好的，遵照你的要求，我已将之前对项目的分析和理解，整理成一份详细的正式文档。

在线模拟考试平台 - 项目文档

1. 项目概述

本项目是一个基于 Node.js 和 React 的全栈在线模拟考试平台。它采用前后端分离的现代化架构，为用户提供从注册登录、选择试卷、进行模拟考试到查看详细结果的全流程体验。

该平台最大的特色是拥有一套功能强大的**后台管理 API**，允许管理员对考试局、学科、试卷、题目、知识点以及题目与知识点的关联（打标签）等核心数据进行精细化的增删改查（CRUD）操作。

1.1. 核心功能

- **用户端**: 用户注册、登录、浏览试卷列表、参加计时考试、提交答卷、查看带有逐题解析的考试结果。
- **管理端 (通过 API)**: 对平台所有核心数据进行全面的后台管理，包括创建复杂的带有刺激材料（stimulus）和特殊布局的题目。

1.2. 技术栈

- **前端**: React (Vite)
- **后端**: Node.js, Express.js
- **数据库**: PostgreSQL
- **ORM**: Prisma
- **认证**: JWT (JSON Web Tokens)

---

2. 数据库设计 (Prisma Schema)

数据模型是整个应用的骨架，定义在 `backend/prisma/schema.prisma` 文件中。

模型 (Model)	描述	关键字段
`User`	存储用户信息。	`id`, `email`, `passwordHash`, `name`
`ExamBoard`	考试局，如 "AP", "A-Level"。	`id`, `name` (唯一)
`Subject`	学科，如 "AP Computer Science A"，关联到一个 `ExamBoard`。	`id`, `name` (唯一), `examBoardId`
`Exam`	试卷，包含标题和时长，关联到一个 `Subject`。	`id`, `title`, `durationMinutes`, `subjectId`
`Question`	题目，项目的核心。一个题目必须属于一个试卷。	`id`, `examId`, `questionText`, `questionType` (枚举), `order` (题号), `stimulusText`, `layoutType`
`MultipleChoiceOption`	选择题的选项，关联到一个 `Question`。设置了**级联删除**，删除问题时会自动删除其所有选项。	`id`, `questionId`, `text`, `isCorrect`
`KnowledgePoint`	知识点，如 "Java Loops", "Recursion"。可与 `Question` 建立多对多关系。	`id`, `name` (唯一), `subjectId`
`ExamSubmission`	用户的答卷提交记录，关联到 `User` 和 `Exam`。	`id`, `userId`, `examId`, `score`, `submittedAt`
`UserAnswer`	用户对单个问题的具体答案，关联到 `ExamSubmission` 和 `Question`。	`id`, `submissionId`, `questionId`, `answerContent`

---

3. API 接口文档

API 设计遵循 RESTful 规范，分为对外的公共 API 和内部使用的管理员 API。所有 API 的基础路径为 `http://localhost:3001/api`。

3.1. 公共 API

3.1.1. 认证接口 (`/auth`)

- `POST /register`: 注册新用户。
  - **Body**: `{ "name": "Alice", "email": "alice@example.com", "password": "securepassword123" }`
  - **成功响应**: `201 Created`，返回新用户信息（不含密码）。
- `POST /login`: 用户登录。
  - **Body**: `{ "email": "alice@example.com", "password": "securepassword123" }`
  - **成功响应**: `200 OK`，返回 `message`, `token` (JWT), 和 `user` 对象。

3.1.2. 试卷接口 (`/exams`)

- `GET /`: 获取所有可用的试卷列表。
- `GET /:examId`: 获取单个试卷的完整信息，包括其下所有问题及选项。
- `POST /`: 创建一份新试卷。
  - **Body**: `{ "title": "...", "durationMinutes": 90, "subjectId": 1 }`

3.1.3. 提交接口 (`/submissions`)

- `POST /`: 提交用户答卷。
  - **Body**: `{ "examId": "...", "answers": { "questionId_1": "optionId_A", ... } }`
  - **成功响应**: `201 Created`，返回包含分数和逐题答案详情的完整提交记录。

3.2. 管理员 API

3.2.1. 试卷管理 (`/admin/exams`)

- `GET /`: 获取所有试卷的列表（管理视图）。
- `POST /`: 创建一份新试卷。
- `PUT /:id`: 更新指定 ID 的试卷信息。
- `DELETE /:id`: 删除指定 ID 的试卷。

3.2.2. 题目管理 (`/admin/exams/:examId/questions`)

- `GET /`: 获取指定试卷下的所有题目。
- `POST /`: 为指定试卷添加一道新题目。
  - **Body**: 包含 `order`, `questionText`, `questionType`, `options` (如果是选择题), `layoutType`, `stimulusText` 等字段。
- `PUT /:id`: 更新指定 ID 的题目信息。
- `DELETE /:id`: 删除指定 ID 的题目。

3.2.3. 知识点管理 (`/admin/knowledge-points`)

- `POST /`: 创建一个新的知识点。
  - **Body**: `{ "name": "Java Loops", "subjectId": 1 }`
- `DELETE /:id`: 根据 ID 删除一个知识点。

3.2.4. 问题标签管理 (`/admin/question-tags`)

- `POST /add`: 为一个问题打上知识点标签（建立关联）。
  - **Body**: `{ "questionId": "...", "newPointId": ... }`
- `DELETE /remove`: 从一个问题移除知识点标签（解除关联）。
  - **Body**: `{ "questionId": "...", "pointIdToRemove": ... }`

---

4. 前端功能模块

前端使用 React 和 Vite 构建，实现了动态、响应式的用户界面。

- `App.jsx`: 应用的根组件，使用 `react-router-dom` 设置所有页面路由，并用 `AuthProvider` 全局包裹，实现全局状态管理。
- `context/AuthContext.jsx`: **全局认证核心**。
  - 使用 React Context API 创建了一个全局共享的 state，包含 `user`, `token`, `login`, `logout`, `isLoading`。
  - 在应用加载时，它会检查 `localStorage` 中是否存在 `authToken`，如果存在则自动解码 `jwt-decode` 并设置用户登录状态，提供了持久化登录体验。
  - 提供了 `login` 和 `logout` 方法供其他组件调用，以更新全局用户状态和 `localStorage`。
- `pages/HomePage.jsx`: 应用主页。
  - 从 `/api/exams` 获取并展示所有试卷列表。
  - 根据 `AuthContext` 中的 `user` 状态，动态显示右上角的登录/注册链接或欢迎信息和退出登录按钮。
- `pages/LoginPage.jsx` & `pages/RegisterPage.jsx`: 登录和注册页面。
  - 提供完整的表单交互，包括输入处理、加载状态（按钮禁用）和错误信息展示。

    - 登录成功后，调用 `AuthContext` 的 `login` 方法，并将用户重定向到主页。
- **`pages/ExamPage.jsx`**: **最复杂的页面**，实现了完整的考试体验。
    - **动态布局**: 能根据题目数据中的 `layoutType` 渲染**堆叠式**或**左右分栏式**的布局。
    - **问题导航器 (`components/QuestionNavigator.jsx`)**: 用户可以随时打开导航器，查看所有题目的完成状态（已答/未答）、标记状态和当前题目位置，并能点击任意题号进行快速跳转。
    - **答题与标记**: 用户可以为每个问题选择答案，并随时**标记/取消标记**以便后续检查。
    - **安全提交**: 在交卷时，若检测到有题目未完成，会弹出 `confirm` 对话框进行二次确认。
- **`pages/ResultPage.jsx`**: 考试结果展示页。
    - 从路由状态 (`location.state`) 中获取交卷后的 `submission` 数据。
    - 清晰地展示总分，并用列表形式逐一展示每道题的题干、用户的答案和正确答案，并用不同颜色和图标 (✔/✖) 标出错对。
- **`services/api.js`**: API 服务层。
    - 将所有对后端 API 的 `axios` 调用封装成独立的函数，如 `fetchAllExams`, `loginUser` 等，使组件代码更简洁，职责更清晰。

---

5. 安装与启动

请遵循以下步骤在本地运行此项目。

5.1. 后端 (Backend)

1. **导航到后端目录**:
   ```
   cd backend
   ```
2. **安装依赖**:
   ```
   npm install
   ```
3. **设置环境变量**:
    - 复制 `.env.example` (如果存在) 为 `.env`。
    - 在 `.env` 文件中配置 `DATABASE_URL`，指向你的 PostgreSQL 数据库。
      ```
      DATABASE_URL="postgresql://USER:PASSWORD@HOST:PORT/DATABASE?schema=public"
      ```
4. **运行数据库迁移**:
   ```
   npx prisma migrate dev
   ```
5. **启动后端服务器**:
   ```
   npm start
   ```
   服务器将在 `http://localhost:3001` 上运行。

5.2. 前端 (Frontend)

1. **导航到前端目录**:
   ```
   cd frontend
   ```
2. **安装依赖**:
   ```
   npm install
   ```
3. **启动开发服务器**:
   ```
   npm run dev
   ```
   前端应用将在 Vite 开发服务器上启动，通常是 `http://localhost:5173`。
