# 智学伴 —— AI 个性化学习与测评平台

## 一、项目简介

**智学伴** 是面向大学生（亦可拓展到中学/培训机构）的 AI 个性化学习与测评平台。用户上传教材、笔记或选择学习目标后，系统自动生成学习计划、出题测评、智能讲解错题，并通过可视化仪表盘展示学习进度与薄弱知识点。

目标：在比赛中展示一个完整的前后端分离系统，重点体现 AI 能力（题目生成、个性化学习路径、RAG 问答/错题讲解）与系统工程能力（部署、可扩展性、安全）。

---

## 二、设计目标与非功能需求

- **功能性目标**：自动生成学习计划、智能出题与批改、错题讲解、学习进度可视化、AI 实时答疑、知识位点追踪。
- **非功能目标**：稳定（支持并发用户演示），可扩展（AI 模块可替换），易部署（Docker + Nginx），安全（鉴权、API 限流、敏感信息过滤）。

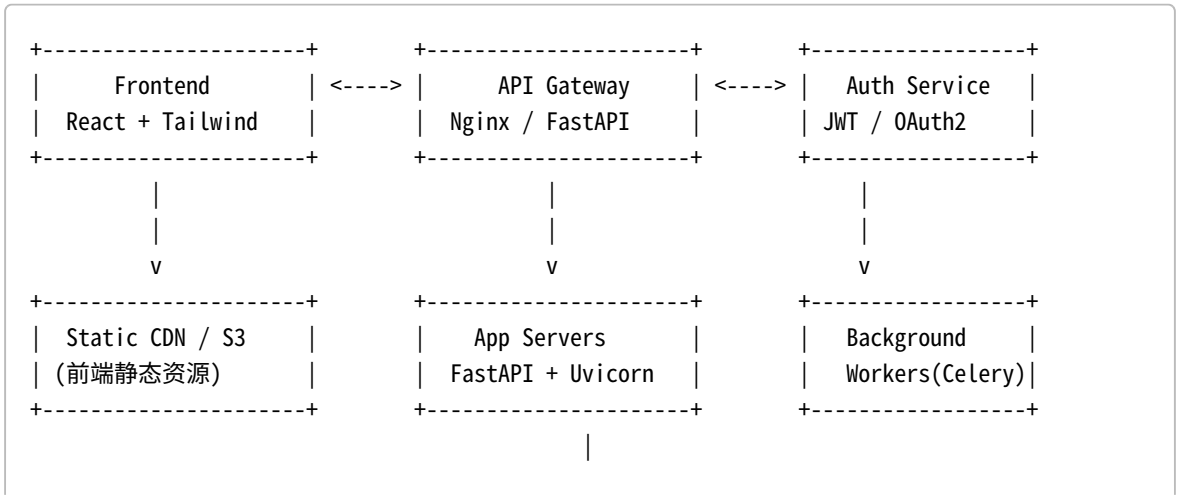
---

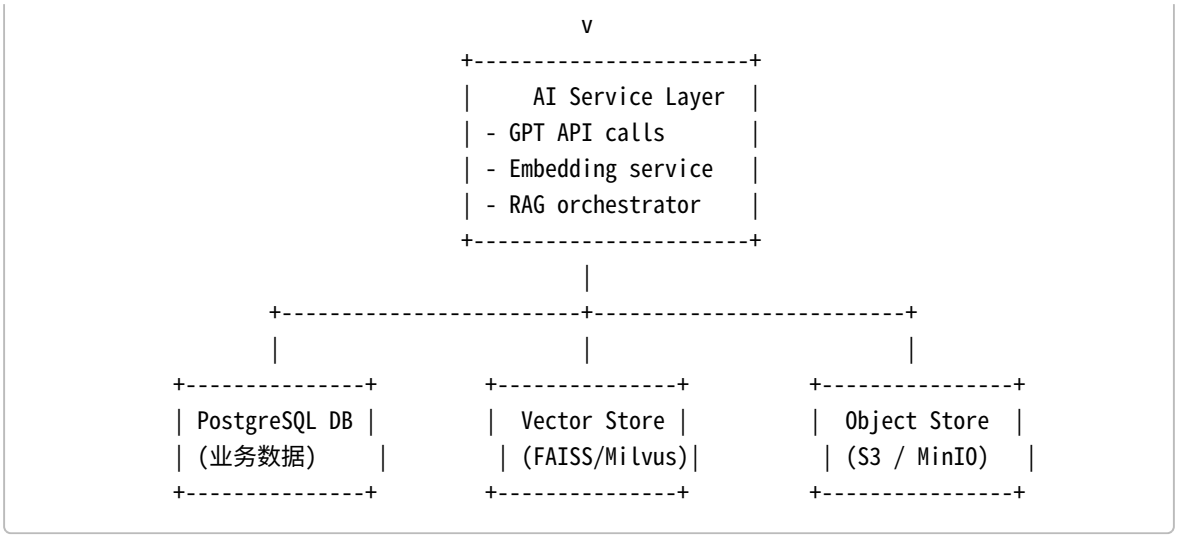
## 三、整体技术栈（推荐）

- 前端：React（或 Next.js） + Tailwind CSS + Recharts / ECharts
- 后端：FastAPI（或 Flask）+ Uvicorn/Gunicorn
- 数据库：PostgreSQL（业务数据）
- 缓存/队列：Redis + Celery（离线题库生成 / 批改任务）
- 向量检索：FAISS 或 Milvus（嵌入向量存储）
- AI：OpenAI GPT-4o（生成题目、解析、对话），OpenAI Embeddings（语义检索）或企业对接模型
- 文件解析：PyMuPDF、Tika（PDF/Word）
- 部署：Docker Compose / Kubernetes（选项），Nginx 反向代理

---

## 四、系统架构图（高层）






说明：Frontend 与 API Gateway（Nginx）对接，所有 AI 请求经由后端统一代理和限流。长耗时任务（批量出题、文件解析、嵌入计算）交给 Celery 异步执行。向量检索服务用于 RAG（基于用户上传资料的检索增强生成）。

---

# 五、模块分解（详细）

## 1. 前端模块（React）

- 登录/注册页（支持邮箱/手机号/第三方）
- 仪表盘（学习进度、正确率、今日任务）
- 学习计划页（查看/调整 AI 生成的学习计划）
- 测验页（在线做题、暂存、提交）
- 错题本/学习历史（逐题回顾、AI 解析）
- 上传资料页（PDF/Markdown/图片）
- AI 聊天助手（侧边栏，基于 RAG 的上下文问答）

**组件树（示例）**

```
App
 ├── Auth
 │    ├── LoginForm
 │    └── RegisterForm
 ├── Dashboard
 │    ├── ProgressCard
 │    └── TodayTask
 ├── StudyPlan
 │    ├── PlanList
 │    └── PlanEditor
 ├── Quiz
 │    ├── QuestionRenderer
 │    └── SubmitPanel
 ├── AIChat
 └── FileUploader
```

### 2. 后端模块（FastAPI）

- Auth Service（JWT）
- User Service（用户资料、学习偏好）
- Content Service（文件上传、解析、存储）
- AI Orchestrator（请求管理、RAG 流程）
- Quiz Service（题库管理、出题策略）
- Grading Service（自动批改、评分规则）
- Analytics Service（学习数据统计）
- Background Worker（Celery）：离线出题、批量嵌入、PDF解析

**关键接口示例**

```
POST /api/v1/auth/login
POST /api/v1/users/{id}/upload（文件上传）
POST /api/v1/ai/generate-plan
POST /api/v1/quiz/generate
POST /api/v1/quiz/submit
GET  /api/v1/analytics/progress
POST /api/v1/ai/chat
```

---

### 3. AI 模块设计（RAG + Generative）

- **上传解析**：用户上传教材 → 后端解析成文本切片 → 生成 Embeddings（向量） → 存入向量库
- **检索（Retrieval）**：用户提问或生成题目时，先在向量库检索 top-k 相关片段
- **生成（Generative）**：将检索到的上下文与用户指令拼接，调用 GPT（或自有模型）生成计划/题目/解析
- **后处理**：对生成结果做格式化、敏感词过滤、难度标签化并存储

**RAG 流程** 1. 请求接入 → 2. 语义检索 top-k → 3. 构造 prompt（含检索片段）→ 4. 调用 LLM → 5. 返回并保存

---

## 六、数据库设计（核心表）

### users

- id, email, hashed_password, name, role, created_at

### profiles

- user_id, learning_level, preferred_subjects, timezone

### documents

- id, user_id, filename, s3_path, parsed_text, status

### embeddings

- id, doc_id, chunk_index, vector (向量ID 关联), text_snippet

### study_plans

- id, user_id, plan_json, generated_at, last_updated

### quizzes

- id, creator_id(auto/AI), quiz_json, difficulty, created_at

### attempts

- id, user_id, quiz_id, answers_json, score, submitted_at

### analytics

- id, user_id, metric_type, metric_value, timestamp

---

## 七、接口细节与示例（伪码）

### 生成学习计划（后端流程）

```
POST /api/v1/ai/generate-plan { user_id, goals, preferred_time }
- 检查权限
- 调用用户资料获取学习历史
- 构造 prompt（包含目标、历史、偏好）
- 调用 LLM -> 返回 plan
- 将 plan 存入 study_plans，返回 plan_id
```

### 生成测验（出题）

```
POST /api/v1/quiz/generate { user_id, subject, num_questions, difficulty }
- 检索知识库（RAG）得到相关片段
- 调用 LLM 生成题目 + 答案 + 解析
- 将题目入库（quizzes）或直接返回给前端
```

---

## 八、部署架构（示例：Docker Compose）

- 服务：frontend (静态) / api / worker / redis / postgres / vectorstore / minio
- Nginx 反向代理，SSL 终端
- CI/CD：GitHub Actions -> Docker镜像 -> 服务器拉取部署

```
server
 ├── Nginx
```

```
├── docker-compose (api, worker, redis, postgres, minio, vectorstore)
└── certbot
```

---

## 九、安全与合规

- **身份认证**：JWT + Refresh Token，接口权限校验
- **API 限流**：对 AI 接口做 IP 或用户级别限流，防止滥用与成本爆炸
- **数据隐私**：上传文件加密存储，敏感信息在客户端遮蔽/脱敏
- **审计日志**：用户操作与 AI 请求日志化，便于复现与审查

---

## 十、演示用例与答辩亮点（建议）

1. **现场演示 5 分钟流程**：上传 PDF -> 生成学习计划 -> 直接做 AI 出题 -> 提交并展示 AI 解析 + 仪表盘数据
2. **对比展示**：展示系统生成的“个性化题目”与随机题目的差异（难度/覆盖率）
3. **性能与成本控制**：说明 API 调用计费控制（缓存 + 限流 + 离线生成）
4. **拓展性**：展示如何接入学校 LMS 或导出成绩报告

---

## 十一、后续可交付物（我可继续产出）

- 详细系统时序图（用户上传 -> RAG -> 生成 -> 存储）
- 数据库 ER 图 + SQL 建表脚本
- 前端页面原型（Figma 或 React 组件树）
- 后端关键代码示例（FastAPI 路由、Celery 任务、向量构建脚本）
- 部署脚本（Docker Compose / Nginx 配置）

---

## 十二、里程碑（建议 4 周 MVP 迭代计划）

- Week1：需求细化 + 架构搭建（前后端脚手架）
- Week2：文件上传/解析 + 向量化流程 + 简单 RAG 问答
- Week3：出题/测评模块 + 自动批改 + 仪表盘
- Week4：UI 美化 + 部署 + 演示材料（视频、说明文档）

---

如果你希望，我可以接着： - 生成**时序图**和**ER 图**（SQL 建表脚本），或 - 直接**生成前端项目模板**（React + Tailwind）并给出部署步骤。

请选择你的下一步，我会把相应内容放到画布里继续完善。