

软件工程 · 第二次团队作业 | 概要设计阶段

智班工坊 · 班小助

系统设计说明书（System Design Specification）V1.0

文档编号	ZBGF-SDS-2026-002
项目名称	班小助 BanXiaoZhu —— 校园通知与事务智能体
团队名称	智班工坊
团队成员	3124004185 肖武阳（队长）；3124004175 刘旭波；3124004181 王梓瀚
所属课程	软件工程（计科 24 级 78 班）
编写日期	2026 年 11 月 20 日
文档版本	V1.0（对应第二次团队作业提交版）
密级	公开（课程作业）

1 引言

1.1 编写目的

本文档是「班小助 BanXiaoZhu」系统的设计说明书，用于把《软件需求规格说明书》中「要做什么」的结论，翻译成「怎么做」的工程方案：划分层次与模块、定义接口、说明关键算法、确定数据结构、规定界面与安全策略，并给出出错处理与演进路线。

本文档的读者包括：课程指导老师与助教（评审设计是否合理、是否逐层细化）；团队三名成员（作为编码与测试的唯一依据）；后续接手维护的同学（理解设计意图与取舍理由）。

本文档与另外两份材料配套使用：《智班工坊_数据库设计说明书》给出全部物理表结构、索引、ORM 映射与初始化数据；《原型设计》给出 16 个页面的界面稿与交互规则。三者共同构成本次作业的设计成果。

1.2 项目背景

班级微信群每天产生 4～6 条需要行动的通知：报名、材料、缴费、活动、临时调课。这些信息以「长文本 + 表情 + 接龙」的形式散落在聊天记录里，同学要靠肉眼筛、靠记性记、靠再问一遍确认。需求调研（见 SRS 第 2 章）显示，问题不在于「通知发得少」，而在于「信息的生产是碎片化的，消费却必须是精确的」——时间、地点、材料、截止日期，任何一个字段错了都会造成真实损失（白跑一趟、错过截止、材料不齐）。

班小助的设计目标，是把一段自然语言的班级通知，转成一张**字段可核对、原文可溯源、时间可提醒**的结构化卡片，并在此基础上提供待办管理、三级提醒与有依据的班级问答。系统不追求「大而全的校园平台」，只把「通知 → 待办 → 提醒 → 答疑」这一条链路做扎实。

1.3 术语与缩写

表 1-1 术语与缩略语

术语 / 缩写	含义与说明
班小助 / BanXiaoZhu	本项目名称，一个面向班级事务的智能体应用。
智能体（Agent）	由大模型驱动、围绕特定目标完成任务的软件单元。本系统设计两个智能体：通知解析智能体、班级答疑智能体。
结构化卡片	把通知文本抽取为「标题 / 截止时间 / 地点 / 材料 / 需要做的事」五类字段后的展示形态。
evidence（证据）	解析结果中每一字段在原文里的逐字片段，用于让用户核对「这个结论是从哪句话来的」。

术语 / 缩写	含义与说明
confidence（置信度）	模型对某字段的自评可信程度，取值 0～1；低于阈值 0.75 的字段在界面上标黄「待确认」。
三级提醒	同一待办在截止前 1 天 / 2 小时 / 10 分钟各提醒一次。
兜底（fallback）	大模型不可用或输出不合规时，改用规则抽取，保证功能不中断。
REST 接口	前后端之间基于 HTTP + JSON 的调用约定。
UML	统一建模语言，本文使用用例图、活动图、类图、时序图、协作图五种视图。
ER 图	实体—联系图，描述数据的概念结构。
ORM	对象关系映射，把数据表映射为程序中的对象。

1.4 参考资料

- GB/T 8567—2006《计算机软件文档编制规范》
- GB/T 9385—2008《计算机软件需求规格说明规范》
- GB/T 8567—2006 中「系统/子系统设计（结构设计）说明」章节模板
- 《智班工坊_软件需求规格说明书》V1.1（本项目第一次团队作业交付物）
- 《智班工坊_数据库设计说明书》V1.0（本次作业配套文档）
- UML 2.5 规范（OMG）；FastAPI、python-docx、DeepSeek API 官方文档

2 总体设计

2.1 需求概述

系统面向班级内的四类角色：普通同学、班委、任课/辅导员（信息发布方）、系统维护者。核心需求可归纳为 11 条功能需求（FR-01～FR-11），摘要如下：

表 2-1 功能需求摘要

编号	功能名称	优先级	设计要点
FR-01	通知文本解析	高	大模型抽取五类字段 + 证据 + 置信度；JSON 强约束；失败降级规则抽取
FR-02	字段溯源与原文高亮	高	按字段回填原文片段并高亮；证据缺失的字段不采信
FR-03	待办生成与管理	高	解析结果由用户确认后入待办；支持编辑标题/截止时间、删除
FR-04	三级提醒	高	守护线程每 60 秒扫描；1 天/2 小时/10 分钟三档；同档去重
FR-05	课表与日程汇总	中	按周聚合待办与课程，冲突检测（后续迭代）
FR-06	一句话周计划	中	把当周待办组织成日程建议（后续迭代）
FR-07	截图/图片通知解析	中	OCR + 多模态解析（后续迭代）
FR-08	纠错回流	中	用户纠正字段后沉淀为样例，改进提示词（后续迭代）
FR-09	班级事务答疑	高	资料库检索 + 提示词约束 + 出处标注 + 无依据拒答
FR-10	班委后台与资料维护	中	通知/资料/提醒日志的增删改查与角色权限控制
FR-11	硬件班牌联动	低	ESP32-S3 班牌播报（加分项，后续迭代）

2.2 设计目标与原则

- **分层清晰**：接入层、前端层、应用层、智能体层、数据与模型层各自职责单一，任一层可独立替换。
- **模型可替换**：大模型只出现在「智能体层」的两个提示词模板之后，换模型只改配置，不改业务代码。
- **结论可核对**：任何模型输出都必须带原文证据，界面把证据显示给用户，避免「看起来很对」。
- **降级不中断**：模型超时、返回非 JSON、字段缺失时，自动降级为规则抽取，页面永远有结果。
- **先窄后宽**：一条链路（通知→待办→提醒）做透，再横向扩展，避免需求铺得太开而每项都不深。

- **可测试**：每个关键算法（证据校验、置信度、提醒去重）都能用确定性用例验收。

2.3 系统总体架构

系统采用五层结构。自上而下：接入层负责访问入口；前端层负责交互与呈现；应用层（接口层）负责鉴权、校验与业务编排；智能体层负责与大模型交互并做结果校验；数据与模型层负责持久化与外部模型服务。

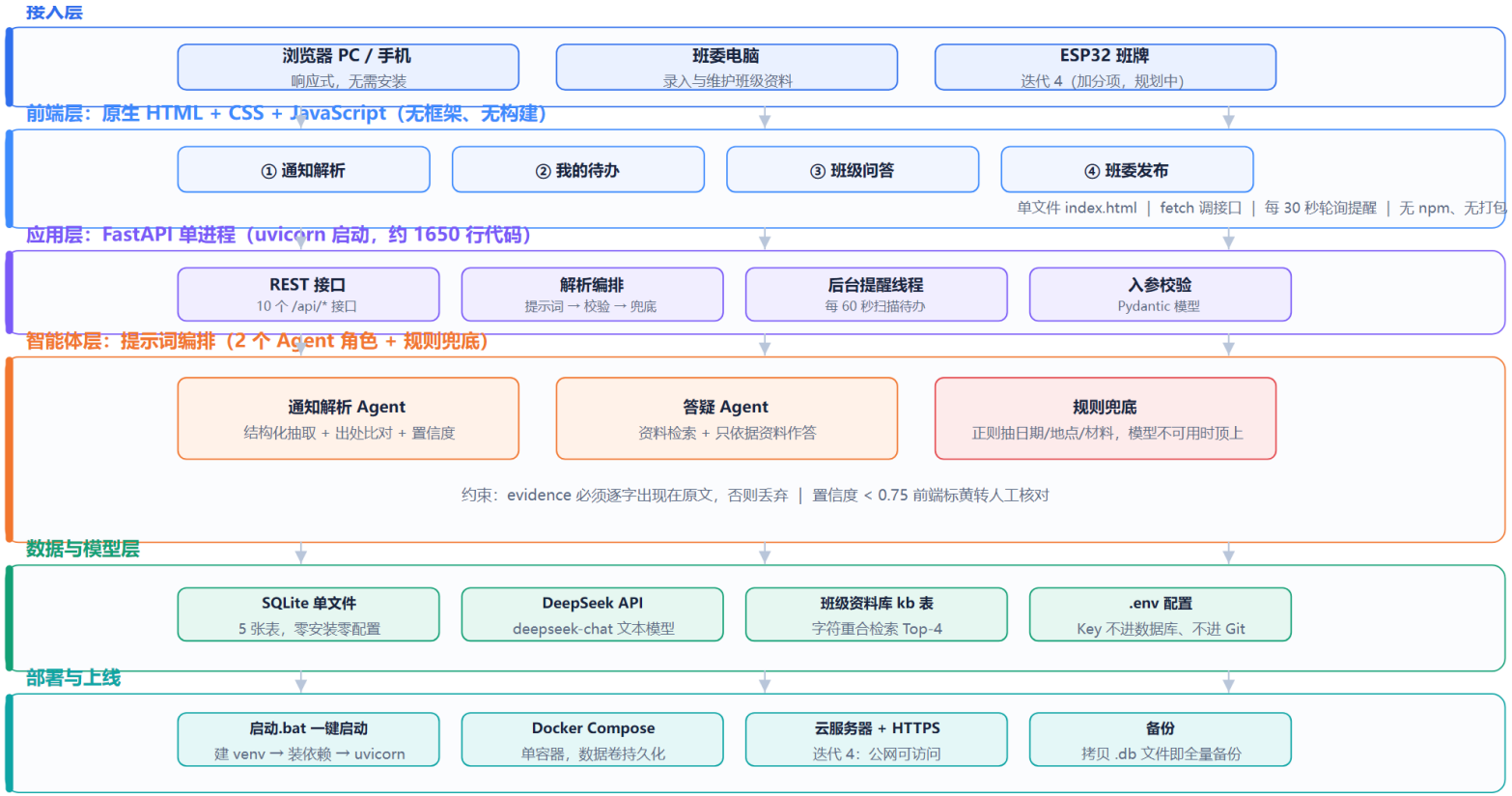
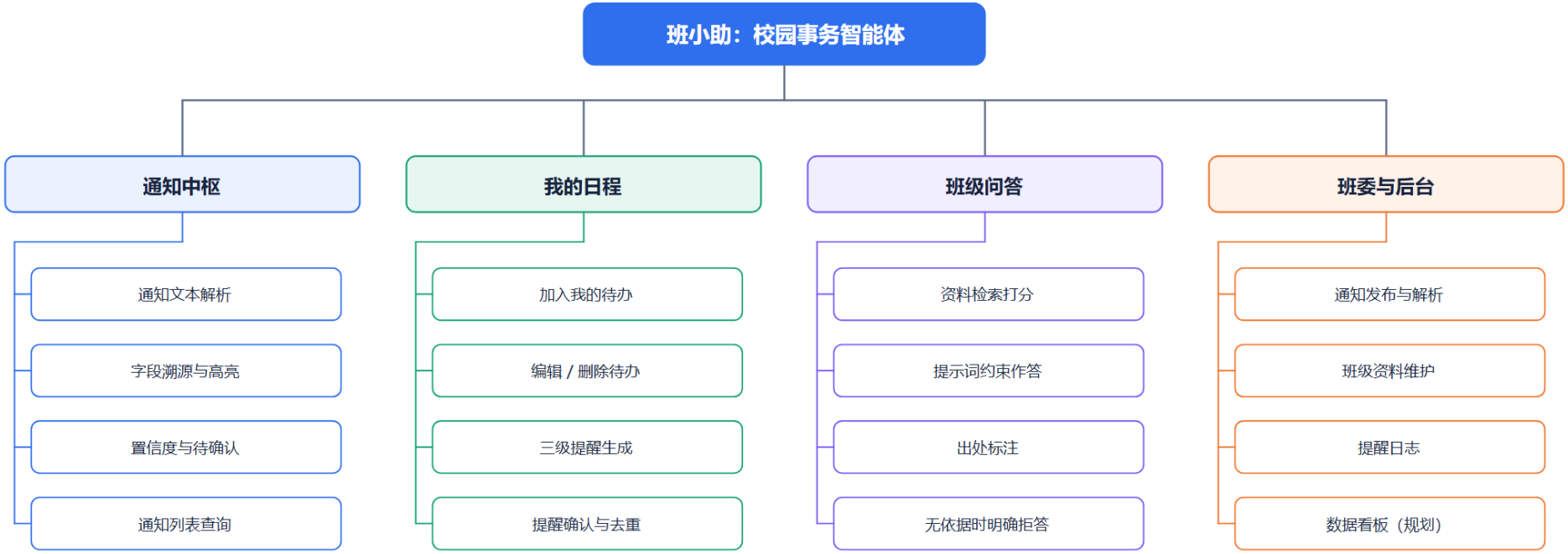


图 2-1 班小助系统分层架构

分层的意义在于「变化被隔离」：换界面只动前端层；换数据库只动数据层；换模型只动智能体层；而接口层的 REST 契约保持稳定，三层的改动互不传导。

2.4 模块划分

按功能域把系统划分为四个模块组，共 16 个功能模块（见图 2-2）。其中「数据看板」为迭代 3 之后的规划项，本期仅保留接口位。



说明：前三个分组的模块已纳入本期设计与开发范围；「数据看板」为迭代 3 之后的规划项，本期仅保留接口位。

图 2-2　系统功能模块结构图

表 2-2 模块职责划分

模块组	模块	职责	对应需求
通知中枢	通知文本解析	调用模型抽取五类字段，输出证据与置信度	FR-01
通知中枢	字段溯源与高亮	按字段回填原文片段，界面高亮定位	FR-02
通知中枢	置信度与待确认	低于阈值标黄，提示用户核对	FR-01/02
通知中枢	通知列表查询	按时间倒序查看历史解析记录	FR-01
我的日程	加入待办	用户确认后把建议待办写入待办表	FR-03
我的日程	编辑 / 删除待办	改标题与截止时间、删除待办并清理提醒	FR-03
我的日程	三级提醒生成	守护线程扫描到期待办，按三档写入提醒	FR-04
我的日程	提醒确认与去重	确认后置 acked_at，同档不重复	FR-04
班级问答	资料检索打分	字符重合度打分，取前 4 条资料	FR-09
班级问答	提示词约束作答	只依据资料作答，禁止自由发挥	FR-09
班级问答	出处标注	回答末尾标注引用资料序号	FR-09
班级问答	无依据拒答	资料不足时明确说「没有资料」并给出建议	FR-09
班委与后台	通知发布与解析	班委贴通知原文，走同一解析链路	FR-10
班委与后台	班级资料维护	资料的增删改查，支撑答疑质量	FR-10
班委与后台	提醒日志	查看已生成/已确认的提醒记录	FR-04/10
班委与后台	数据看板（规划）	解析量、纠正率、提醒到达率统计	FR-10

2.5 运行环境

表 2-3 运行环境要求

项目	最低要求	说明
服务端操作系统	Windows 10 / macOS 13 / Linux（含 Docker）	跨平台，同一份代码三种方式启动
运行时	Python 3.10 及以上（推荐 3.12）	使用标准库 sqlite3、threading，无重型依赖
服务进程	单进程即可（Uvicorn）	班级量级下无需多进程与反向代理
数据库	SQLite 3（单文件）	随应用发布，免安装、免运维
浏览器	Chrome / Edge 100+	支持 fetch、CSS Grid
外部服务	DeepSeek Chat API（HTTPS 443）	需要有效 API Key；不可用时自动降级
网络	解析与答疑需可访问外网；其余功能离线可用	待办、提醒均在本地完成

2.6 关键设计取舍

表 2-4 关键设计取舍

设计点	选定方案	放弃的方案	理由
前端形态	原生 HTML/CSS/JS 单页	Vue3 + 构建工具	四人周内交付，无构建链路故障面；样式与逻辑可直接阅读
数据库	SQLite 单文件	PostgreSQL + 向量库	班级量级（千级记录）无并发压力；向量检索用字符打分即可满足 FAQ 场景
定时任务	进程内守护线程	Celery + 消息队列	只服务一个班级，无需分布式调度；少一套中间件即少一类故障
智能体编排	提示词模板 + 结果校验（2 个智能体）	可视化工作流平台编排 6 个智能体	链路只有两段，代码内编排更可调试、更可复现
图片通知	本期只支持文本粘贴	直接上多模态 OCR	先验证「字段抽取是否可信」，再扩输入形态，避免同时引入两个不确定因素

3 接口设计

3.1 用户接口（用例视图）

系统对外暴露两类使用者：班级成员（普通同学、班委）通过浏览器访问；系统内部由守护线程定时驱动提醒生成；解析与答疑通过外部大模型服务完成抽取得出结论。用例视图如下：

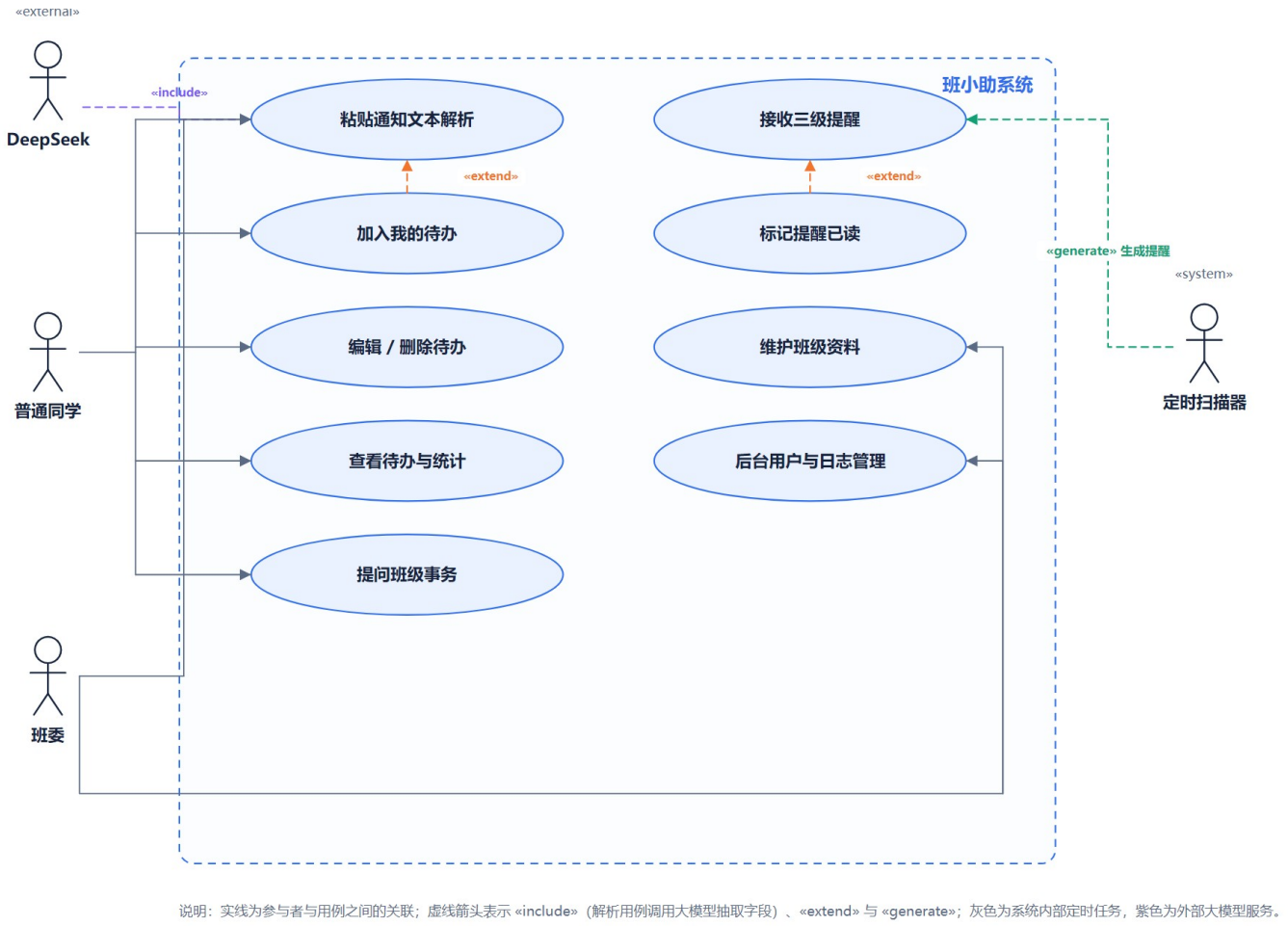


图 3-1　系统用例图

用例之间的关系：解析用例 «include» 调用大模型抽取字段；「加入我的待办」与「标记提醒已读」分别 «extend» 于解析用例与提醒用例，表示它们在主流程之外被用户可选触发；提醒用例由定时扫描器 «generate» 驱动。

3.2 内部接口（REST API）

前后端通过 HTTP + JSON 通信，统一前缀 /api，返回体统一为 JSON 对象，出错时返回 {"detail": "人类可读的中文提示"} 与相应状态码。接口清单如下：

表 3-1 接口清单（17 条路由）

方法	路径	入参要点	返回要点	用途
GET	/api/health	—	ok / model / has_key	健康检查与模型连通性
POST	/api/login	name, role	用户对象（含 id）	以昵称 + 角色进入，无需密码
POST	/api/parse	text, user_id	notice_id、五类字段、evidence、置信度、suggest	解析通知文本（不自动建待办）
GET	/api/notices	limit	通知列表（含原始文本与字段）	查看历史解析
GET	/api/tasks	user_id	该用户待办列表	我的待办页
POST	/api/tasks	user_id, title, due_at, notice_id	新建的待办对象	用户点「加入我的待办」
PUT	/api/tasks/{id}	title, due_at	更新后的待办	编辑待办（改时间会重建提醒）
DELETE	/api/tasks/{id}	—	{"ok": true}	删除待办（连带清理提醒）
POST	/api/tasks/{id}/done	—	{"ok": true}	标记完成
GET	/api/reminders	user_id	未确认提醒列表	前端 30 秒轮询
POST	/api/reminders/{id}/ack	—	{"ok": true}	确认提醒（去重）
POST	/api/ask	question, user_id	answer、sources、used_ids	班级事务答疑
GET	/api/kb	—	资料列表	班级资料库
POST	/api/kb	question, answer	新建的资料	班委新增资料
PUT	/api/kb/{id}	question, answer	更新后的资料	班委修改资料
DELETE	/api/kb/{id}	—	{"ok": true}	班委删除资料
GET	/	—	index.html	单页应用入口

3.3 核心接口的报文体示例

解析接口的返回体是整个系统数据结构的中枢，设计上要求「字段 + 证据 + 置信度」三者同时返回：

POST /api/parse 请求：{"text": "【通知】请各班于 11 月 8 日 18:00 前，把奖学金申请材料交到实验楼 A302 办公室……", "user_id": 1}

响应：{"notice_id": 12, "title": "奖学金材料收集", "deadline": "2026-11-08T18:00", "location": "实验楼 A302 办公室", "materials": ["成绩单", "家庭经济困难证明", "个人申请书（800 字以内）"], "action": "按材料清单准备并提交", "confidence": 0.95, "evidence": {"deadline": "11 月 8 日 18:00 前", "location": "实验楼 A302 办公室"}, "suggest": {"title": "奖学金材料收集", "due_at": "2026-11-08T18:00"}}

- `notice_id` 指向已落库的原文记录，界面据此做高亮回溯；
- `suggest` 是「建议待办」，只有用户点击确认后才写入待办表——避免误解析直接污染用户日程；
- `evidence` 里没有出现的字段，前端一律视为「证据不足」，即使模型给了值也显示为待确认。

3.4 外部接口

表 3-2 外部接口与降级策略

外部系统	接口形式	用途	异常与降级
DeepSeek Chat API	HTTPS POST /chat/completions，JSON 模式	字段抽取与答疑生成	超时/限流/非 JSON → 规则兜底或明确提示
班级群通知	人工粘贴文本（本期）	解析输入来源	格式混乱由提示词与规则共同容忍
SQLite 文件	进程内文件读写	全部持久化	写锁冲突重试；文件损坏从备份恢复
浏览器剪贴板	Clipboard API	一键粘贴通知	无权限时提示手动粘贴

4 功能模块详细设计

4.1 通知解析模块

4.1.1 处理流程

解析是系统的入口环节，包含「模型抽取 → 证据校验 → 置信度计算 → 卡片渲染」四个阶段，并在两处设置了降级分支：证据不命中时改用规则抽取；置信度不足时把字段标为待确认。

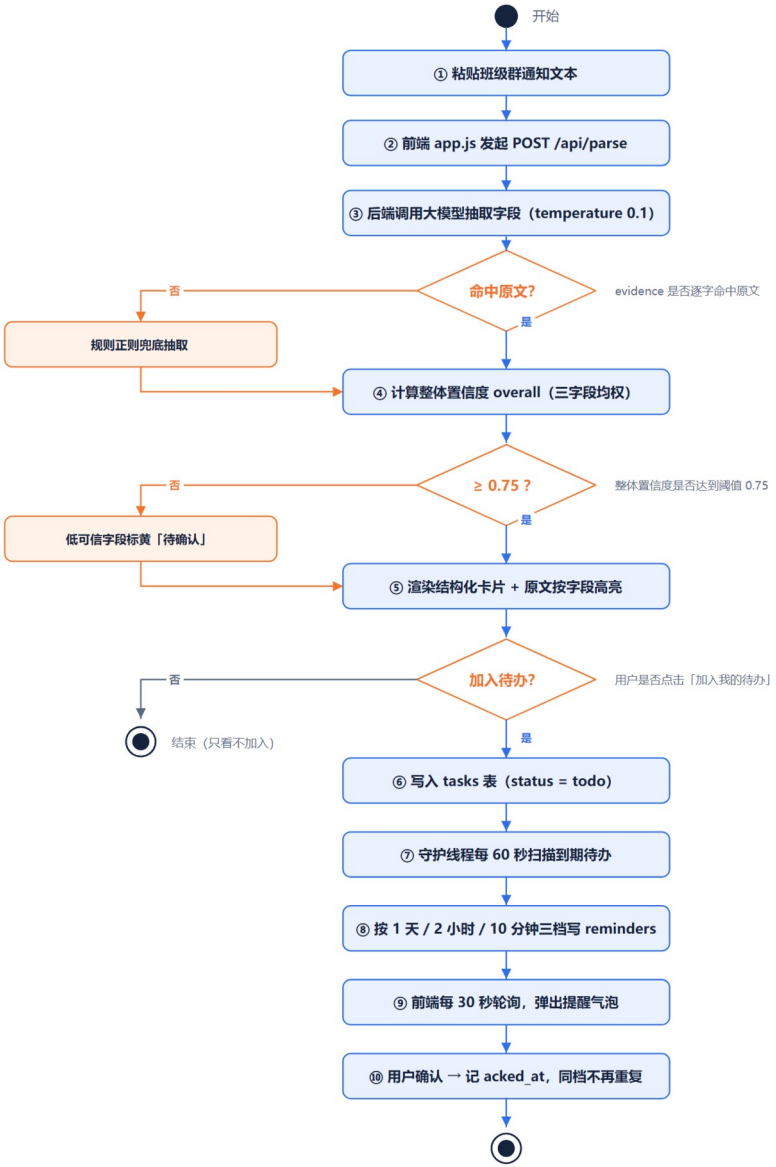


图 4-1　通知解析与三级提醒活动图

4.1.2 字段抽取算法

抽取采用「提示词约束 + JSON 模式 + 确定性校验」三段式，而不是让模型自由生成段落：

- **输入约束**：系统提示词要求模型只输出一个 JSON 对象，字段固定为 title、deadline、location、materials、action、confidence、evidence，多一个字段都不返回。
- **格式约束**：调用时启用 JSON 输出模式（response_format=json_object），并令 temperature=0.1，把随机性压到最低。
- **时间归一**：要求 deadline 输出 ISO 8601 格式（如 2026-11-08T18:00）；相对时间（明天下午三点）由提示词中给定的「今天是哪一天」推算，避免模型自行猜测年份。
- **证据约束**：evidence 必须逐字复制原文片段，不允许改写、不允许翻译；后端对每个证据做子串匹配校验，不命中的字段视为抽取失败。
- **置信度**：整体置信度取 title、deadline、location 三个关键字段的均值，并夹在 0～1 之间；低于 0.75 时字段在界面上标黄「待确认」。

伪代码（后端校验环节）：

```
for field in (title, deadline, location, materials):
    ev = data.evidence.get(field)
    if ev and ev not in raw_text:          # 证据不逐字命中
        data[field] = None                 # 该字段不采信
        data.confidence[field] = 0
overall = mean(conf[title], conf[deadline], conf[location])
```

4.1.3 时序设计

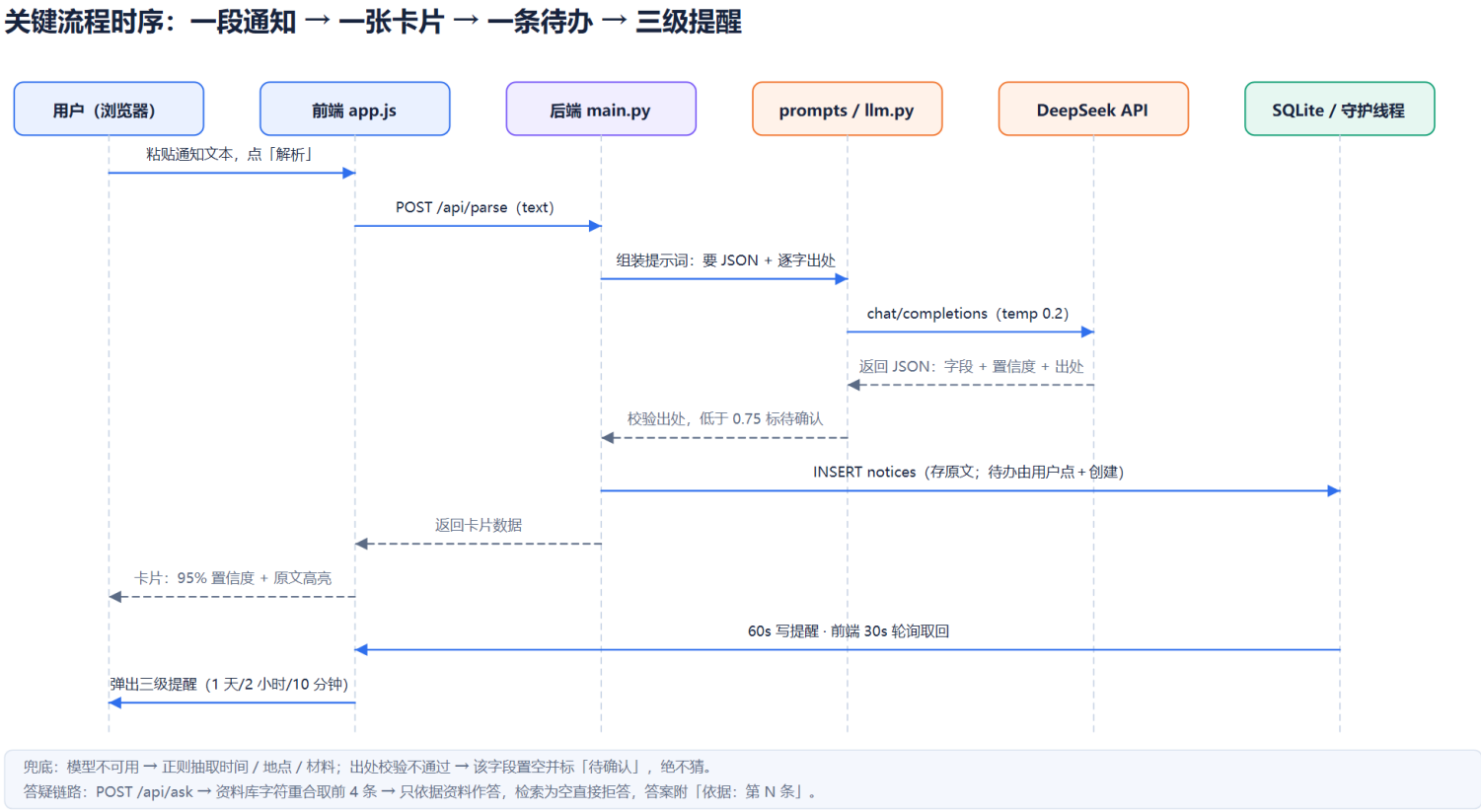


图 4-2 解析主链路时序图

时序图强调两点设计意图：第一，写库动作发生在「校验之后」，脏数据不进数据库；第二，待办写入被拆成独立的一次用户确认动作，解析与日程解耦。

4.1.4 对象协作设计

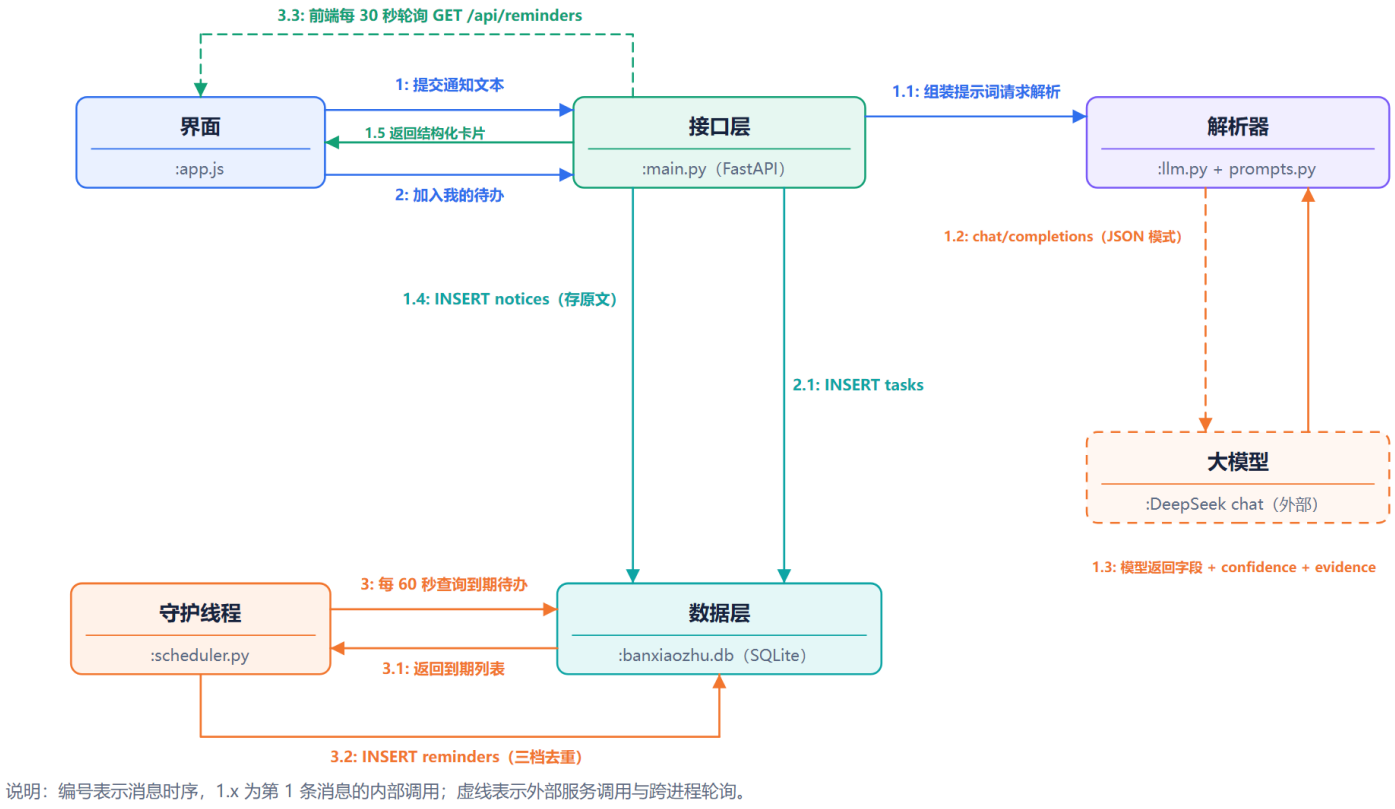


图 4-3 解析链路协作图

协作图给出同一链路中对象之间的消息编号：1 号消息进入接口层，1.1 组装提示词交给解析器，1.2/1.3 与模型往返，1.4 落库，1.5 返回卡片；2 号消息是用户确认后的待办落库；3 号消息由守护线程在后台驱动，3.1/3.2 完成提醒写入，3.3 由前端轮询取走。

4.2 我的日程与三级提醒模块

提醒采用「拉式生成 + 主动轮询」：后台线程只负责按时间窗把提醒写进数据库，前端每 30 秒拉一次未确认提醒并弹气泡。这样即使页面关掉再打开，未读提醒也不会丢。

- **扫描周期**：守护线程每 60 秒执行一次扫描，扫描成本与待办数量线性相关，班级量级下可忽略。
- **命中条件**：待办状态为 todo，且 due_at 落在 (now + level) 的窗口内，level 取 1440 / 120 / 10 分钟。
- **幂等保证**：提醒表对 (task_id, level) 建唯一约束，重复扫描不会产生重复提醒；写入时用 INSERT OR IGNORE，天然幂等。
- **编辑联动**：用户修改截止时间时，先删除该待办的历史提醒再按新时间重算，避免「旧时间提醒还在响」。
- **确认语义**：用户点「知道了」写入 acked_at，该条提醒不再出现在轮询结果里，但不删除记录（留作审计）。

提醒生成伪代码：

```
for task in tasks where status='todo' and due_at is not null:
    for level, label in [(1440, '提前 1 天'), (120, '提前 2 小时'), (10, '提前 10 分钟')]:
                                if 0 < due_at - now <= level:
        INSERT OR IGNORE INTO reminders(task_id, level, content) ...
```

4.3 班级问答模块

答疑不做「让模型自由回答」，而是「先检索、再约束、后标注」：

- **检索**：把资料库每条记录的「问题 + 答案」拼成文本，与用户提问做字符集合重合度打分，score = |Q ∩ D| / (|Q| + 1)，取分数最高的前 4 条。中文短问句下该指标足够区分相关性。
- **约束**：系统提示词规定「只依据给定资料作答；资料里没有的，明确说没有，并提示去问班委」，temperature 设为 0.3 以保持表达自然但不发散。
- **标注**：返回体携带 sources（命中的资料原文），界面在回答下方显示「依据：班级资料第 N 条」，用户可一键跳转核对。
- **拒答**：当检索结果为空或全部低于阈值时，直接返回模板话术，不调用模型，节省成本也避免编造。

4.4 班委与后台模块

后台面向班委，提供通知解析、资料维护、提醒日志三类操作。权限按角色粗粒度控制：普通同学只能读写自己的待办与提醒；班委额外拥有资料与通知的写权限。

表 4-1 角色权限矩阵

操作	普通同学	班委	说明
解析通知	√	√	两者都可粘贴解析
新增/编辑/删除自己的待办	√	√	只能操作本人待办
查看他人待办	×	√	班委看板（规划）
新增/编辑/删除班级资料	×	√	写操作限班委
查看提醒日志	×	√	含生成与确认时间
后台用户管理	×	√	规划项

4.5 模块间调用关系

表 4-2 模块间调用关系

调用方	被调用方	调用方式	说明
通知解析模块	智能体层（解析）	函数调用 + HTTPS	组装提示词、调用模型、校验结果
通知解析模块	数据层（notices）	写库	原文与字段一起落库，供溯源
我的日程模块	通知解析模块	读 suggest	用户确认后建待办，不重复解析
我的日程模块	数据层（tasks / reminders）	读写库	待办 CRUD 与提醒生成
班级问答模块	数据层（kb）	检索	打分取前 4 条
班级问答模块	智能体层（答疑）	HTTPS	约束作答 + 出处
后台模块	数据层（全部表）	读写库	资料维护与日志查看

5 数据设计

5.1 领域类图

领域模型包含 12 个类：User 抽象出 Student 与 Committee；Notice 是解析结果的主体，对应 TaskItem 待办；ServiceRequest 描述事务办理；Conversation / Message / AgentRun 记录人机交互过程；KnowledgeDoc 是班级资料；Device 为班牌设备；ReminderLog 记录提醒流水。

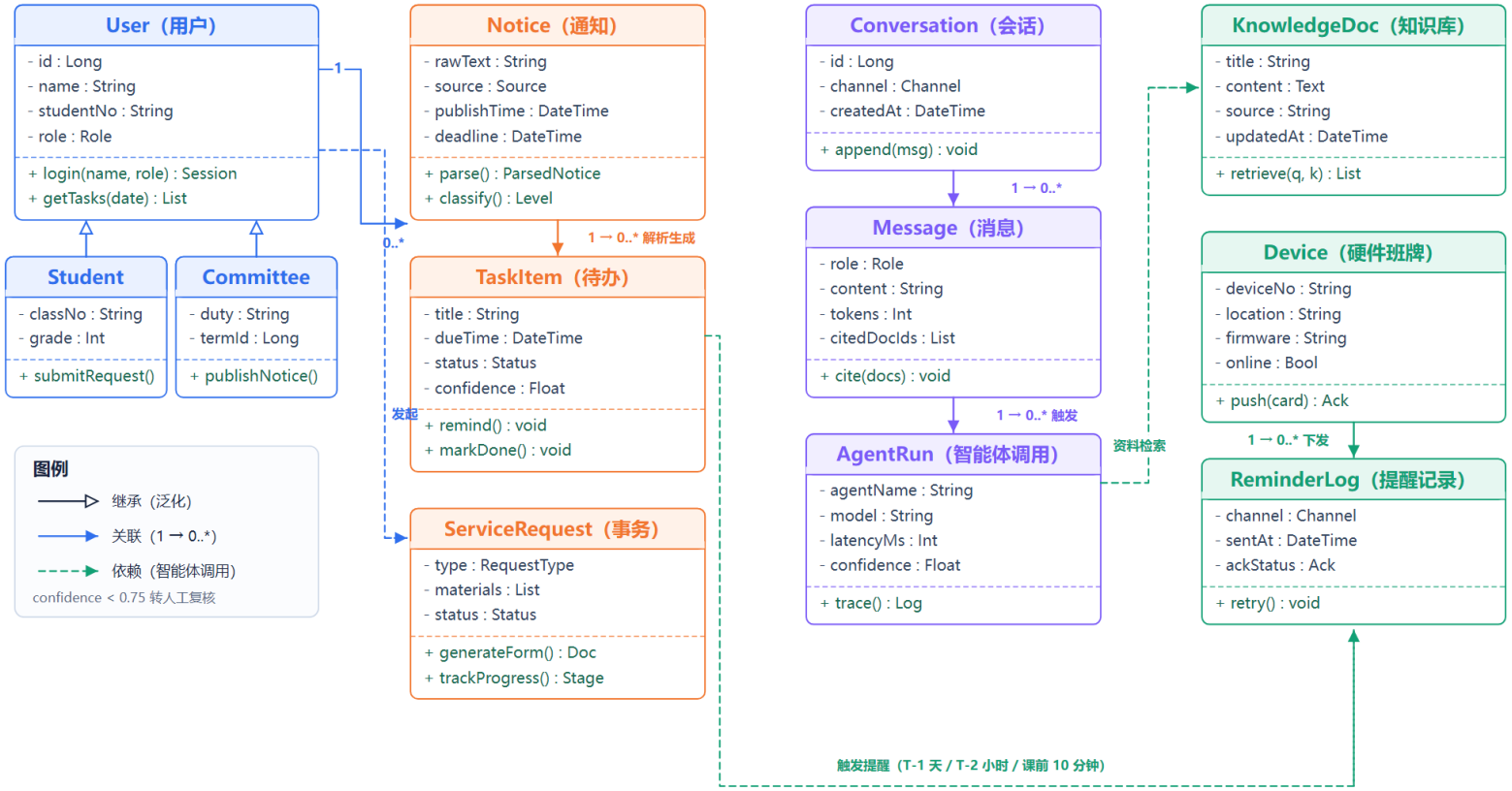


图 5-1 领域类图

类图与数据库表并非一一对应：本期物理实现把 User/Student/Committee 合并为 users 表，把 Conversation / Message / AgentRun 收敛为「不单独存储，日志留痕」的策略，以降低一期实现复杂度；ServiceRequest 与 Device 属后续迭代，仅保留设计。具体表结构见《数据库设计说明书》。

5.2 数据实体概览

表 5-1 数据实体概览

实体	对应表	关键字段	用途
用户	users	id, name, role	区分同学与班委，绑定待办归属
通知	notices	id, raw_text, 五类字段, confidence	解析结果与原文留痕
待办	tasks	id, user_id, notice_id, due_at, status	用户日程的最小单位
提醒	reminders	id, task_id, level, content, created_at, acked_at	三级提醒流水，唯一约束去重
班级资料	kb	id, question, answer	答疑检索语料

5.3 数据流

数据在系统中的流动可概括为三条：①原文 → 解析字段 → 卡片展示（读多写少，允许重复解析）；②建议待办 → 用户确认 → tasks → reminders（写操作，需保证幂等）；③资料录入 → kb → 检索 → 回答与出处（读多写少，靠班委维护质量）。三条流之间只通过「用户 id」与「通知 id」两个外键产生弱耦合，任一环节失败不影响其余环节。

6 界面设计

6.1 设计原则

- **一屏一件事**：前台四个页签分别对应「解析 / 待办 / 答疑 / 资料」，不做大杂烩首页。
- **结论先行**：卡片顶部给出一句话结论（标题 + 截止），细节与证据在下方可展开。
- **可信度可见**：每个字段右侧显示可信百分比，低于阈值转为「待确认」黄色态。
- **可回溯**：点击字段定位原文高亮，「结论从哪来」永远可查。
- **少一步**：粘贴即可解析，解析后一键加入待办，不要求填表单。

6.2 页面清单

表 6-1 页面清单（含后期规划页）

编号	页面	端	主要元素	对应原型
P-01	登录 / 身份选择	前台	昵称输入、角色选择（同学 / 班委）	原型 01
P-02	通知解析	前台	粘贴框、示例按钮、解析按钮、结果卡片区	原型 02
P-03	解析结果卡片	前台	五类字段、可信度、原文高亮、加入待办按钮	原型 03
P-04	我的待办	前台	待办列表、勾选完成、行内编辑、删除	原型 04
P-05	提醒气泡与提醒中心	前台	气泡、未读列表、确认按钮	原型 05
P-06	班级问答	前台	提问框、回答区、依据列表、常用问题	原型 06
P-07	班级资料库	前台/班委	资料列表、新增、编辑、删除	原型 07
P-08	我的 / 设置	前台	身份切换、模型状态、数据清理	原型 08
P-09	周视图日程（规划）	前台	周历、待办分布、冲突提示	原型 09
P-10	硬件班牌联动（规划）	前台	设备绑定、播报内容预览	原型 10
P-11	后台首页 / 数据看板	后台	解析量、提醒到达率、纠错率	原型 11
P-12	通知管理	后台	通知列表、详情、下线	原型 12
P-13	班级资料管理	后台	表格 + 编辑弹窗	原型 13
P-14	提醒日志	后台	记录列表、筛选、导出	原型 14
P-15	用户与角色	后台	用户列表、角色调整	原型 15
P-16	系统设置	后台	模型参数、提醒档位、阈值	原型 16

6.3 状态与交互规范

表 6-2 界面状态规范

状态	触发条件	表现	用户可做的动作
空态	首次进入或列表为空	居中提示 + 主操作按钮 + 一句示例	粘贴示例 / 新增
加载中	请求发出到返回	按钮置灰 + 文案变「解析中…」，卡片区骨架屏	等待，可取消
成功	返回 200 且字段完整	卡片渲染，可信度标签为绿色或灰色	加入待办 / 复制 / 再解析
待确认	某字段置信度 < 0.75	该字段标黄并附「依据在哪句」	人工核对原文后修改
降级	模型超时或返回非 JSON	顶部黄色横幅「已使用规则解析，请核对时间地点」	核对 / 重试
失败	网络或服务异常	红色提示 + 重试按钮 + 错误码	重试 / 切离线模式

6.4 视觉规范

表 6-3 视觉规范（与原型稿一致）

名称	色值	用途
主色 深蓝	#16233D	标题、表头、深色页底
强调 蓝	#2F6FED	主按钮、链接、进度
提示 橙	#F2762E	待确认、风险、超时
成功 绿	#12A075	解析成功、已完成
背景 浅蓝灰	#F4F7FC	页面背景、隔行底色
正文 墨	#1A2B4A	正文文字
辅助 灰	#5B6472	说明文字、次要信息

字体统一使用「微软雅黑」，正文 14px、辅助 12px、卡片标题 16px 加粗；圆角 10px，卡片阴影 0 1px 3px rgba(22,35,61,.08)。16 个页面的高保真原型见配套《原型设计》（墨刀可点击链接见博客「原型设计展示」一节）。

7 安全设计

7.1 密钥与配置

- 模型 API Key 只存于服务端 .env 文件，前端任何位置都不出现；代码仓库通过 .gitignore 排除该文件。
- 打包分发时支持「无 Key 版」，由使用者填入自己的 Key，避免把团队额度带出去。
- 配置读取集中在 config 模块，缺失 Key 时服务仍可启动，仅在解析/答疑接口返回明确提示。

7.2 输入校验与输出编码

- 所有写接口对必填字段做非空校验，返回 400 与中文提示（如「截止时间格式不对，应该是 2026-11-08T18:00 这样」）。
- 截止时间统一走 datetime.fromisoformat 解析，解析失败即拒绝，不做模糊猜测。
- 前端渲染用户可控文本（标题、资料、回答）一律经转义函数 escapeHtml 处理，避免注入。
- 后端不拼接 SQL 字符串，全部使用参数化查询。

7.3 权限与越权

本期采用「昵称 + 角色」的轻量身份（不做密码体系，符合课程场景）。服务端对每类写操作校验归属：待办的编辑/删除/完成必须满足 task.user_id 与请求中的 user_id 一致；资料写操作要求 role=committee。虽然前端已隐藏入口，但校验必须在服务端完成。

7.4 数据安全与备份

- 数据库为单文件，随应用目录存放；提供「导出备份」与「一键重置」两个管理动作。
- 提醒、解析记录保留原始文本，便于回溯；不采集手机号、学号以外与业务无关的个人信息。
- 班级群聊天原文属于班级内部信息，仅本班成员可访问，不做对外共享与二次分发。

8 出错处理设计

8.1 错误分类与提示

表 8-1 错误分类与处理

错误类型	触发场景	用户看到的提示	系统行为
输入为空	未粘贴通知就点解析	请先把通知内容粘贴进来	不发请求
时间格式非法	截止时间无法解析	截止时间格式不对，应该是 2026-11-08T18:00 这样	返回 400，不落库
模型超时	外部服务 60 秒无响应	模型响应慢，已用规则解析，请核对时间地点	降级规则抽取
模型返回非 JSON	输出被截断或格式错误	解析结果不稳定，已尽量提取关键字段	清洗后重试一次，再降级
字段证据缺失	证据不逐字命中	该字段标黄：待确认	该字段置空并要求人工核对
记录不存在	编辑已删除的待办	这条待办不存在	返回 404
无相关资料	答疑检索为空	班级资料里没有这方面的内容，建议直接问班委	不调用模型，省成本
数据库忙	并发写入冲突	稍等一下，正在保存	重试 3 次后报错

8.2 三级降级策略

- **一级（格式层）**：先尝试标准 JSON 解析；失败则剥掉可能的代码块围栏、截取首尾花括号再解析。
- **二级（规则层）**：正则抽取日期、时间、地点关键词与材料列表，产出低置信度但结构完整的卡片。
- **三级（提示层）**：兜底也失败时，明确告知用户「这次没解析出来」，并保留原文供人工阅读，绝不编造字段。

三级降级的意义在于：宁可让用户看到「不确定」，也不能让用户看到「看起来确定的错误」。本系统的核心风险不是「解析不出来」，而是「解析错了还理直气壮」。

8.3 日志与排错顺序

日志分三类：启动日志（端口、模型名、Key 是否加载）、请求日志（方法、路径、耗时、状态码）、异常日志（错误类型与关键字段）。排错优先级依次为：配置（Key/端口）→ 网络（能否访问模型）→ 数据库（文件权限）→ 业务（字段校验）→ 前端（脚本报错）。

9 系统演进与迭代计划

9.1 迭代计划

项目迭代计划（2026-11-03 → 2027-01-10）



图 9-1　迭代与里程碑

表 9-1 迭代计划

阶段	时间	目标	主要产出	评审点
阶段 0	第 1 周	立项与设计	需求规格、系统设计、数据库设计、原型	设计评审（本次作业）
迭代 1	第 2 周	通知中枢打磨	截图 OCR、纠错回流、评测集 30 条	解析准确率 ≥ 90%
迭代 2	第 3 周	日程与课表	周视图、冲突检测、一句话周计划	冲突识别可用
迭代 3	第 4 周	事务办理与答疑增强	材料清单生成、申请文本生成、资料质量评分	办理链路跑通
迭代 4	第 5 周	硬件与上线	ESP32 班牌、公网部署、监控	外网可访问

9.2 技术债登记

表 9-2 技术债登记

编号	技术债	影响	计划
TD-01	无密码身份体系	无法防冒名，仅适用课程场景	迭代 4 引入校园统一认证（如可行）
TD-02	提醒依赖前端轮询	页面关闭即收不到提醒	迭代 3 接入邮件/微信通知
TD-03	检索为字符重合	语义相近但用词不同的问句召回弱	资料量超过 500 条后引入向量检索
TD-04	单进程守护线程	无法横向扩容	部署到公网后改为独立调度进程
TD-05	日志未结构化	排错依赖人工阅读	迭代 4 接入 JSON 日志与检索