基于全栈架构的 AP Biology 智能学习平台设计与实现

王致远

摘要

Biochem-niche 是一款面向 AP Biology 学习者的线上智能学习平台，以“生态龛”（niche）理念为设计哲学，旨在为每一个生物学习者提供适切其个人需求的学习环境。本平台基于 Next.js 14（App Router）+ React + TypeScript + Tailwind CSS 构建前端，以 Prisma ORM + PostgreSQL（Supabase）作为数据层，通过 Vercel 实现全球部署。经历了从 v1.0 纯静态网站到 v2.1 全栈动态应用的迭代演进，当前版本已实现知识库浏览、在线小测、专业词典、真题模考等核心功能，并支持用户注册登录、角色权限管理、国际化双语支持等扩展功能。平台已录入 AP Biology 2024 Full Exam, Asia 卷共 66 道真题（60 道 MCQ + 6 道 FRQ），前端复刻了 Bluebook 考试界面风格，支持自动判分、记录存档、PPT/PDF 导出等功能。本项目是北京师范大学附属中学首个系统化的 AP 生物学习平台，也是校内 PBL（Project-Based Learning）项目式学习的成果转化典范。

关键词：AI 赋能；后台技术；教育平台

1 引言

1.1 问题提出

AP Biology 是科学类 AP 课程中内容体系最庞大、备考难度最高的科目之一。然而，其学习资源的获取在校内外均存在显著不对称：部分学校拥有系统的辅导资料与模拟训练，而更多学生只能依赖零散的网络资源进行自学。这种差异不仅降低了备考效率，也在一定程度上加剧了教育不公平。

对于国内 AP 学生而言，备考困境更为具体。首先，官方资料分散在不同平台，缺乏系统性整合与联动。其次，线上资源质量参差不齐，知识点错误、超纲或过时内容难以辨别。再次，主流资源均为纯英文，对英文基础较弱的学生构成额外理解障碍。因此，构建一个结构化、双语支持、可达性强的 AP Biology 在线学习平台，具有明确的现实需求。

1.2 项目定位

Biochem-niche 正是基于上述需求而开发的 AP Biology 智能学习平台。本项目源于开发者作为 AP Biology 学生的真实学习体验，是校内 PBL（Project-Based Learning）项目式学习的实践成果。平台以"为每一个生物学习者提供适切的学习环境"为设计理念，将分散的学习资源整合为结构化的整体解决方案，面向本校 AP 生物学社区开放使用。

项目开源地址：https://github.com/Bio-innovator/biochem-V2 **[7]**

2 需求分析

2.1 用户画像

本平台的目标用户主要分为两类：学生用户与教师用户。学生用户是平台的核心使用群体，主要为本校正在修读或准备修读 AP Biology 的高中生，年龄范围通常在 15-18 岁之间。这一群体具有以下特征：科学素养较好但需要结构化的知识框架；英文阅读能力参差不齐，双语支持能够有效降低学习门槛；普遍使用移动设备进行线上学习，对响应式设计要求较高；需要即时的练习反馈与错题归档功能来追踪学习进度。

教师用户主要为本校 AP Biology 授课老师，其核心需求包括查看班级统计数据、监控学生学习进度等。管理员权限由项目开发者本人持有，负责系统维护、内容审核与用户管理。当前版本中，学生、教师与管理员三种角色已通过数据库级别的权限控制实现。

2.2 功能需求

结合用户调研与本人作为 AP Biology 学生的亲身经验，平台的功能需求可分为以下几个模块。知识库模块需要按照 AP Biology CED 2025 的 8 个 Unit、55 个 Topic 组织，支持每个 Topic 的概述、关键概念与拓展问题。小测练习模块需要支持按 Unit 筛选题目、自定义题量（10/20/40 题）、自动判分与错题归档。真题模考模块需要完整复制 AP 官方考试的界面与流程，包括计时器、Mark for Review 标记、Section 切换等功能。词典模块需要提供按单元和字母梳选的双维检索。**[1] [2]**

2.3 非功能需求

非功能需求方面，首先是双语支持。平台所有用户面板文本均以中文为主、英文为辅，保证国内学生的可读性同时不失去对 AP 专业术语的接触。其次是响应式设计，平台必须在桌面端和移动端均有良好的浏览体验。性能方面，首屏加载时间应控制在 3 秒以内，页面跳转采用客户端渲染以减少白屏时间。可用性方面，系统可用率目标为 99.9%，数据库自动备份，支持快速恢复。

3 系统架构与技术选型

3.1 技术栈总览

Biochem-niche v2.1 采用了当今前端开发领域主流的全栈方案。前端基于 Next.js 14 的 App Router 模式，搭配 React 18 + TypeScript + Tailwind CSS。后端未采用独立服务器框架，而是利用 Next.js 内建的 API Routes 机制，将前后端代码统一在同一个项目中。数据库采用 PostgreSQL，通过 Supabase 托管，ORM 使用 Prisma。部署平台选择 Vercel，图片存储使用 Supabase Storage。**[3] [4] [5] [6]**

3.2 选型决策分析

选择 Next.js 14 作为前端框架，基于以下几点考量。首先，App Router 引入了基于文件系统的路由机制，与本项目按 Unit/Topic 组织内容的方式高度契合，页面结构清晰可见。其次，Next.js 的服务端渲染（SSR）和静态网站生成（SSG）能力可以显著提升首屏加载速度与 SEO 表现，对于内容型平台至关重要。再次，Next.js 内建 API Routes，让前后端共存于同一个项目，减少了多个服务之间的协调成本，非常适合个人开发者独立完成的项目。**[3] [4] [5]**

选择 Supabase 作为 PostgreSQL 托管方案，主要因为其免费层提供了充足的数据库容量与连接数，并且内建了认证系统、Storage 存储和实时订阅等扩展功能。Prisma ORM 的加入则是因为其类型安全、自动迁移管理和优秀的开发者体验。相比于原生 SQL，Prisma 的 Schema 定义方式让数据库结构一目了然，且能够在编译阶段发现类型错误。

3.3 v1.2 静态架构的局限性

在 v2.0 全栈重构之前，项目经历了 v1.0 到 v1.2 的静态网站阶段。v1.0 使用纯 HTML/CSS/JavaScript 手动编写，托管于 GitHub Pages。这一版本虽然快速验证了需求，但存在严重的技术债务：HTML 实体编码导致中文乱码；生物与化学数据混杂在同一个站点中，缺乏专业聚焦；课程结构未按 AP Biology CED 标准组织；更严重的是，部分内容生成依赖 AI，导致出现了“SAT”等与 AP Biology 无关的幻觉文本。

v1.1 与 v1.2 依次修复了上述问题：v1.1 删除了所有化学内容，按 AP Biology CED 2025 重构了 8 个 Unit（55 个 Topic），修复了乱码，并将词典重构为按单元与字母梳选的双维检索。v1.2 进一步修复了测试题数量 Bug（选择 10/20 题实际只出 5 题），并增加了一个基于 localStorage 的模拟登录系统。

然而，v1.2 的架构局限性仍然显著。第一，GitHub Pages 作为静态托管平台，无法支持服务端逻辑，所有数据处理都必须在客户端完成。第二，localStorage 存储的数据仅保存在本地浏览器中，换设备后完全丢失，且清除浏览器缓存即可导致所有学习记录消失。第三，模拟登录系统没有真正的用户体系与权限验证，管理员审核功能名存实亡。这些结构性缺陷，最终推动了 v2.0 的全栈重构决策。

3.4 v2.0 全栈重构的核心改进

v2.0 以"新建仓库、完全重写"的方式启动。新仓库名为 biochem-V2，技术栈践行 Next.js 14 + Prisma + Supabase + Vercel 的组合。核心配置文件（Prisma schema、JWT 认证中间件、API Routes 骨架）由 Kimi Agent 基于结构化提示词生成，开发者负责架构审核、字段校验与部署调试。这一版本实现了以下核心突破：数据库从浏览器 localStorage 迁移至 PostgreSQL，实现了持久化存储；用户系统从模拟登录升级为基于 JWT 的真实认证系统，支持注册、登录、登出与密码修改；角色权限体系分为 STUDENT、TEACHER、ADMIN 三级，通过 Prisma 中间件检查 role 字段实现接口级别权限控制；前后端通过 Prisma Client 交互，所有数据操作均在服务端完成。

在迁移过程中，一个典型的技术问题是 Supabase 连接配置。本地开发环境使用 IPv4，而 Supabase 的 Direct Connection 默认使用 IPv6，导致连接超时失败。最终解决方案是切换至 Session Pooler 模式，并在连接字符串中启用 pgbouncer=true 参数，这一经验也被记录在项目的技术文档中。

4 核心功能实现

4.1 项目启动与静态网站构建

作为个人开发者独立完成的课程项目，项目初期面临的首要约束是零成本快速验证。在对比了多种托管方案后，选择 GitHub Pages 作为静态托管平台——免费、稳定、

与代码仓库天然集成。域名注册选择了 Dynadot（排除了需要 Veriff 人脸/身份证验证的 porkbun，出于隐私安全考虑），最终确定 biochem-niche.com 作为主域名。

v1.0 的代码并非从零手写，而是基于 Kimi Agent 根据功能需求提示词生成，开发者在此基础上进行人工审核与路径调整。这种"AI 生成 + 人工校验"的开发模式，在高中生独立开发项目中显著降低了技术门槛。生成后的文件通过 GitHub 网页端直接上传至仓库，开启 GitHub Pages 部署。

部署初期遇到两个典型问题：一是 **DNS 传播延迟**——配置 A 记录指向 GitHub Pages IP 后，需要 10-20 分钟等待全球 DNS 缓存刷新，期间反复点击"Check Again"验证；二是 **GitHub 路径扁平化**——上传 zip 文件解压后，因连续嵌套文件夹被 GitHub 自动合并，导致文件路径失效，多次排查后才定位到该机制。

完成 DNS 配置、开启 Enforce HTTPS 自动签发 SSL 证书、添加 www 子域名记录后，v1.0 版本正式上线。这一"AI 生成 + 人工校验"的开发模式在后续 v2.0 与 v2.1 的迭代中延续使用：开发者负责需求拆解、架构设计与提示词工程，Kimi Agent 负责代码生成与 SQL 编写，开发者再进行审核、调试与部署。

4.2 静态迭代

v1.0 上线后，基于初步使用反馈与开发者自身作为 AP Biology 学习者的体验，项目进入快速迭代期。v1.1 的核心任务是"做减法"：首先，彻底删除所有化学内容，导航栏移除"化学"入口，品牌聚焦为 BioNiche，以解决 v1.0 双学科定位导致的内容深度不足与专业聚焦缺失问题。其次，按照 AP Biology CED 2025 标准重构知识结构，将知识库重组为 8 个 Unit 共 55 个 Topic，测验题精简为 40 道纯生物题目，词汇表调整为 190 个词条，均按 Unit 分类，使平台内容与校内 AP Biology 课程严格对应。再次，修复了 v1.0 遗留的全部 HTML 实体编码乱码问题，导航、标题、描述与按钮文本实现正常渲染，特殊符号正确显示。最后，词典模块重构为"顶部搜索框 + A-Z 字母栏 + 左侧单元列表 + 右侧单词详情"的双维检索布局，每个单元内包含单词、音标、释义与例句，并移除了干扰学习节奏的"每日推荐"功能。

然而 v1.1 的修改并不彻底，多项细节问题持续暴露。首先是内容残留：网站介绍文案中仍包含"化学"字样及破折号后的多余内容，页脚小字中"SAT"字样未完全清除，网站名称在迭代过程中被意外修改，需手动恢复为"Biochem-niche"。这些文本错误看

似微小，实则反映了静态网站缺乏服务端渲染模板机制的深层局限——每一处文本修改都需要手动定位 HTML 文件，极易遗漏。其次是功能缺陷：词典界面布局仍不够合理，原整体深色背景影响可读性，亟需改为白色并配套调整文字、卡片等配色；测试题数量存在严重 Bug，用户选择 10 题或 20 题后实际固定生成 5 题，根源在于题目生成逻辑未正确传递和使用题数参数。更关键的是，v1.1 仍未解决用户账户系统的缺失，网站没有注册、登录、管理员审核与退出机制，用户管理与学习进度保存无从谈起。

v1.2 针对上述遗留问题进行了系统性修复。化学内容被彻底删除，网站名称恢复为 Biochem-niche，底部"SAT"字样完全移除。词典界面完成重构，采用顶部搜索框、A-Z 字母栏、左侧单元列表与右侧单词详情的布局。全局背景色改为白色，文字、卡片、边框同步适配为深色文字与浅色卡片，确保白底可读性与美观度。测试题数量 Bug 被修复，实际生成题目严格对应所选数量。此外，v1.2 增加了一个基于 localStorage 的模拟登录系统，用户可选择学生、教师或管理员身份进入平台，登录状态显示在导航栏右上角，支持查看用户名与退出操作。

但 v1.2 的优化也更为深刻地暴露了静态架构的系统性瓶颈。第一，服务端逻辑缺失。GitHub Pages 仅支持静态文件托管，无法执行后端代码，导致管理员审核功能完全名存实亡——注册请求无法发送至任何服务端，管理员无法接收、查看或批准待审核账户，审核流程沦为前端逻辑的自我欺骗。第二，教师数据隔离。教师与管理员账号无法查看学生的实际学习数据，教学反馈与学习监督功能无法实现。第三，数据持久性与安全性危机。所有学习进度、答题记录与用户信息存储于浏览器 localStorage，清除缓存或更换设备即全部丢失；且 localStorage 无加密保护，数据可被直接读取或篡改，存在显著的隐私与安全风险。第四，多端数据无法同步。同一用户在不同设备登录时，学习数据完全隔离，无法跨设备同步进度。这些结构性缺陷表明，静态网站已触及能力边界，任何进一步的功能扩展——如真实用户体系、跨设备同步、数据聚合分析——都必须依赖全栈架构。这直接催生了 v2.0 的重构决策。

4.3 全栈重构与架构迁移

v1.2 的静态架构已触及能力边界，任何涉及服务端逻辑、跨设备数据同步或用户权限管理的功能都无法在 GitHub Pages 上实现。因此，v2.0 采用"新建仓库、完全重写

"的策略，将技术栈从纯静态 HTML/CSS/JavaScript 迁移至 Next.js 14 + Prisma ORM + PostgreSQL（Supabase）+ Vercel 的全栈架构。

平台注册与项目初始化是重构的第一步。首先在 GitHub 新建仓库 biochem-V2，用于隔离历史代码与技术债务。随后在 Vercel 使用 GitHub 账号完成注册，部署配置中选择 Next.js 作为 Framework Preset，并添加两个核心环境变量：DATABASE_URL（Supabase 连接字符串）与 JWT_SECRET（认证密钥）。数据库托管选择 Supabase，在其控制台建立位于首尔节点的 PostgreSQL 实例——Supabase 在当时未提供中国大陆节点，首尔是网络延迟与合规性之间的平衡选择。

项目骨架搭建延续了 v1.0 的"AI 生成 + 人工校验"模式。核心配置文件（Prisma schema、JWT 认证中间件、API Routes 骨架）由 Kimi Agent 基于结构化提示词生成，开发者负责架构审核、字段校验与部署调试。数据库表结构通过 Supabase SQL Editor 直接执行建表语句完成，v1.2 的静态数据（55 个 Topic、40 道测验题、190 个词汇词条）通过重新录入的方式迁移至 PostgreSQL，同时利用 Kimi Agent 将 v1.2 的前端页面逻辑适配为 Next.js 组件，生成新文件后逐一上传至 GitHub 仓库。

然而，v2.0 的部署过程遭遇了远比预期复杂的技术阻力，主要体现为三个层面的问题。

第一，AI 协作路线冲突。 在重构初期，Kimi Agent 始终建议开发者在本地计算机创建 Next.js 项目并逐行执行 npm 命令，这与开发者"不跑本地代码、直接通过云端构建"的路线产生冲突。开发者最终强行跳过本地初始化步骤，直接要求 AI 生成完整文件后通过 GitHub 网页端上传，由 Vercel 在线完成构建。这一经历表明，在与 AI 协作进行工程开发时，开发者必须保持独立的技术判断，AI 的推荐路径存在基于通用场景的幻觉，未必适配特定的部署约束。

第二，GitHub 文件上传与路径解析机制。 开发者在上传文件时遇到持续性的路径失效问题：AI 提供的代码以 zip 压缩包形式下载，解压后的文件夹结构在视觉上与正常目录无异，但直接上传后页面全部 404。经反复排查发现，GitHub 存在"文件夹扁平化合并"机制——当 zip 文件解压后出现连续嵌套且中间层级无其他同级文件时，GitHub 会自动跳过中间文件夹，导致文件路径与代码中的 import 引用完全不匹配。解决方案是放弃 zip 上传，改为直接拖拽解压后的独立文件，并严格控制目录层级，避免无意义的连续嵌套。

第三，网络协议与数据库连接配置。 这是 v2.0 阶段最具代表性的工程挑战。项目成功构建后，登录功能与其他 API 调用全部失败。检查 Vercel 运行日志发现，Vercel 的服

务器默认使用 IPv4 网络，而 Supabase 的 Direct Connection 默认走 IPv6 地址，两者协议不匹配导致连接超时。开发者首先在 Supabase 控制台将连接方式切换为 Session Pooler，并在连接字符串中追加 pgbouncer=true 参数，同时在 Vercel 环境变量中更新数据库 URL。然而更新后问题仍未解决，日志持续报密码错误，开发者与 Kimi Agent 就连接字符串格式与密码正确性进行多轮排查均未定位根因。随后切换至 Qwen 3.7 协助排查，先定位到动态渲染的代码问题，但再次陷入类似的连接配置拉扯。最终切换至 Gemini 时，登录功能突然恢复正常。事后推测，环境变量在 Vercel 控制台更新后并非即时生效，需要等待部署流水线重新构建完成，而此前的反复调试恰好覆盖了部署延迟周期。这一经历使开发者首次深入理解了 Serverless 平台的异步部署机制与数据库连接池的配置逻辑。

域名切换与资产迁移在架构稳定后完成。开发者在 Vercel 控制台添加现有域名 biochem-niche.com，随后返回 Dynadot 删除此前指向 GitHub Pages 的 A 记录，替换为 Vercel 提供的 DNS 记录，www 子域名同步更新。至此，域名解析从静态托管平滑迁移至 Serverless 平台。

v2.0 的成功上线标志着项目从静态展示页进化为具备真实用户体系、持久化存储与角色权限控制的全栈应用。数据库从浏览器 localStorage 迁移至 PostgreSQL，用户系统升级为基于 JWT 的真实认证，支持注册、登录、登出与密码修改，角色权限分为 STUDENT、TEACHER、ADMIN 三级并通过 Prisma 中间件实现接口级控制。这一架构升级为后续 v2.1 的功能扩展奠定了不可替代的技术基础。

4.4 功能扩展

v2.0 的全栈架构验证通过后，项目进入以功能深化与用户体验优化为核心的迭代阶段。v2.1 的工作重点并非单一功能的堆砌，而是将平台从"能跑"推进到"好用"——这包括前端交互重构、双语适配、真题模考系统搭建，以及更为关键的全链路工程问题修复。与 v2.0 的"架构迁移"不同，v2.1 的迭代逻辑是"在真实用户场景中持续打磨"，每一项功能都需要经过数据流全链路的验证。

4.4.1 前端界面重构与用户体验优化

v2.0 上线后，首页仍沿用早期静态版本的简单布局，缺乏品牌辨识度与功能引导。v2.1 首先对信息架构进行了重构。

导航栏调整是用户体验优化的起点。我们在 components/Navbar.tsx 中增加了"故事"（/story）与"介绍"（/about）两个入口，移除了冗余的"注册"按钮——因为 v2.0 的真实用户体系已支持登录后自动跳转，独立注册入口反而增加了认知负担。同时，将平台功能页的底部入口名称从"教师控制台"和"管理后台"调整为更贴近学生认知的"班级统计"和"AP 题目测试"，使功能定位更直观。

首页采用滑动式全屏设计，划分为 Hero 屏、Features 屏、Units 屏与 Footer 屏四个独立页面。这一改动通过可复用的 FullPageScroll 组件实现，支持鼠标滚轮切换、键盘方向键导航与侧面圆点指示器，并设置了 200ms 的滚动防抖，防止连跳。Hero 屏突出品牌标识与登录入口；Features 屏以六宫格卡片展示平台六大功能模块；Units 屏按 AP Biology CED 标准呈现 8 个单元；Footer 屏再次强化行动号召。

内容页面统一也是 v2.1 的细节重点。Story 页面按"发音→误区→命名→使命"四段叙事，About 页面按"开发者→项目来源→服务范围→指导老师"五段展开。在调整过程中，学校页面的文字样式经历了多次迭代——最初直接平铺的表述与整体设计风格不统一，最终统一为"emoji + 粗体标题 + 青色副标题 + 小字解释"的双层信息结构，确保视觉层级与 Story、About 页面保持一致。

4.4.2 国际化与双语支持

v2.1 对首页、Story 页和 About 页进行了全面的双语改造。考虑到平台主要面向本校 AP Biology 学生，双语策略定为"中文为主、英文为副标题或说明文字"——这既保证了国内学生的可读性，又让学生始终接触 AP Biology 的专业英文术语。

具体实现上，首页的"AP Biology 智能学习平台"下方增加了"AP Biology Intelligent Learning Platform"小字；功能卡片的标题下方增加英文标签（如 Knowledge / Quiz / Glossary）；登录弹窗的标签采用"用户名 Username"内联双语格式，placeholder 也改为双语提示。Story 页面的四段中文叙事下方均配有英文翻译，About 页面的学校介绍与项目介绍同样采用中英对照。这种设计不是简单的翻译堆砌，而是将英文作为"状态提示"嵌入中文语境，降低阅读障碍的同时维持专业感。

4.4.3 真题模考系统：从数据层到表现层的完整闭环

真题模考系统是 v2.1 最具代表性的功能扩展，但搭建过程远比预期复杂，大量精力消耗在数据链路打通与前端状态修复上。

数据导入是首个工程挑战。我们完整录入了 AP Biology 2024 Full Exam, Asia 卷共 66 题（60 道 MCQ + 6 道 FRQ）。66 道题目的 INSERT 语句由 AI 基于标准化提示词生成，开发者负责审核答案正确性、字段映射与语法校验。导入过程中发现第 30 题与第 31 题题干完全重复但答案不同，经核对原始 PDF 后，将第 30 题替换为真实的"larval stage 蛋白质表达"题目，并修正单元归属为 Unit 6。

图片存储针对真题中的图表与实验数据需求而设计。我们在 Supabase Storage 中创建公开 bucket exam-images，将题目截图上传后通过公开 URL 访问。考虑到单道题可能包含多张图片（如实验组图或比较图表），imageUrl 字段采用逗号分隔的多 URL 存储方案，前端通过 split(',') 解析并循环渲染。图片上传后发现数据库 ExamQuestion 表缺少 explanation、imageUrl、type、unit 列，需通过 ALTER TABLE 动态补全；同时 answer 列的 NOT NULL 约束与 FRQ 的 NULL 值冲突，需执行 DROP NOT NULL 才能正常导入。

前端界面复刻 Bluebook 风格，包含顶部 Section II 标识栏、计时器、Hide/Show 切换、Directions 下拉说明、Mark for Review 标记以及底部题号下拉选择器。FRQ 部分采用务实策略：在线显示题目文本，学生在纸笔作答后提交查看 Scoring Guidelines 进行自我评估，避免了不成熟的自动判分带来的误导。

系统性的前端 Bug 修复消耗了大量时间。初始部署后，FRQ 题目被前端过滤逻辑隐藏，做完第 60 题后无法进入第 61 题，原因是考试页面错误地过滤了 questions 数组。修复后，未作答直接提交却被判定为 100 分——根因是 API 未返回 answer 字段，导致 undefined === undefined 被误判为全部正确。修复方案包括：API 层确保返回完整题目数据（含答案），前端判分逻辑增加 q.answer !== null && q.answer !== undefined 的防护判断。

4.4.4 管理功能扩展

v2.1 在管理员后台增加了"清除学生数据"功能。该功能允许管理员清除指定学生的 QuizResult、ExamResult、ErrorBook 与 KnowledgeProgress 记录，但保留 User 账号本身。实现上基于 Prisma 事务（$transaction）确保所有删除操作的原子性——要么全

部成功，要么全部回滚，避免数据不一致风险。前端在学生列表的操作列增加条件渲染按钮，仅对 role === 'STUDENT' 的账户显示，防止误操作教师或管理员数据。

4.4.5 持续部署中的工程问题与修复

v2.1 的开发过程伴随着密集的前后端联调与部署修复，这些"看不见的工程问题"占据了迭代周期的主要部分，也是全栈架构迁移后持续打磨的真实写照。

数据库连接协议与连接池配置是 v2.1 阶段最具代表性的工程挑战。项目成功构建后，所有 API 调用频繁出现连接超时或 prepared statement "s0" already exists 错误。排查发现，Vercel 的 Serverless 节点默认使用 IPv4 网络，而 Supabase 的 Direct Connection 默认走 IPv6 地址，两者协议不匹配导致连接不稳定。解决方案是将连接方式切换为 Session Pooler，并在连接字符串中追加 pgbouncer=true 参数。然而，环境变量在 Vercel 控制台更新后并非即时生效，需要等待部署流水线重新构建完成，此前的反复调试恰好覆盖了部署延迟周期。这一经历使开发者首次深入理解了 Serverless 平台的异步部署机制与数据库连接池的配置逻辑。

Prisma Schema 与数据库实际结构的同步问题反复出现。v2.1 迭代过程中，数据库通过 ALTER TABLE 新增了多列，但 schema.prisma 未及时同步，导致 Prisma Client 查询时抛出类型错误甚至整个 API 崩溃。典型案例包括：Major 表实际字段为 nameEn/nameZh/level，但 schema 中仍保留旧的 name/category，导致专业页面 API 返回 500 并引发前端白屏；QuizResult 表缺少 unit 和 correctCount 列，学生交卷时直接报错；QuizAnswer 表的数据库字段为 selectedOption/isCorrect，但早期 API 代码沿用了旧字段名 selected/correct，导致批量插入失败。修复方案是逐一对齐 schema 与数据库结构，并在 Supabase SQL Editor 中执行 ALTER TABLE 补全缺失列。

enum 类型缺失导致权限系统崩溃。User 表的 role 和 status 字段在数据库中最初以 TEXT 类型存储，但 schema.prisma 中定义为 enum Role 和 enum Status。Prisma 运行时查找数据库的 public.Role enum 类型时发现不存在，任何涉及 User 表查询的 API（包括 classroom、admin/users、auth）全部返回 500。修复方案是在 Supabase 中手动创建 enum 类型，并通过 ALTER TABLE ... USING 将现有 TEXT 列转换为 enum，同时处理默认值冲突。

API 路由职责混淆与认证函数不一致。开发过程中，app/api/classroom/route.ts 误用了不存在的 verifyAuth 函数（项目中实际只有 verifyToken），且采用了 req as

unknown as ... 这种不安全的类型转换，导致运行时崩溃。app/api/exams/route.ts 曾被写成单条查询（findUnique + params.id），但 /api/exams 访问的是列表页，没有 id 参数，Prisma 查询直接崩溃。这些错误提醒我们，在 Next.js App Router 中，路由边界必须严格区分，认证函数必须全局统一。

前端字段不匹配引发的页面崩溃。app/majors/page.tsx 中访问了 major.courses、major.descriptionZh 等字段，但数据库实际返回的是 description、careers、skills，导致 JavaScript 读取 undefined.slice() 时直接白屏。修复方案是逐一对齐前端解构字段与数据库返回字段，并增加条件渲染（{major.careers && (...)}）避免空值崩溃。

v2.1 的迭代表明，全栈架构的迁移并非终点，而是持续工程打磨的起点。从数据库连接池配置到 Prisma schema 同步，从前端状态管理到 API 路由边界，每一项功能都需要经过数据流全链路的验证。这一阶段的经验也强化了我们对类型安全、错误边界与原子性操作的认识，为后续的真题库扩充与功能迭代奠定了工程基础。

4.4 国际化与 UI 优化

v2.1 对首页、Story 页和 About 页进行了全面的双语改造。所有用户面板文本均以中文为主、英文为副标题或说明文字，这种设计既保证了国内学生的可读性，又让学生始终接触 AP Biology 的英文术语。首页采用了滑动式全屏设计，共 4 个屏幕：Hero 屏、Features 屏、Units 屏与 Footer，每个屏幕都精心设计，给用户留下专业的第一印象。

Story 页面是 v2.1 的一个特色内容。页面以“niche”的发音故事引入——这个单词在英语中有两种发音，分别对应“生态位”与“适当的地位”，恰恰是项目理念的精髓。页面还阐述了教育公平的理念，将技术项目与社会价值相连接。

4.5 管理员功能

v2.1 增加了管理员后台的“清除学生数据”功能。管理员可以在后台点击按钮，清除指定学生的 QuizResult、ExamResult、ErrorBook 和 KnowledgeProgress 记录，但保

留 User 账号本身。这一功能基于 Prisma 事务实现，确保所有删除操作要么全部成功，要么全部回滚，避免了数据不一致的风险。

5 数据库设计

本平台的数据库基于 PostgreSQL，通过 Prisma ORM 进行管理。当前 Schema 共包含 13 张表，覆盖了用户管理、知识库、小测系统、真题模考、词典与专业方向等核心业务模块。以下对核心表结构进行详细说明。

5.1 用户与权限

User 表是整个系统的核心，存储用户的基本信息与角色权限。其中，role 字段使用枚举类型，可取值 STUDENT、TEACHER 或 ADMIN，通过 Prisma 中间件检查实现接口级别的访问控制。密码经过 bcrypt 哈希后存储，不会以明文形式保存。

5.2 小测系统

Quiz 表存储小测题目的基本信息，包括题目文本、选项数组、正确答案、解析与所属单元。QuizResult 表则记录每次小测的结果，包括用户 ID、得分、答题记录与用时。这两张表通过外键关联，支持学生查看历史成绩与错题分析。

5.3 真题模考

真题模考系统是 v2.1 的核心功能，其数据库设计需要支持复杂的考试流程。Exam 表存储试卷的基本信息，包括年份、试卷名称、时长限制与题量。ExamQuestion 表存储每一道题的详细信息，其中 questionNumber 表示题号，options 字段存储选项数组，answer 为正确答案，explanation 为解析文本，imageUrl 以逗号分隔存储多个图片 URL，type 字段为枚举类型（MCQ 或 FRQ），unit 字段表示该题所属的 CED Unit。

ExamResult 表记录每次模考的综合成绩，包括用户 ID、试卷 ID、总分、正确率、用时等。ExamAnswer 表则记录每一道题的答案明细，支持按题回顾。这四张表构成了完整的真题模考数据模型。

5.4 其他核心表

KnowledgeTopic 表存储知识库的 55 个 Topic，每个 Topic 归属于某个 Unit，包含标题、概述、关键概念与扩展问题。Glossary 表存储专业词汇，支持按单元与字母梳选。Major 表存储专业探索方向的信息。ErrorBook 表记录学生的错题，支持按单元筛选与复习。KnowledgeProgress 表记录学生对每个 Topic 的学习进度。

5.5 ERD 逻辑关系

整体数据库设计遵循以下逻辑：User 表位于中心，通过外键与 QuizResult、ExamResult、ErrorBook、KnowledgeProgress 等表关联。Exam 表与 ExamQuestion 表为一对多关系，一套试卷包含多道题目。ExamResult 与 ExamAnswer 表也为一对多关系，一次模考记录包含多个答题明细。KnowledgeTopic 表独立组成知识网络，与用户学习进度通过 KnowledgeProgress 表关联。整体设计具有良好的扩展性，可以方便地支持更多试卷、更多题型的录入。**[4]**

以下是 Prisma schema 中 ExamQuestion 表的定义示例：

```
model ExamQuestion {
  id             Int      @id @default(autoincrement())
  examId         Int
  exam           Exam     @relation(fields: [examId], references: [id])
  questionNumber Int
  question       String
  options        String[] // PostgreSQL array type
  answer         String
  explanation    String
  imageUrl       String?  // comma-separated URLs
  type           QuestionType // MCQ or FRQ
  unit           String
}
```

6 测试与部署

6.1 真题数据导入

真题模考系统的核心是高质量的题目数据。我们完整录入了 AP Biology 2024 Full Exam, Asia 卷共 66 道题目，其中 60 道 MCQ 和 6 道 FRQ。每道题都包含完整的题目

文本、选项（MCQ）、正确答案、解析与所属 Unit。对于含图题目，将图片上传至 Supabase Storage，并在数据库中记录对应的公开 URL。

数据导入采用了批量 SQL 脚本的方式。首先将所有题目数据整理为结构化的 JSON 格式，然后通过 Prisma 的 createMany 方法一次性导入数据库。这种方式既保证了数据一致性，也方便后续的题目更新与维护。

6.2 部署流程

平台部署于 Vercel，利用其与 Next.js 的深度集成实现了自动构建与部署。开发者只需将代码推送至 GitHub 仓库，Vercel 会自动检测变更并触发构建流水线。构建过程中，Prisma Client 会自动生成，数据库连接使用环境变量配置。**[3] [4] [5]**

部署过程中遇到的一个典型问题是数据库连接配置。本地开发环境使用 IPv4 网络，而 Supabase 的 Direct Connection 默认使用 IPv6 地址，导致本地开发时连接超时失败。解决方案是在 Supabase 控制台切换至 Session Pooler 模式，并在连接字符串中添加 pgbouncer=true 参数。这一配置在本地开发、预览环境和生产环境中均已验证通过。

6.3 持续部署与版本管理

项目采用基于 Git 的版本管理工作流。主分支为 main，所有功能开发在独立分支上完成，通过 Pull Request 合并。Vercel 为每个推送自动创建预览链接，方便在合并前进行可视化验证。生产环境的部署则由 main 分支的更新触发，实现了持续部署（CD）的工作流。

7 教育价值与社会意义

7.1 校内首个系统化 AP 生物学习平台

Biochem-niche 是北京师范大学附属中学首个由学生独立开发、面向全校开放的 AP Biology 学习平台。在此之前，校内的 AP Biology 备考主要依赖教师个人提供的资料和学生自主搜集的网络资源，缺乏统一的、与课程结构对应的在线学习工具。本平台的出现填补了这一空白。

平台的意义不仅仅在于提供学习资料。更重要的是，它为学生提供了一个结构化的学习框架——知识库按 CED 标准组织，小测系统支持按单元练习，错题本帮助定位知识盲区，模考系统让学生在真实考试环境中检验自己。这种系统性的学习路径，对于自学能力较强但缺乏方向感的学生特别有帮助。**[1]**

7.2 PBL 从课堂到产品的转化

本项目是 AP Biology 课程 PBL（Project-Based Learning）项目式学习的成果。PBL 的核心理念是让学生通过解决真实问题来学习知识和技能，而不是被动地接受知识灌输。Biochem-niche 正是这一理念的典范——它源于开发者作为 AP Biology 学生的真实学习困境，通过迭代开发最终成为一个被全校师生使用的学习工具。

这种“从课堂到产品”的转化具有重要的教育意义。它证明了学生不仅可以是知识的消费者，也可以是知识工具的创造者。对于开发者本人而言，这次经历涵盖了前端开发、后端架构、数据库设计、用户体验、项目管理等多个维度，是一次完整的产品级开发实践。

7.3 降低资源门槛

平台的最终目标是降低 AP Biology 的学习门槛。无论是资源充足的学校还是资源相对薄弱的学校，学生都可以通过一个统一的入口获取结构化的学习资源。平台完全免费开放，没有付费墙或会员限制，这一点与许多商业化的教育应用形成了鲜明对比。开源代码仓库（https://github.com/Bio-innovator/biochem-V2）的公开，也意味着其他学校或开发者可以基于此项目进行二次开发，进一步扩大其影响力。

8 反思与展望

8.1 当前局限

尽管 v2.1 已经实现了核心的模考与练习功能，但从内容深度与系统成熟度来看，仍存在若干已知的结构性局限。

第一，FRQ 自动判分尚未实现。 当前 FRQ 部分仍需学生在纸笔上完成作答后，自行对照 Scoring Guidelines 进行自我评估，系统无法对文字回答进行自动判分。这并非前端交互的遗漏，而是在线考试系统的普遍技术难题——自然语言理解的评分标准与 AP 官方阅卷的精确度要求之间存在显著差距，也是未来需要重点攻克的方向。

第二，题库规模严重不足，难以支撑反复练习与精准诊断。 真题模考模块目前仅录入 AP Biology 2024 Full Exam, Asia 卷一套试卷，共 66 道题目（60 道 MCQ + 6 道 FRQ）。对于备考周期通常长达数月的 AP Biology 学生而言，一套试卷只能完成一次完整模考，无法通过多次模拟实现"刷题巩固"与"时间压力脱敏"。更关键的是，单次考试的题目分布与学生的个人薄弱单元未必匹配，缺乏多卷交叉比对，模考系统的诊断价值被严重削弱。

小测系统的题目储备同样面临数量瓶颈。当前 Quiz 模块仅录入 40 道题目，分布在 AP Biology CED 的 8 个 Unit 中，平均每个 Unit 不足 5 题。这一密度远不足以支撑有效的单元针对性练习：学生完成某一 Unit 的测试后，系统无法基于足够大的样本量生成可靠的错题模式分析，也无法通过"同知识点变式题"帮助学生验证修复效果。从教育测量学的角度，40 题的总量仅够完成一次粗略的能力筛查，而达不到"形成性评估"所需的题量门槛。

8.2 未来方向

展望未来，平台有以下几个可能的发展方向。第一，真题库扩充。计划录入更多历年的官方试卷，使学生能够进行多次模拟考试而不重复鼎题。第二，AI 辅助解析。借助大语言模型的能力，为学生提供智能的错题解析和知识点推荐。特别是对于 FRQ，可以尝试利用 AI 对学生的文字回答进行初步评分和反馈。第三，社区功能。增加讨论区、笔记分享等功能，让学生之间可以互相帮助、共同进步。第四，移动端优化。虽然当前平台已经具备响应式设计，但在移动设备上的体验还有进一步优化的空间。

每一个方向都有其挑战，但也都承载着让平台更加完善的期待。无论如何，项目的初衷不会改变——为每一个生物学习者找到属于自己的“生态龛”。

9 结论

Biochem-niche 是一个从真实学习需求出发、经历多轮迭代演进的全栈 Web 应用。从最初的静态网站到如今的 Next.js + Prisma + Supabase + Vercel 全栈架构，每一步都基于用户反馈和技术债务的反思。v2.1 版本的真题模考系统、双语支持和管理员功能，标志着平台从“学习工具”进化为“考试平台”的关键转折。项目不仅是校内首个系统化的 AP Biology 学习平台，也是 PBL 教学模式与工程实践相结合的成功案例。开源代码仓库已发布至 GitHub（https://github.com/Bio-innovator/biochem-V2），欢迎更多开发者和教育工作者共同参与建设。**[7]**

参考文献

[1] College Board. AP Biology Course and Exam Description [EB/OL]. 2025.

[2] Campbell N A, Urry L A, Cain M L, et al. Campbell Biology [M]. 12th ed. Pearson, 2021.

[3] Vercel. Next.js Documentation [EB/OL]. https://nextjs.org/docs.

[4] Prisma. Prisma ORM Documentation [EB/OL]. https://www.prisma.io/docs.

[5] Supabase. Supabase Documentation [EB/OL]. https://supabase.com/docs.

[6] Tailwind CSS. Tailwind CSS Documentation [EB/OL]. https://tailwindcss.com/docs.

[7] GitHub. Biochem-niche V2 Repository [EB/OL]. https://github.com/Bio-innovator/biochem-V2.

[8] College Board. Bluebook Testing Application [EB/OL]. https://bluebook.collegeboard.org/.