浙江水利水电学院

课程设计报告

（ 2025-2026 学年 1 学期）

|  |  |
| --- | --- |
| 设计题目 | AI智能·学习搭子 |

|  |  |
| --- | --- |
| 姓 名： | 王泽桉 |
| 学 号： | 2022b11043 |
| 专 业： | 软件工程 |
| 班 级： | 软件工程22-2 |
| 教学单位： | 计算机科学与技术学院 |
| 指导教师： | 杨欐 |

2025年9月10日

**总分:** 分 **五级计分制:**

**评阅教师签名：**

**评语：**

**目 录**

[1.设计介绍 2](#_Toc210591890)

[**1.1 目的和意义** 2](#_Toc210591891)

[**1.2 设计环境与要求** 2](#_Toc210591892)

[2.设计内容 3](#_Toc210591893)

[**2.1 需求分析** 3](#_Toc210591894)

[2.1.1 用户画像 3](#_Toc210591895)

[2.1.2 竞品分析 4](#_Toc210591896)

[2.1.3 功能还原 4](#_Toc210591897)

[2.1.4 系统构建 6](#_Toc210591898)

[2.1.5 技术栈确认 6](#_Toc210591899)

[**2.2 详细设计** 7](#_Toc210591900)

[**2.3 功能实现** 7](#_Toc210591901)

[**2.3.1** 7](#_Toc210591902)

[3.设计总结 7](#_Toc210591903)

[**3.1 设计过程碰到的难点与解决方法** 7](#_Toc210591904)

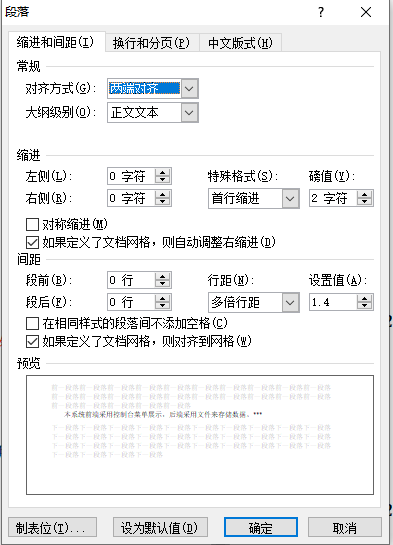
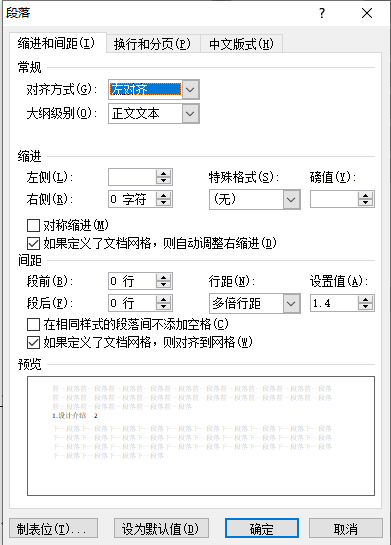
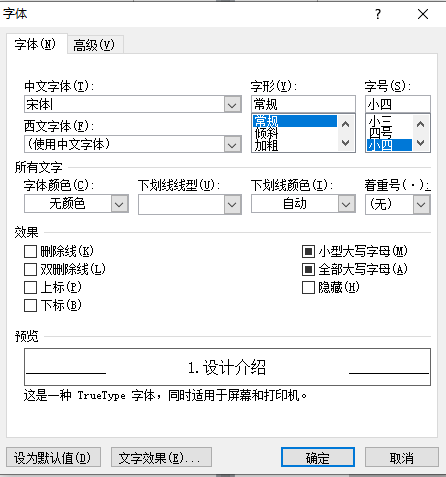
[**3.2 成果与收获** 8](#_Toc210591905)

**[完成后，删除此段红色文字**

**报告撰写结束后，右键更新域——更新整个目录，同时修改目录格式**

**目录格式：字体：小四，宋体，不加粗，不倾斜；1.4倍行距，段前段后为0**

**正文格式（除设定的标题外）：小四，宋体，段落1.4倍行距，段前段后为0，首行缩进两字符**

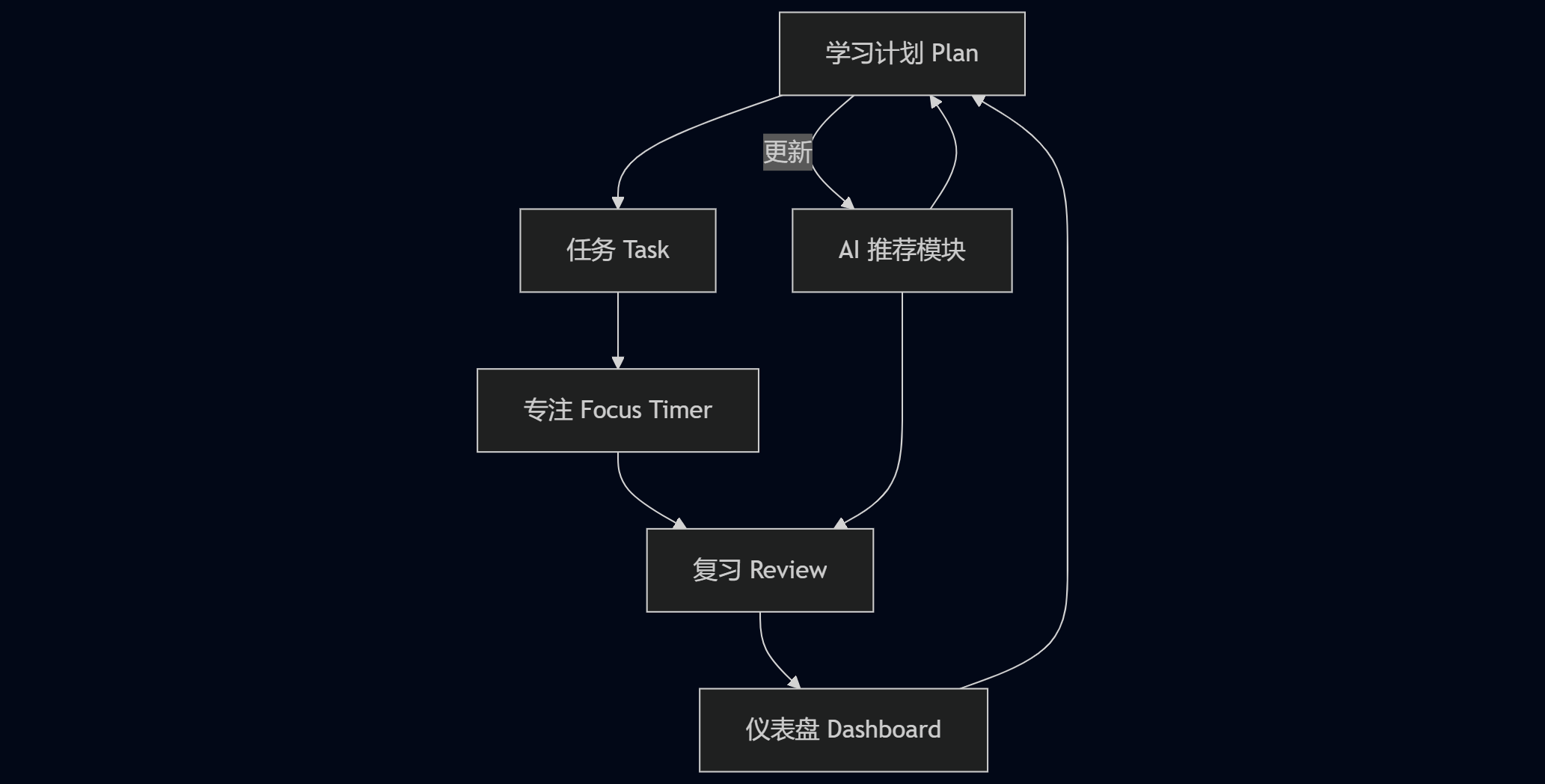
**]**

# **1.设计介绍**

## **1.1 目的和意义**

1.**应对信息过载与知识更新加速**  
在信息爆炸时代，知识更新极快，学习者很难从海量信息中高效筛选并吸收。系统的建设就是为了帮助学习者应对这一挑战，提供更精准的学习资源和路径

2.**突破传统学习模式的局限**  
传统“千人一面”的教学方式，往往无法满足个体的学习节奏和兴趣偏好。而系统的目标是实现差异化、个性化的学习支持，提升学习效率和体验



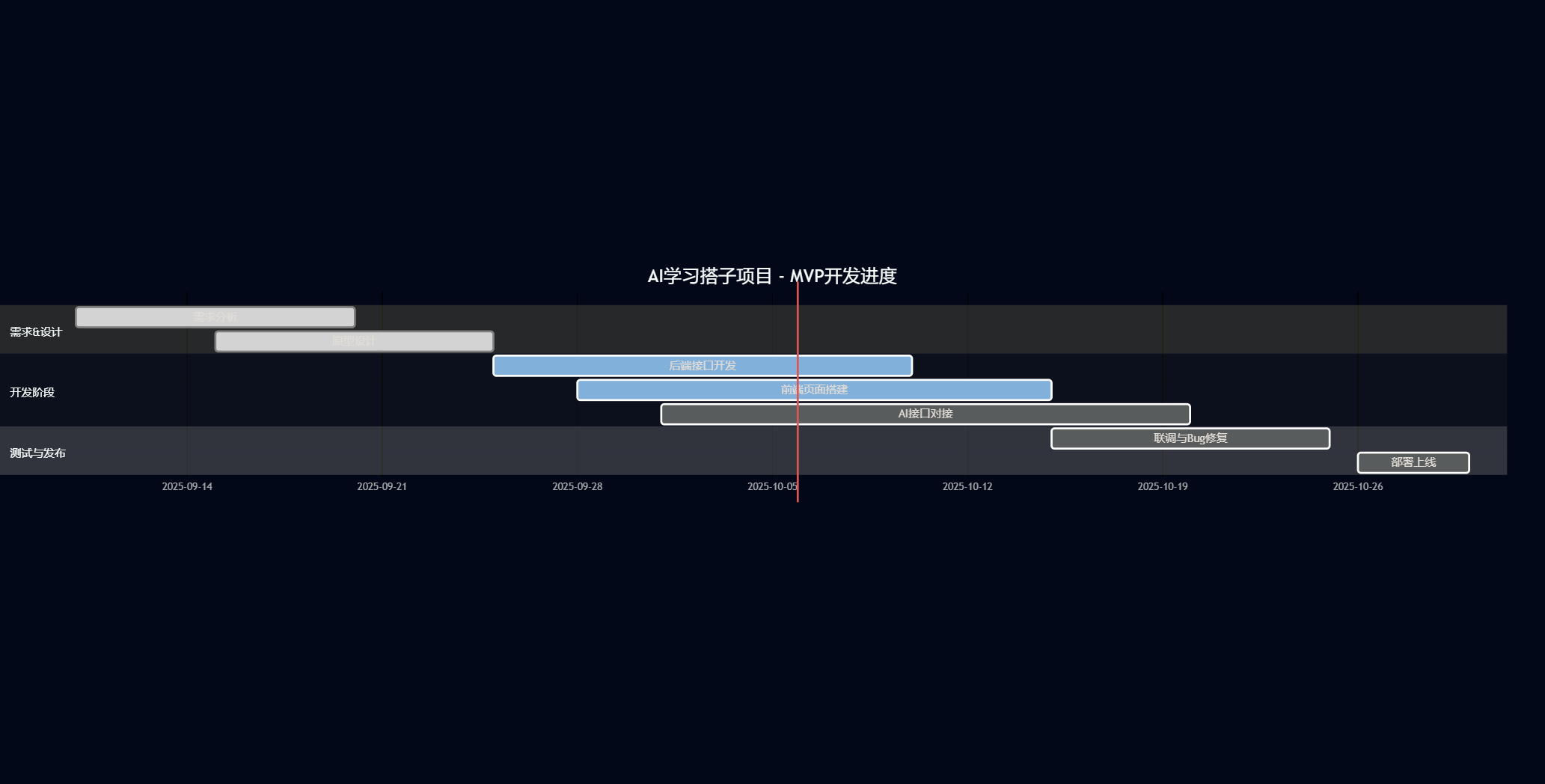
学习计划与专注模块关系

3.**缓解学习过程中的孤独与困境**  
学习者常常面临孤独感、缺乏即时反馈、缺少个性化指导等问题。系统通过 AI 智能搭子功能，能提供陪伴、互动和定制化反馈，减少学习的枯燥感和挫败感

4.**推动教育与人工智能的深度融合**  
通过该系统，可以更好地探索 AI 在教育场景中的应用价值，不仅为学习者赋能，也为教育模式创新提供实践意义

## **1.2 设计环境与要求**

**一、设计环境**



开发迭代甘特图

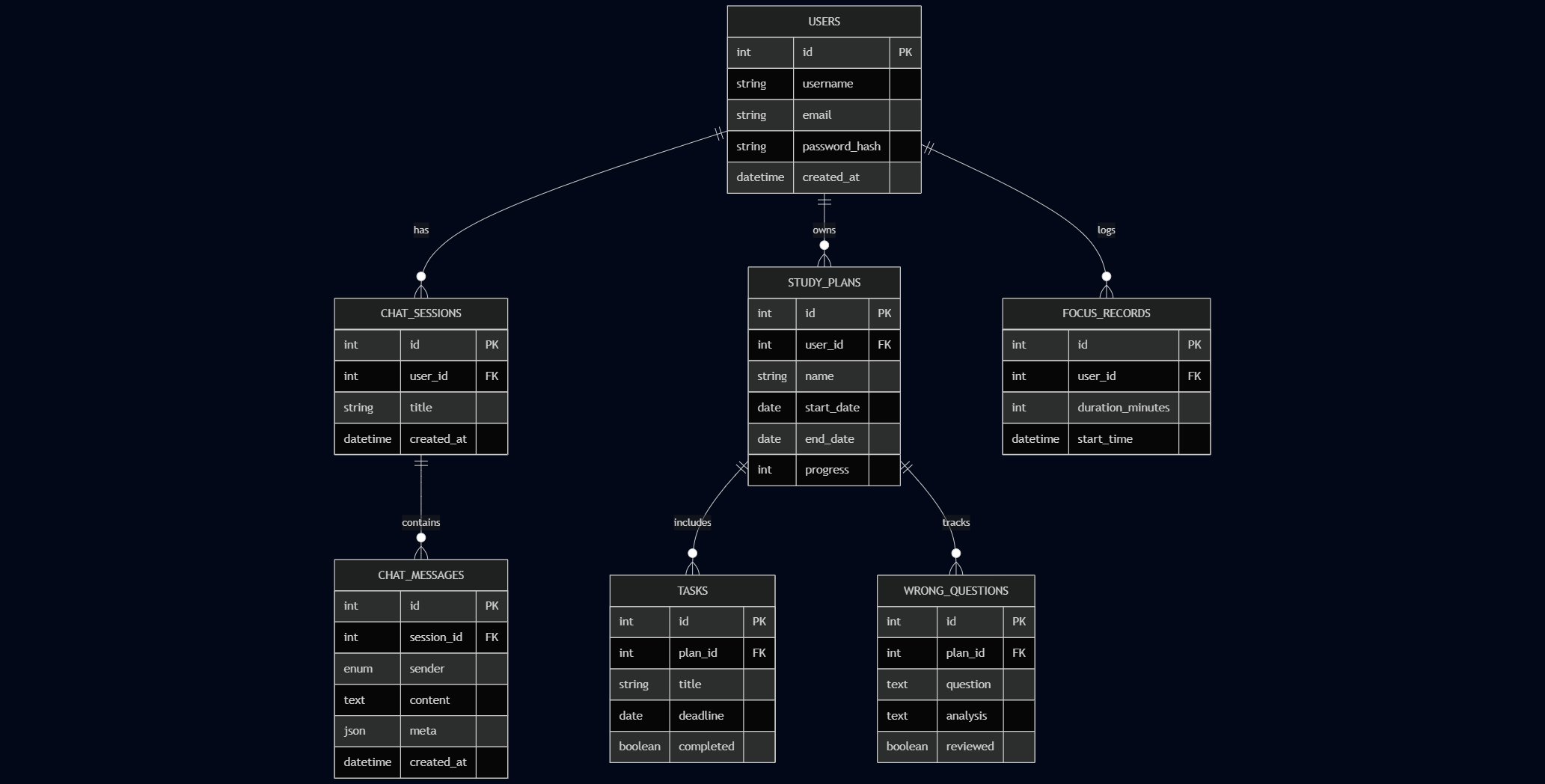
1. **开发框架与语言**

**后端**：Spring Boot（基于 Java），用于实现业务逻辑、用户管理、接口服务

**前端**：Vue.js（渐进式框架），用于构建交互式页面和学习场景的可视化界面

**数据库**：MySQL，存储用户信息、学习记录、任务配置等结构化数据

**知识向量检索**：阿里巴巴云端知识库，提供向量化语义检索功能，支撑个性化推荐与智能答疑



数据库逻辑结构

1. **运行环境**

**服务器环境**：Windows11

**依赖环境**：JDK 17+、Node.js 16+、Nginx（前后端分离部署）

**云平台**：阿里云，利用其 **知识库向量检索服务** 与 **对象存储（OSS）**

1. **AI 支持**

调用Qwen3大语言模型 API，结合向量数据库，实现个性化学习支持

支持知识库的增量更新与语义匹配，保证学习资源实时性

**二、设计要求**

1. **功能性要求**

**个性化推荐**：基于学习者的行为数据和知识库向量检索，提供定制化学习路径与资源

**互动陪伴**：解决学习孤独感，提供 AI 智能搭子功能，实现即时问答与激励反馈

**学习记录管理**：支持学习进度追踪、错题本、知识点复盘

**多端适配**：支持 Web 端为主，未来可扩展至移动端

1. **性能要求**

系统并发能力强，能够同时支持多人学习交互

向量检索延迟需控制在 **200ms** 以内，保证用户体验

系统可扩展，支持知识库规模化扩充

1. **安全性要求**

用户信息加密存储（密码哈希化、通信采用 HTTPS）

数据访问权限控制，确保知识库与用户学习数据的安全

1. **可维护性要求**

前后端分离架构，方便模块化维护与迭代开发

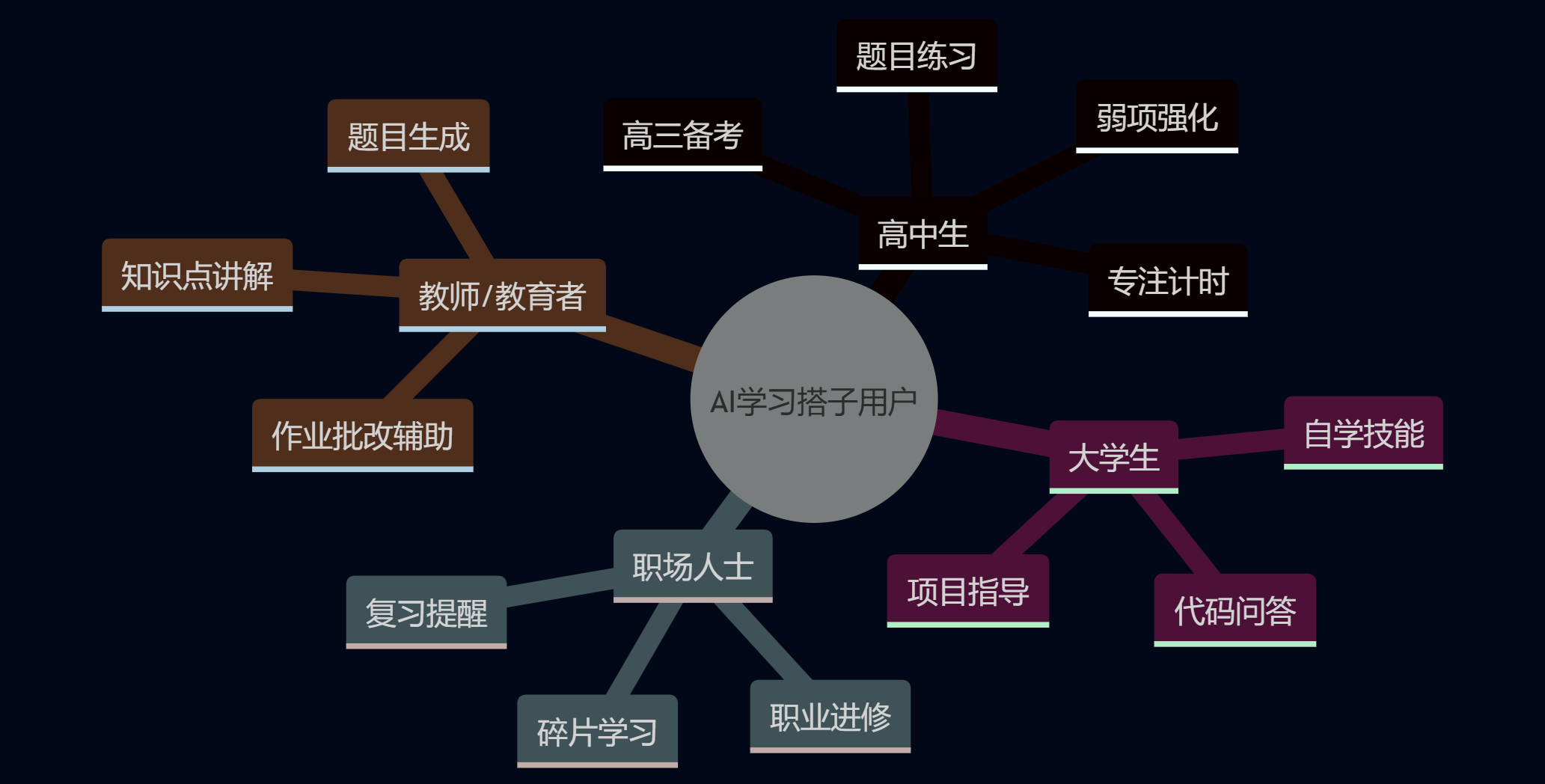
日志监控与错误报警机制，便于运维

代码风格统一，使用 Github 进行版本管理

# **2.设计内容**

## **2.1 需求分析**

### 2.1.1 用户画像



用户画像分类图

1. **李华 — 高三备考生（主要用户）**

年龄：17–18，学习目标：高考冲刺

痛点：题量大、时间紧、需要针对弱项练习、需要随时问答与错题分析

典型场景：在晚自习/回家路上，用手机问题、做定时“专注”练习（Pomodoro 风格），查看错题回顾与学习计划

1. **小王 — 大学自学者（偏技能/编程）**

年龄：20–24，目标：自学编程、完成小项目

痛点：缺少即时代码/概念讲解，容易走弯路

场景：在图书馆/寝室用“聊天”向 AI 咨询概念或代码思路，保存学习笔记

1. **张姐 — 职场进修者**

年龄：28–35，目标：职业证书或技能提升

痛点：时间碎片化、需要高效复习与练习题库

场景：利用“工作”视图安排学习任务与复习提醒，AI 帮忙把长文本浓缩成考点

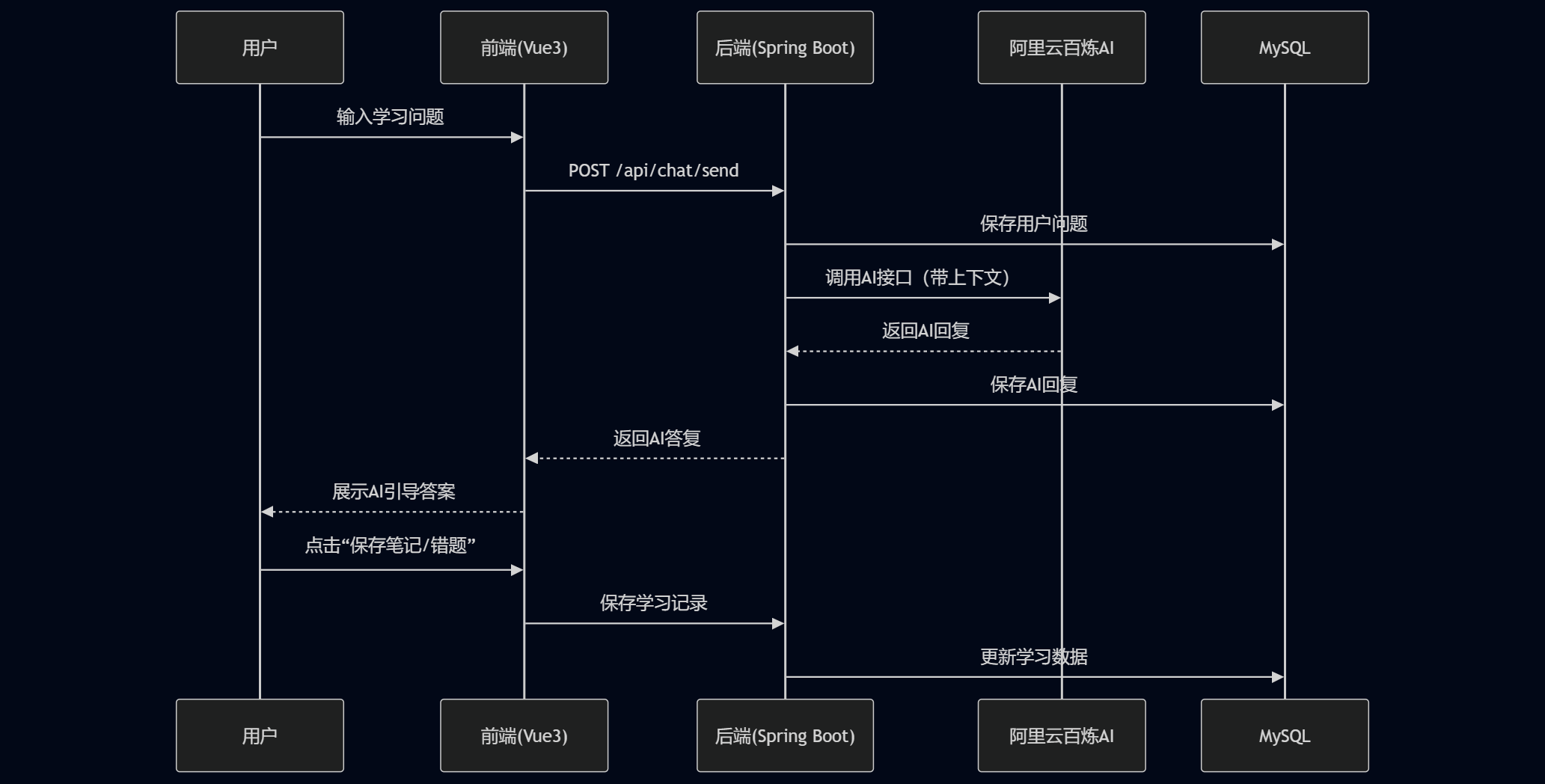
1. **林老师 — 终身学习/教师（次要用户）**

年龄：35+，目标：备课、布置作业与判题效率

场景：用 AI 生成练习题、参考答案、讲义摘要（教师端功能可作为中长期拓展）

这些画像将直接影响功能优先级（例如：高三生优先要题目练习、错题本、专注模式；大学生更需要开放问答与笔记导出）

### 2.1.2 竞品分析



核心业务流程图

**作业帮（国内）**

亮点：已经把 AI 用在学习机/作业批改/视频答疑等场景，2024年报告过大量自动批改与高频视频答疑数据，侧重“中小学生作业辅导与题库”

**猿辅导 / 小猿（国内）**

亮点：基于自有数据构建知识图谱与大模型（大量教学视频/题库支持），推出小猿学练机与 AI 驱动功能，强调练习路径个性化与多模态交互

**Khanmigo（Khan Academy，国际）**

亮点：作为非营利教育组织的 AI 辅助教学（Khanmigo），采取引导式教学（不直接给出答案而是引导学生思考），重视学习伦理与教育效果。适合以“引导+练习”为核心的产品策略参考

**Quizlet（国际）**

亮点：强大的记忆练习（闪卡）生态，正在把 AI 用于内容生成与个性化课程（Course-Powered）以增强协作学习

**ChatGPT / 大型通用模型（背景）**

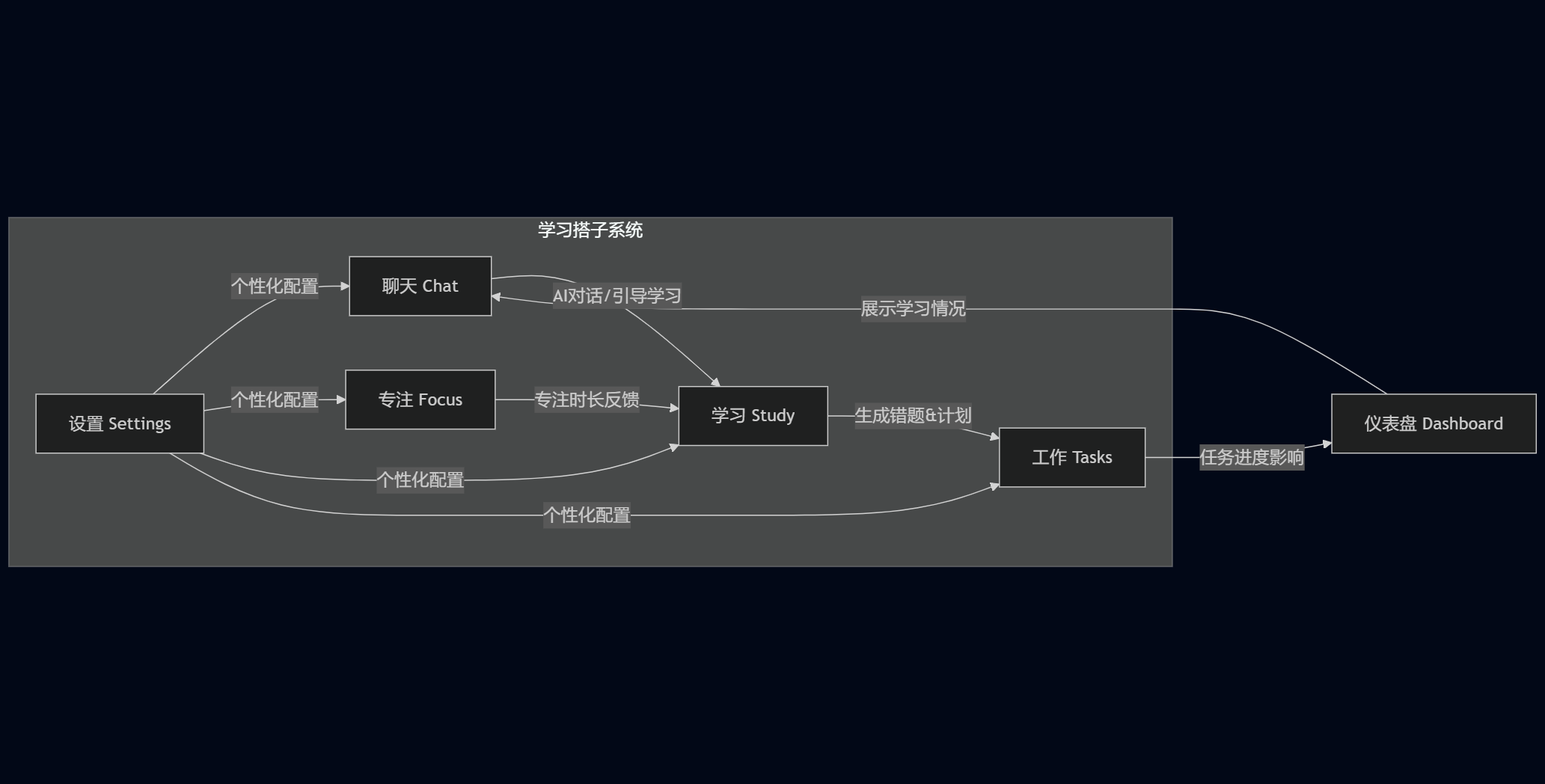
趋势：多家通用 AI 服务正加入“学习/教学模式”（如 ChatGPT 的 Study Mode），体现行业方向：把大模型从“答案引擎”转为“教学引导器”。这说明产品设计上需要把“引导式学习（Socratic）”与“直接答疑”做权衡

**竞品给我们的启示（机会）**

差异化策略：聚焦“陪伴式学习 + 轻量化社交（学习搭子） + 目标导向的学习计划”，而非单纯题库或单一答疑工具

技术上：竞品倾向把 AI 用在题目生成、个性化练习安排与多模态（语音/图像）交互。我们可以从“对话 + 学习计划 + 专注模式 + 进度反馈”切入，形成差异化（不需要一次性具备超大题库，先以智能陪伴与适配练习为 MVP）

### 2.1.3 功能还原



功能模块关系图

**A. 聊天（Chat） — 核心 MVP 功能之一**

**目标**：实现多轮上下文感知的学习对话，能回答问题、引导学习并生成练习题或讲解要点

**用户故事**

作为学习者，我希望能在聊天中输入问题并得到分步引导的解题思路（而不是直接给出答案），以加深理解

作为学习者，我希望能保存有价值的对话为笔记/题目，方便复习

**功能细分**

多轮对话上下文追踪（会话 ID 管理）

分步引导模式（可切换“直答/引导”）

会话历史查看、标星与保存为“知识卡”

后端调用阿里云百炼/AI 接口生成回复并做缓存（配置化 prompt 与温度）

**验收标准**

用户发送问题 → 在 3s 内返回 AI 初次响应（具体延时依赖外部 AI 接口）

会话历史可以分页/检索，用户可标记并导出为文本/JSON

**示例 API（REST）**

POST /api/chat/send

body: { userId, sessionId?, message, mode: "guide"|"direct" }

返回: { replyId, text, aiMeta:{model, promptUsed, tokens?} }

GET /api/chat/history?userId=&sessionId=&page=

POST /api/chat/save-note（把会话片段保存为笔记）

**数据库（MySQL）表设计示例（简要）**

users（id, username, hashed\_password, email, created\_at）

chat\_sessions（id, user\_id, title, created\_at, last\_active）

chat\_messages（id, session\_id, sender:ENUM('user','ai'), content TEXT, meta JSON, created\_at）

**B. 专注（Focus） — 学习习惯培养**

**目标**：提供定时/专注机制（例如 Pomodoro），并在专注结束后自动记录学习时长与打卡

**功能**

开始/暂停/结束专注任务（持续时长、任务标签）

专注统计（日/周/月）并与学习计划关联

专注时禁止/屏蔽通知（前端控制）

**验收**

用户能创建一个专注任务并在完成后查看累计时长与任务清单

**C. 学习（Study） — 内容与练习管理**

**目标**：管理学习计划、题库、错题本与进度追踪

**功能**

创建/编辑学习计划（目标、起止日期、每天计划时长）

系统根据计划每天推荐练习（基于错题、知识点）——此处 AI 负责“推荐题目/习题难度调整”逻辑（调用阿里云 AI 做能力评估与题目生成或筛选）

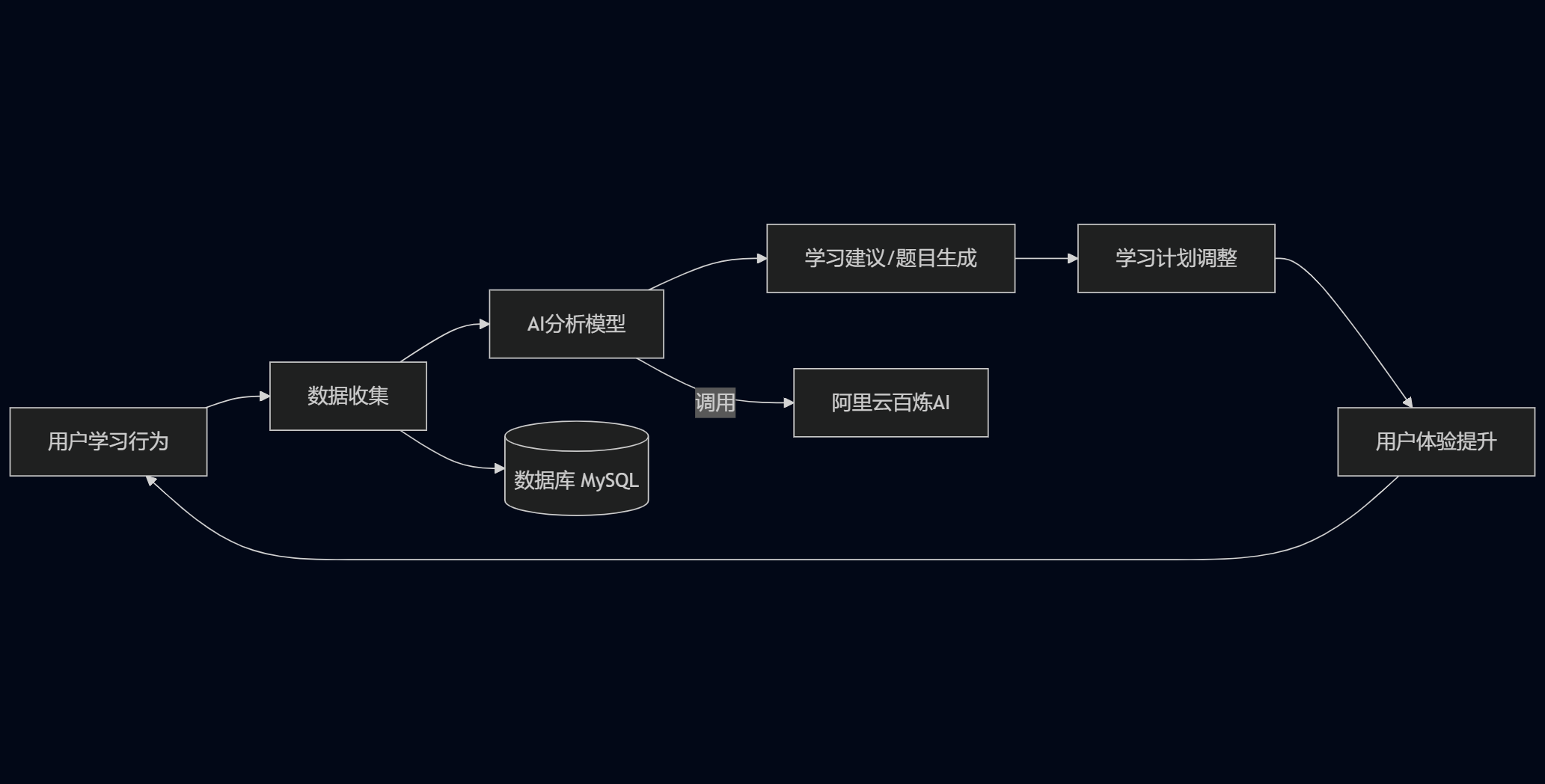
AI学习搭子项目

错题本（记录错题、AI 给出逐题解析、复习提醒）

学习数据仪表盘（完成率、正确率、弱点分布）

**验收**

学习计划可被创建并生成当天推荐练习列表；错题复习能够标记“已掌握”



学习行为与数据闭环图

**D. 工作（Tasks / Productivity）**

**目标**：把“工作”视为学习内的任务管理（任务、提醒、待办）

**功能**

任务创建（deadline、优先级、关联学习计划）

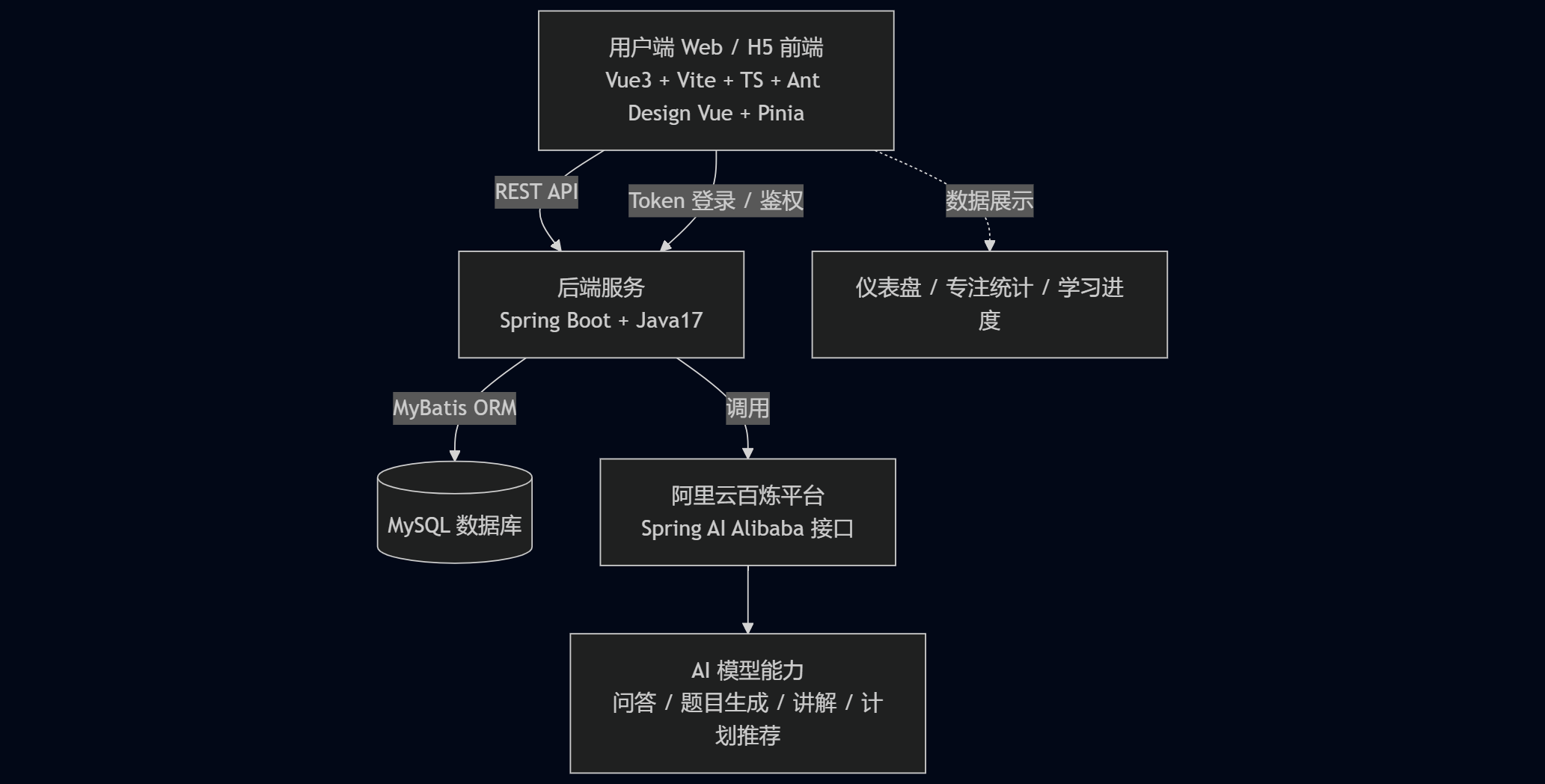
日历/日程视图（可在未来迭代）

任务完成触发自动奖励/日统计

**E. 设置（Settings）**

账户管理、隐私与数据导出、AI 模式切换（直接答/引导）、通知偏好、主题/外观等

### 2.1.4 系统构建



项目总体结构（系统架构图）

**总体架构**

**客户端（SPA, Vue 3 / TypeScript）**：负责 UI、状态管理（Pinia）、路由（Vue Router），与后端通过 RESTful API 通信（项目现状已如此组织）

**后端（Spring Boot）**：提供 REST 接口、业务逻辑、持久化（MyBatis 对接 MySQL），并作为 AI 中间层负责与阿里云百炼平台（通过 Spring AI Alibaba 或自定义 SDK）交互以获取模型响应与题目生成

**数据库（MySQL）**：存储用户、会话、学习计划、题目记录、错题与统计数据（见上面数据表示例）

**AI 服务**：通过后端调用阿里云的 AI 接口（API Key 等配置保存在后端配置文件/环境变量中，走 HTTPS），后端负责 prompt 管理、上下文拼接与对 AI 响应的后处理（例如抽取题目、生成解析）

**关键交互**

1. 用户前端发送问题 POST /api/chat/send → 后端接收并写入 chat\_messages（user）
2. 后端构建 prompt（包含必要上下文）并调用阿里云 AI 接口（同步或异步）
3. AI 返回响应 → 后端将 AI 内容写入 chat\_messages（ai），并返回给前端展示
4. 若用户“保存为错题”，后端在 problem\_records 表插入条目并更新学习计划统计

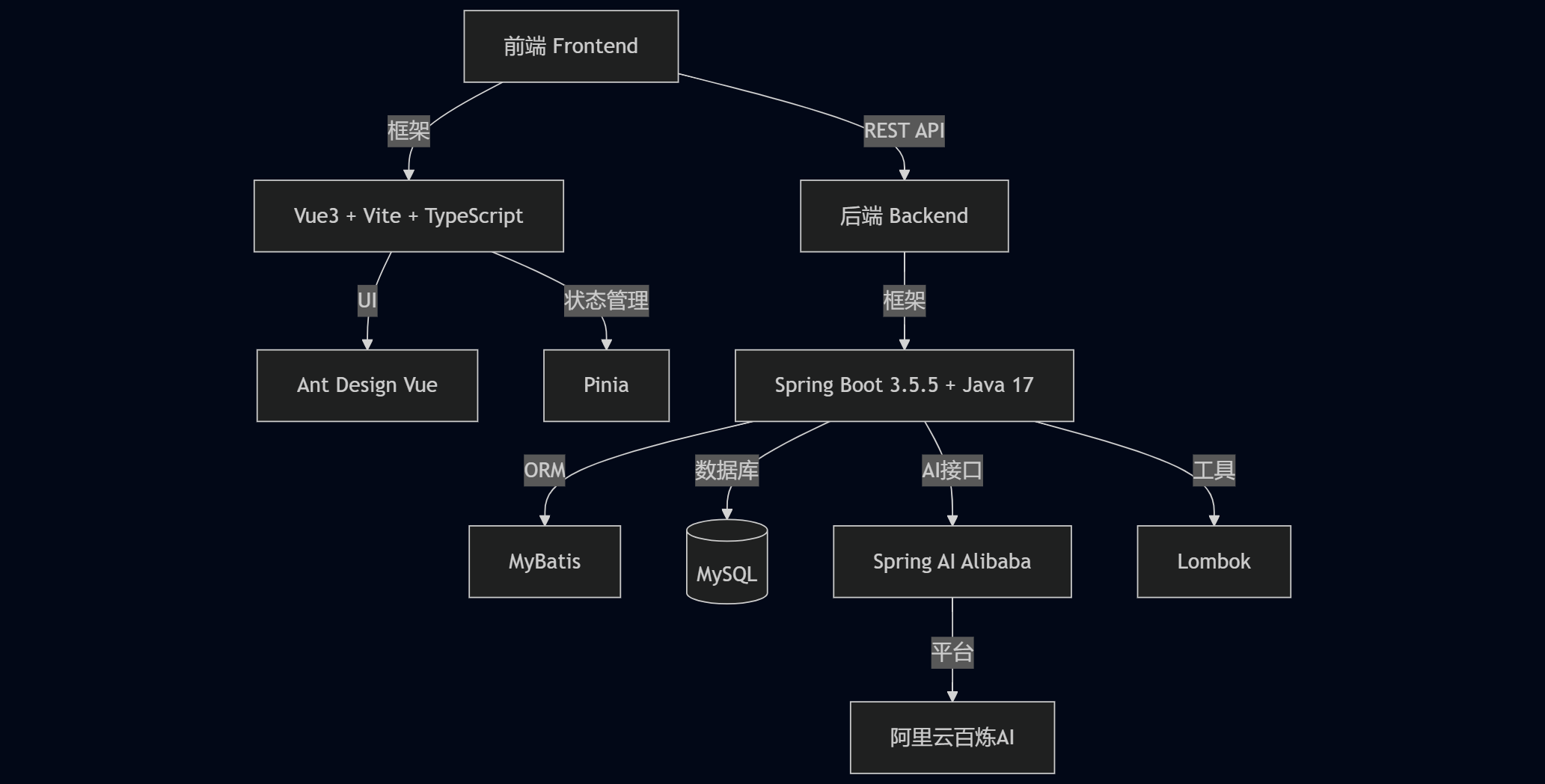
**配置与安全**

AI 密钥与配置放在后端（application.yml）通过环境变量注入，敏感信息不进仓库

认证采用 token-based 机制（前端持有 token，后端校验）。（项目 router 中已包含认证路由配置，提示已有认证需求）

数据隐私：对话日志/错题属于用户敏感学习数据，导出/清除应提供用户操作权限并有审计记录

### 2.1.5 技术栈确认



技术栈确认图

**前端**

Vue 3（Composition API）— SPA 框架

TypeScript — 静态类型

Vue Router — 路由

Pinia — 状态管理

Ant Design Vue — UI 组件库

**后端**Java 17（运行时）

Spring Boot 3.5.5（核心后端框架）

Spring Web (MVC) — REST API

MyBatis Spring Boot Starter — ORM/Mapper 层

MySQL Connector/J — 与 MySQL 连接

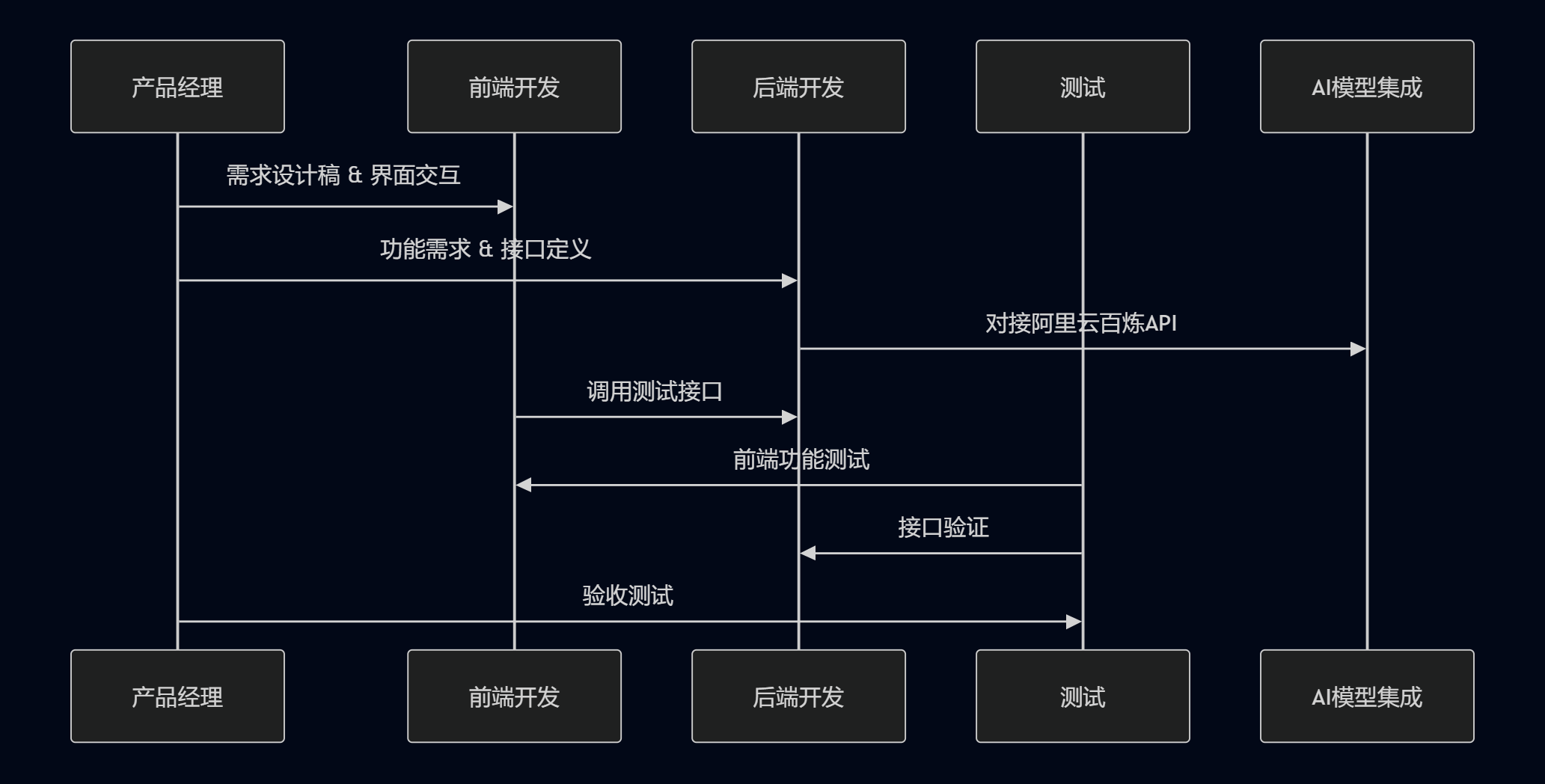
Spring AI Alibaba / 阿里云百炼平台集成（用于 AI 能力调用）

Lombok — 简化实体/DTO/VO 等 Java 样板代码

**开发 / 构建说明**

前端：npm install / npm run dev / npm run build

后端：Maven 构建：mvn clean install / mvn spring-boot:run



项目开发泳道图

## **2.2 详细设计**

### **2.2.1登录模块**

**功能概述**

登录模块负责验证用户身份，允许已注册用户访问系统资源。该模块包括用户身份验证、会话管理和安全保护等功能。

**系统架构/技术栈**

后端框架：Spring Boot 3.5.5

数据库：MySQL

ORM框架：MyBatis

安全框架：Spring Security (建议)

密码加密：BCrypt

**相关组件**

Controller层：UserController

Service层：UserService (待实现)

Mapper层：UserMapper

Entity层：User实体

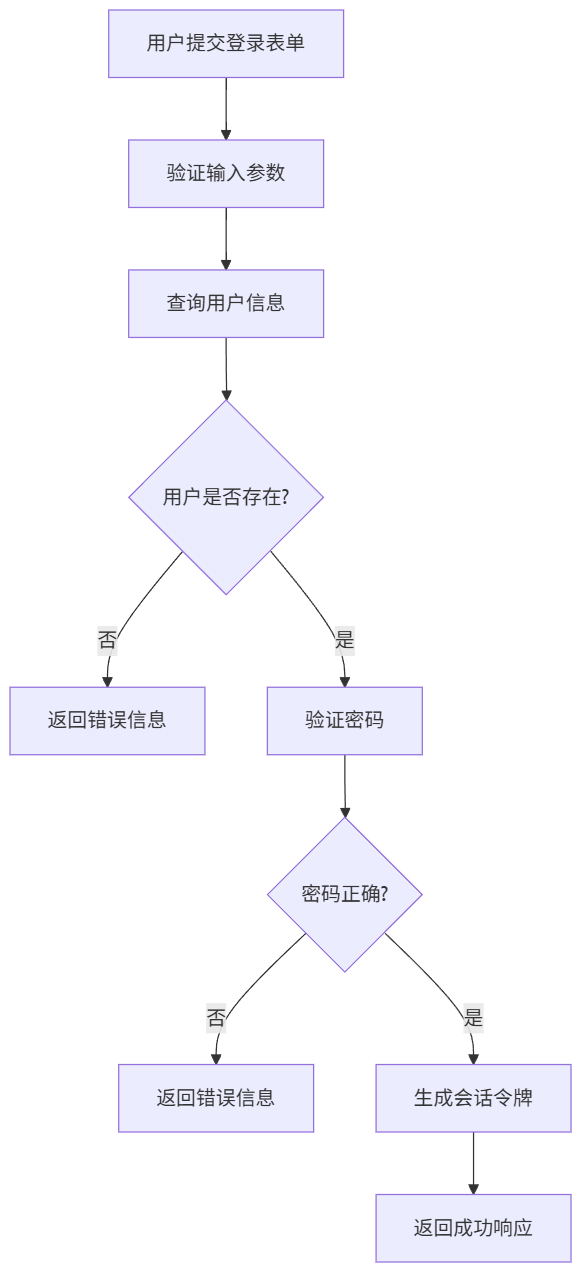
DTO层：LoginRequest, LoginResponse

**数据库设计**

用户表(t\_user)

根据现有SQL文件分析：

|  |
| --- |
| CREATE TABLE IF NOT EXISTS t\_user(  id INT AUTO\_INCREMENT PRIMARY KEY COMMENT '用户id',  email VARCHAR(255),  password VARCHAR(255)  ) |



### **2.2.2注册模块**

**功能概述**

登录模块负责验证用户身份，允许已注册用户访问系统资源。该模块包括用户身份验证、会话管理和安全保护等功能。

**技术栈**

后端框架：Spring Boot 3.5.5

数据库：MySQL

ORM框架：MyBatis

安全框架：Spring Security (建议)

密码加密：BCrypt

**相关组件**

Controller层：UserController

Service层：UserService (待实现)

Mapper层：UserMapper

Entity层：User实体

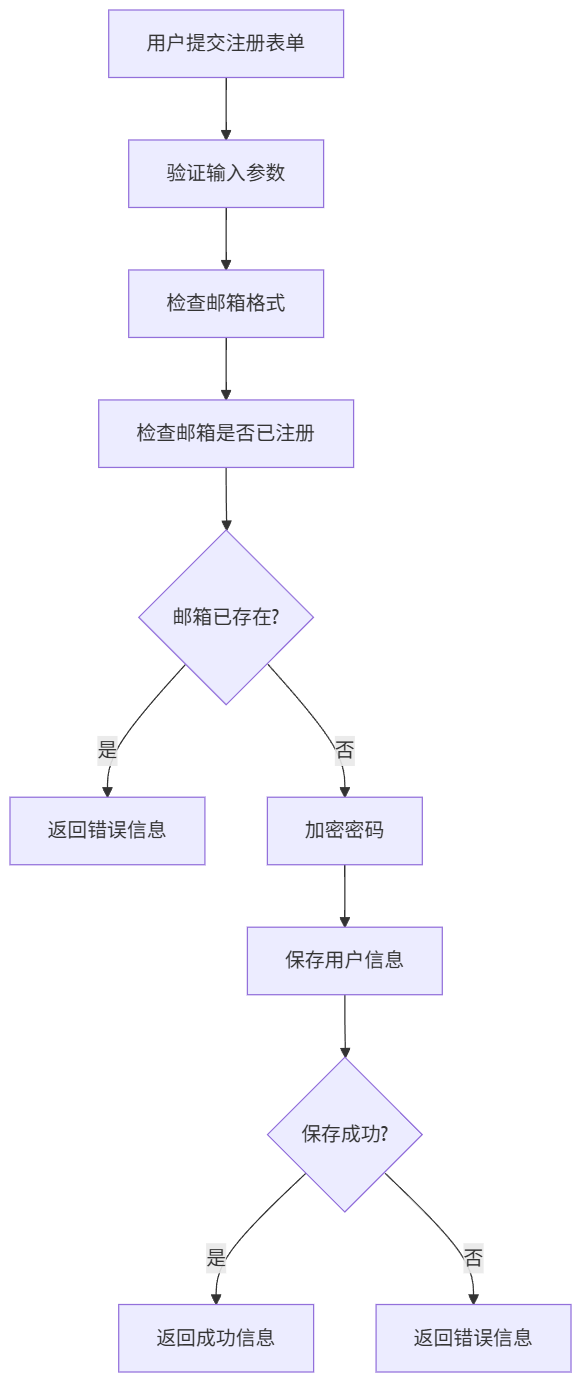
DTO层：LoginRequest, LoginResponse

**数据库设计**

用户表(t\_user)

根据现有SQL文件分析：

|  |
| --- |
| CREATE TABLE IF NOT EXISTS t\_user(  id INT AUTO\_INCREMENT PRIMARY KEY COMMENT '用户id',  email VARCHAR(255),  password VARCHAR(255)  ) |



### 2.2.3学习搭子AI对话模块

**功能概述**

学习搭子AI对话模块是本系统的核心功能之一，旨在为用户提供智能化的学习伙伴服务。该模块利用阿里云百炼平台的AI能力，结合用户的学习需求，提供个性化的学习指导、答疑解惑、学习计划制定等服务。

**技术栈**

后端框架：Spring Boot 3.5.5

AI平台：阿里云百炼(DashScope)

数据库：MySQL

ORM框架：MyBatis

对话记忆：Spring AI Alibaba Memory

**相关组件**

Controller层：ChatAIController, ChatMemoryController

Service层：AliyunBailianApplicationService, AliyunBailianDashScopeService

Mapper层：ChatMessageMapper, ConversationMapper

Entity层：ChatMessage, Conversation

POJO层：AliyunBailianApplication

配置层：AliyunConfig, ChatMemoryConfig

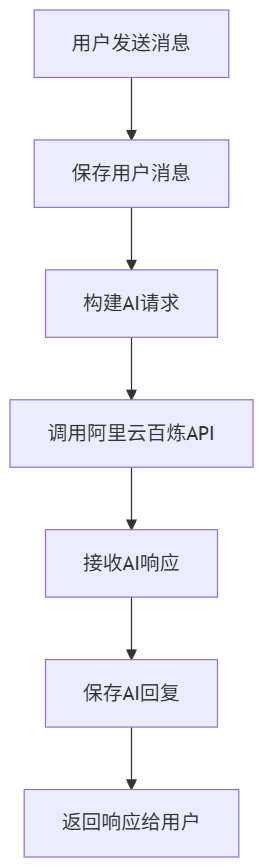
**数据库设计**

对话消息表(t\_chat\_message)

|  |
| --- |
| CREATE TABLE IF NOT EXISTS t\_chat\_message (  message\_id BIGINT AUTO\_INCREMENT COMMENT '消息ID',  conversation\_id BIGINT COMMENT '对话ID',  sender ENUM('user', 'ai') NOT NULL COMMENT '发送者',  content TEXT NOT NULL COMMENT '消息内容',  create\_at DATETIME DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',  PRIMARY KEY (message\_id),  FOREIGN KEY (conversation\_id) REFERENCES t\_conversation(conversation\_id)  ); |

对话表(t\_conversation)

|  |
| --- |
| CREATE TABLE IF NOT EXISTS t\_conversation (  conversation\_id BIGINT AUTO\_INCREMENT COMMENT '对话ID',  user\_id INT COMMENT '用户ID',  title VARCHAR(255) NOT NULL COMMENT '对话标题',  create\_at DATETIME DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',  update\_at DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT '更新时间',  PRIMARY KEY (conversation\_id),  FOREIGN KEY (user\_id) REFERENCES t\_user(id)  ); |



## **2.3 功能实现**

主要功能的实现，可以按以下示例方式进行说明：

### **2.3.1 登录**

，其核心功能实现的伪代码如下所示：

|  |
| --- |
| // 登录功能伪代码  // 1. 前端发送登录请求  POST /api/users/login  Body: {  "email": "user@example.com",  "password": "userpassword"  }  // 2. UserController处理登录请求  @RestController  @RequestMapping("/api/users")  public class UserController {    @Autowired  private UserMapper userMapper;    @PostMapping("/login")  public ResponseEntity<Map<String, Object>> login(@RequestBody LoginRequest request) {  // 验证输入参数  if (request.getEmail() == null || request.getPassword() == null) {  return ResponseEntity.badRequest().body(Map.of("message", "邮箱和密码不能为空"));  }    // 在数据库中查找用户  User user = userMapper.findByEmail(request.getEmail());    // 验证用户是否存在及密码是否正确  if (user != null && verifyPassword(request.getPassword(), user.getPassword())) {  // 登录成功，生成token或session  String token = generateToken(user);    Map<String, Object> response = new HashMap<>();  response.put("message", "登录成功");  response.put("token", token);  response.put("userId", user.getId());  response.put("email", user.getEmail());    return ResponseEntity.ok(response);  } else {  // 登录失败  Map<String, Object> response = new HashMap<>();  response.put("message", "邮箱或密码错误");  return ResponseEntity.status(401).body(response);  }  }  }  // 3. UserMapper接口和实现  public interface UserMapper {  @Select("SELECT \* FROM t\_user WHERE email = #{email}")  User findByEmail(String email);  }  // 4. User实体类（已有）  public class User {  private String id;  private String email;  private String password;  // getters and setters...  }  // 5. 辅助方法  private boolean verifyPassword(String rawPassword, String encodedPassword) {  // 实现密码验证逻辑（如BCrypt加密对比）  return passwordEncoder.matches(rawPassword, encodedPassword);  }  private String generateToken(User user) {  // 生成JWT Token或其他认证令牌  return jwtTokenUtil.generateToken(user);  }  // 6. LoginRequest DTO类  public class LoginRequest {  private String email;  private String password;  // getters and setters...  } |

本功能的界面实现如下：

### **2.3.2 注册**

，其核心功能实现的伪代码如下所示：

|  |
| --- |
| // 注册功能伪代码  // 1. 前端发送注册请求  POST /api/users/register  Body: {  "email": "newuser@example.com",  "password": "newuserpassword"  }  // 2. UserController处理注册请求  @RestController  @RequestMapping("/api/users")  public class UserController {    @Autowired  private UserMapper userMapper;    @PostMapping("/register")  public ResponseEntity<Map<String, Object>> register(@RequestBody RegisterRequest request) {  // 验证输入参数  if (request.getEmail() == null || request.getPassword() == null) {  return ResponseEntity.badRequest().body(Map.of("message", "邮箱和密码不能为空"));  }    // 检查邮箱是否已存在  User existingUser = userMapper.findByEmail(request.getEmail());  if (existingUser != null) {  Map<String, Object> response = new HashMap<>();  response.put("message", "该邮箱已被注册");  return ResponseEntity.status(409).body(response);  }    // 创建新用户  User user = UserFactory.getUser();  user.setEmail(request.getEmail());  user.setPassword(encryptPassword(request.getPassword())); // 密码加密存储    // 保存到数据库  int result = userMapper.register(user);    // 返回结果  Map<String, Object> response = new HashMap<>();  if (result > 0) {  response.put("message", "注册成功");  response.put("userId", user.getId());  return ResponseEntity.ok(response);  } else {  response.put("message", "注册失败");  return ResponseEntity.status(500).body(response);  }  }    // 辅助方法：加密密码  private String encryptPassword(String rawPassword) {  // 使用BCrypt或其他加密算法加密密码  return passwordEncoder.encode(rawPassword);  }  }  // 3. UserMapper接口（需要添加register方法）  public interface UserMapper {  // ... existing code ...    @Insert("INSERT INTO t\_user(email, password) VALUES(#{email}, #{password})")  @Options(useGeneratedKeys = true, keyProperty = "id")  int register(User user);    // ... existing code ...  }  // 4. RegisterRequest DTO类  public class RegisterRequest {  private String email;  private String password;  // getters and setters...  }  // 5. UserFactory类（已有）  public class UserFactory {  private static User user = new User();  public static User getUser() {  return user;  }  } |

本功能的界面实现如下：

### 2.3.3学习搭子AI对话模块

其核心功能实现的伪代码如下所示：

|  |
| --- |
| // 学习搭子AI对话模块伪代码  // 1. 前端发起对话请求  POST /api/chat/send  Body: {  "conversationId": 12345,  "content": "帮我解释一下量子力学的基本概念",  "userId": "user001"  }  // 2. ChatAIController处理对话请求  @RestController  @RequestMapping("/api/chat")  public class ChatAIController {    @Autowired  private AliyunBailianApplicationService aliyunBailianApplicationService;    @Autowired  private ChatMessageMapper chatMessageMapper;    @Autowired  private ConversationMapper conversationMapper;    @PostMapping("/send")  public ResponseEntity<Map<String, Object>> sendMessage(@RequestBody ChatRequest request) {  try {  // 验证用户输入  if (request.getContent() == null || request.getContent().trim().isEmpty()) {  return ResponseEntity.badRequest()  .body(Map.of("error", "消息内容不能为空"));  }    // 保存用户消息到数据库  ChatMessage userMessage = new ChatMessage();  userMessage.setConversationId(request.getConversationId());  userMessage.setSender("user");  userMessage.setContent(request.getContent());  userMessage.setCreateAt(new Date());  chatMessageMapper.insertMessage(userMessage);    // 获取对话历史用于上下文（最多取最近10条消息）  List<ChatMessage> history = chatMessageMapper  .selectRecentMessagesByConversationId(request.getConversationId(), 10);    // 构建发送给AI的上下文  String context = buildContext(history);    // 构建AI请求参数  AliyunBailianApplication appRequest = new AliyunBailianApplication();  appRequest.setContent(context); // 包含历史对话上下文  appRequest.setMemoryId("memory\_" + request.getConversationId()); // 对话记忆ID    // 调用阿里云百炼AI服务  String aiResponse = aliyunBailianApplicationService.callWithMessage(appRequest);    // 保存AI回复到数据库  ChatMessage aiMessage = new ChatMessage();  aiMessage.setConversationId(request.getConversationId());  aiMessage.setSender("ai");  aiMessage.setContent(aiResponse);  aiMessage.setCreateAt(new Date());  chatMessageMapper.insertMessage(aiMessage);    // 更新对话的最后更新时间  conversationMapper.updateConversationTime(request.getConversationId());    // 构造返回给前端的响应  Map<String, Object> response = new HashMap<>();  response.put("success", true);  response.put("messageId", aiMessage.getMessageId());  response.put("content", aiResponse);  response.put("sender", "ai");  response.put("createdAt", aiMessage.getCreateAt());    return ResponseEntity.ok(response);    } catch (Exception e) {  // 错误处理  Map<String, Object> errorResponse = new HashMap<>();  errorResponse.put("success", false);  errorResponse.put("error", "对话失败: " + e.getMessage());  return ResponseEntity.status(500).body(errorResponse);  }  }    // 构建对话上下文的方法  private String buildContext(List<ChatMessage> history) {  StringBuilder context = new StringBuilder();  for (ChatMessage message : history) {  if ("user".equals(message.getSender())) {  context.append("用户: ").append(message.getContent()).append("\n");  } else {  context.append("AI助手: ").append(message.getContent()).append("\n");  }  }  return context.toString();  }    // 创建新对话  @PostMapping("/conversations")  public ResponseEntity<Map<String, Object>> createConversation(  @RequestBody CreateConversationRequest request) {  try {  // 创建新的对话记录  Conversation conversation = new Conversation();  conversation.setUserId(Integer.parseInt(request.getUserId()));  conversation.setTitle(request.getTitle() != null ?  request.getTitle() : "新对话");  conversation.setCreateAt(new Date());  conversation.setUpdateAt(new Date());    // 保存到数据库  conversationMapper.insertConversation(conversation);    // 返回结果  Map<String, Object> response = new HashMap<>();  response.put("success", true);  response.put("conversationId", conversation.getConversationId());  response.put("title", conversation.getTitle());  response.put("createdAt", conversation.getCreateAt());    return ResponseEntity.ok(response);  } catch (Exception e) {  Map<String, Object> errorResponse = new HashMap<>();  errorResponse.put("success", false);  errorResponse.put("error", "创建对话失败: " + e.getMessage());  return ResponseEntity.status(500).body(errorResponse);  }  }    // 获取对话历史消息  @GetMapping("/conversations/{conversationId}/messages")  public ResponseEntity<Map<String, Object>> getConversationMessages(  @PathVariable Long conversationId) {  try {  // 从数据库获取对话历史  List<ChatMessage> messages = chatMessageMapper  .selectMessagesByConversationId(conversationId);    Map<String, Object> response = new HashMap<>();  response.put("success", true);  response.put("messages", messages);    return ResponseEntity.ok(response);  } catch (Exception e) {  Map<String, Object> errorResponse = new HashMap<>();  errorResponse.put("success", false);  errorResponse.put("error", "获取对话历史失败: " + e.getMessage());  return ResponseEntity.status(500).body(errorResponse);  }  }  }  // 3. AliyunBailianApplicationService实现AI调用  @Service  public class AliyunBailianApplicationServiceImpl  implements AliyunBailianApplicationService {    // 创建阿里云百炼客户端  public static com.aliyun.bailian20231229.Client createClient() throws Exception {  com.aliyun.teaopenapi.models.Config config = new com.aliyun.teaopenapi.models.Config()  .setAccessKeyId(System.getenv("ALIBABA\_CLOUD\_ACCESS\_KEY\_ID"))  .setAccessKeySecret(System.getenv("ALIBABA\_CLOUD\_ACCESS\_KEY\_SECRET"));  // 设置接入点  config.endpoint = "bailian.ap-southeast-1.aliyuncs.com";  return new com.aliyun.bailian20231229.Client(config);  }    // 调用阿里云百炼应用  @Override  public String callWithMessage(AliyunBailianApplication appRequest) throws Exception {  // 创建客户端  com.aliyun.bailian20231229.Client client = createClient();    // 构建请求  com.aliyun.bailian20231229.models.CreateChatAppCompletionRequest request =  new com.aliyun.bailian20231229.models.CreateChatAppCompletionRequest();    // 设置应用ID  request.setAppId(appRequest.getAppId());    // 设置对话记忆ID（用于保持上下文）  if (appRequest.getMemoryId() != null) {  request.setMemoryId(appRequest.getMemoryId());  }    // 设置用户输入内容  request.setPrompt(appRequest.getContent());    // 设置知识库ID（如果有的话）  if (appRequest.getKnowledgeBaseId() != null &&  !appRequest.getKnowledgeBaseId().isEmpty()) {  request.setKnowledgeIds(appRequest.getKnowledgeBaseId());  }    // 发起请求  com.aliyun.bailian20231229.models.CreateChatAppCompletionResponse response =  client.createChatAppCompletion(request);    // 解析并返回AI的回复内容  if (response.getBody() != null &&  response.getBody().getData() != null &&  response.getBody().getData().getResponse() != null) {    return response.getBody().getData().getResponse().getContent();  }    // 如果没有获取到回复内容，抛出异常  throw new RuntimeException("未能从AI获取有效回复");  }  }  // 4. 实体类定义  public class ChatMessage {  private Long messageId;  private Long conversationId;  private String sender; // "user" 或 "ai"  private String content;  private Date createAt;    // getters and setters...  }  public class Conversation {  private Long conversationId;  private Integer userId;  private String title;  private Date createAt;  private Date updateAt;    // getters and setters...  }  // 5. DTO类定义  public class ChatRequest {  private Long conversationId;  private String content;  private String userId;    // getters and setters...  }  public class CreateConversationRequest {  private String userId;  private String title;    // getters and setters...  }  // 6. Mapper接口定义  public interface ChatMessageMapper {  // 插入新消息  @Insert("INSERT INTO t\_chat\_message(conversation\_id, sender, content, create\_at) " +  "VALUES(#{conversationId}, #{sender}, #{content}, #{createAt})")  @Options(useGeneratedKeys = true, keyProperty = "messageId")  void insertMessage(ChatMessage message);    // 根据对话ID查询所有消息  @Select("SELECT \* FROM t\_chat\_message " +  "WHERE conversation\_id = #{conversationId} " +  "ORDER BY create\_at ASC")  List<ChatMessage> selectMessagesByConversationId(Long conversationId);    // 查询最近N条消息  @Select("SELECT \* FROM t\_chat\_message " +  "WHERE conversation\_id = #{conversationId} " +  "ORDER BY create\_at DESC LIMIT #{limit} " +  "ORDER BY create\_at ASC")  List<ChatMessage> selectRecentMessagesByConversationId(  Long conversationId, int limit);  }  public interface ConversationMapper {  // 插入新对话  @Insert("INSERT INTO t\_conversation(user\_id, title, create\_at, update\_at) " +  "VALUES(#{userId}, #{title}, #{createAt}, #{updateAt})")  @Options(useGeneratedKeys = true, keyProperty = "conversationId")  void insertConversation(Conversation conversation);    // 更新对话时间  @Update("UPDATE t\_conversation SET update\_at = #{updateAt} " +  "WHERE conversation\_id = #{conversationId}")  void updateConversationTime(Long conversationId);  }  // 7. 阿里云百炼应用请求参数类  @Data  @AllArgsConstructor  @NoArgsConstructor  public class AliyunBailianApplication {  private String content; // 用户输入内容  private final String appId = "bf963293c1d54f4085a2e210f69c6d0a"; // 固定应用ID  private String memoryId; // 对话记忆ID  private List<String> knowledgeBaseId = List.of("uhym8y9eq3"); // 知识库ID列表  } |

# **3.设计总结**

## **3.1 设计过程碰到的难点与解决方法**

*不能少于5点，撰写时删除此行*

（1）问题1：

解决方案：

（2）问题2：

解决方案：

（3）问题3：

解决方案：

（4）问题4：

解决方案：

（5）问题5：

解决方案：

（6）问题6：

解决方案：

（7）问题7：

解决方案：

（8）问题8：

解决方案：

（9）问题9：

解决方案：

（10）问题10：

解决方案：

## **3.2 成果与收获**

*不能少于600字，，撰写时删除此行*

|  |
| --- |
| 评语：报告撰写选择一项。  成绩：选择一项。 |