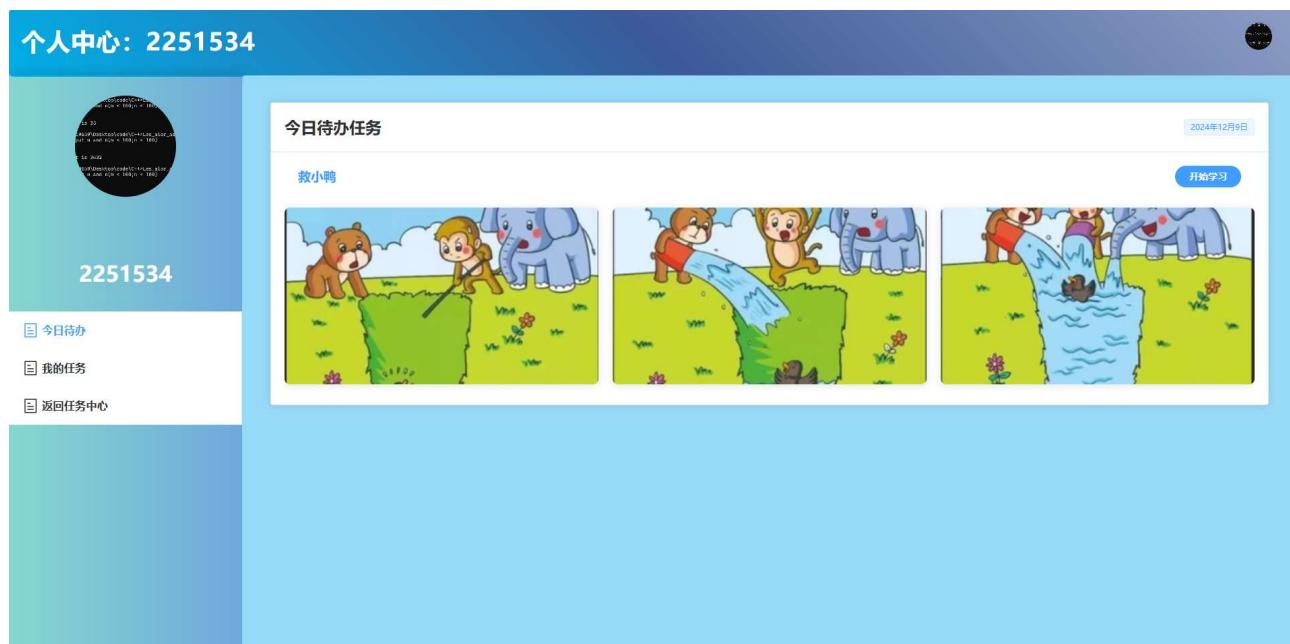


图 7.11: 今日待办界面

# 第八章 系统部署

## 8.1 系统部署概述

本系统采用分布式部署方案，使用三台服务器分别部署后端服务、数据库服务和前端服务。这种部署方式可以提供更好的性能和可扩展性，同时也便于维护和管理。

## 8.2 部署环境说明

### 8.2.1 硬件环境

- 服务器: 8核CPU, 32GB内存, 256GB存储

### 8.2.2 软件环境

- 操作系统: Ubuntu 22.04 LTS

- 后端环境:

- Java OpenJDK 17
  - Maven 3.6+
  - Spring Boot 2.7

- 数据库环境:

- MySQL 8.0

- 前端环境:

- Node.js 18 LTS
  - Nginx 1.18+

## 8.3 部署流程

### 8.3.1 后端服务器部署

#### 安装必要环境

```
1 sudo apt update && sudo apt upgrade -y
2 sudo apt install openjdk-17-jdk maven git curl -y
```

## 部署后端代码

```
1 # clone backend code
2 git clone <>
3 cd <>
4 mvn clean package
```

## 配置系统服务

创建系统服务配置文件:

```
1 [Unit]
2 Description=Spring Boot Backend Service
3 After=network.target
4
5 [Service]
6 User=ubuntu
7 WorkingDirectory=/home/ubuntu/<>
8 ExecStart=/usr/bin/java -jar /home/ubuntu/<>/target/app.jar
9 SuccessExitStatus=143
10 Restart=always
11
12 [Install]
13 WantedBy=multi-user.target
```

## 启动服务

```
1 sudo systemctl daemon-reload
2 sudo systemctl start backend.service
3 sudo systemctl enable backend.service
```

### 8.3.2 数据库服务器部署

#### 安装MySQL

```
1 sudo apt update && sudo apt upgrade -y
2 sudo apt install mysql-server -y
```

#### 初始化数据库

```
1 CREATE DATABASE mydatabase CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
2 CREATE USER 'myuser'@'%' GRANT ALL PRIVILEGES ON mydatabase.* TO 'myuser'@'%';
3 FLUSH PRIVILEGES;
```

#### 配置远程访问

修改MySQL配置文件:

```
1 bind-address = 0.0.0.0
```

### 8.3.3 前端服务器部署

#### 安装Node.js

```
1 curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
2 sudo apt install -y nodejs
```

#### 构建和部署前端代码

```
1 npm install
2 npm run build
3 scp -r dist/* ubuntu<IP>:/var/www/html/
```

#### 配置Nginx

```
1 server {
2     listen 80;
3     server_name <IP>;
4
5     root /var/www/html;
6     index index.html;
7
8     location / {
9         try_files $uri /index.html;
10    }
11 }
```

## 8.4 系统部署图

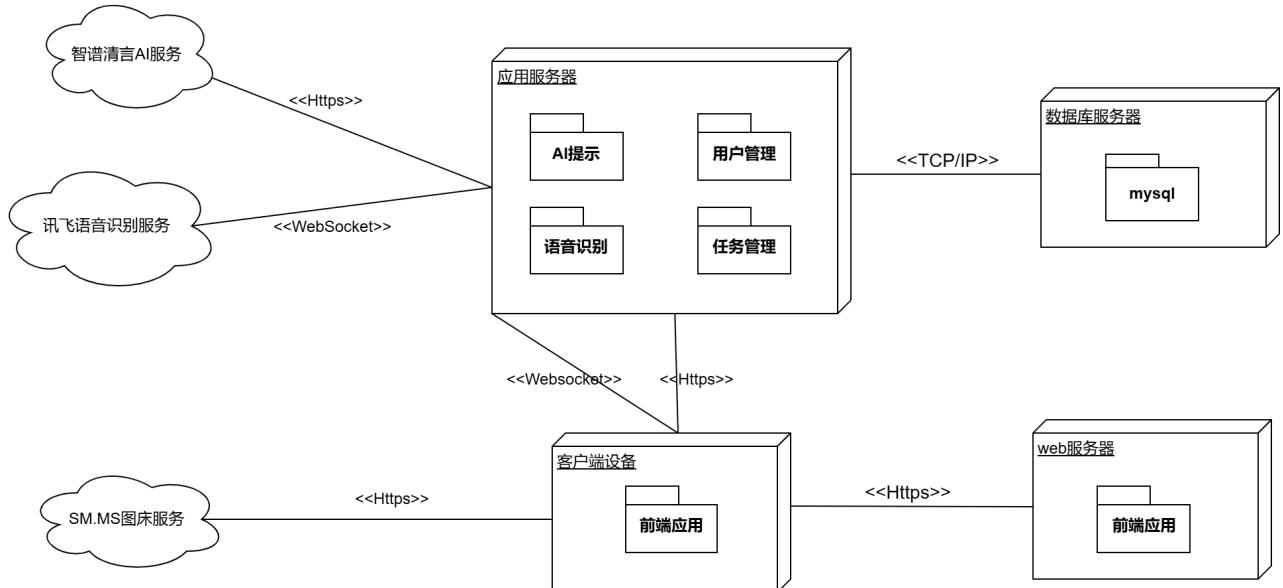


图 8.1: 系统部署图

## 8.5 安全配置

### 8.5.1 防火墙配置

后端服务器

```
1 sudo ufw allow 8080  
2 sudo ufw enable
```

数据库服务器

```
1 sudo ufw allow 3306  
2 sudo ufw enable
```

前端服务器

```
1 sudo ufw allow 'Nginx Full'  
2 sudo ufw enable
```

## 8.6 部署检查清单

后端服务是否正常运行

数据库能否远程连接

前端页面是否可以访问

系统各模块之间是否能正常通信

防火墙规则是否正确配置

日志是否正常记录

系统性能是否满足要求