太好了，我会把你给的《AI Tutor — 技术方案（v1）》专门“按前端界面搭建”的角度拆成一串可独立执行的小任务，并给出一步步的实施指南（用假数据或本地 mock；不要求你打通真实 API）。

**总览：你要完成的“前端专属”里程碑**

**里程碑 0：项目骨架与基础设施**  
初始化 Next.js + TS + Tailwind + shadcn/ui + React Query + 状态管理 + 表单校验 + i18n + 测试与代码风格。

**里程碑 1：设计系统与主题（设计令牌）**  
把 Figma 的变量（颜色/字号/间距等）落到 CSS Variables 与 Tailwind，并准备亮/暗主题切换。

**里程碑 2：全局框架与路由**  
App Shell（侧边栏/顶部导航/面包屑/面板区域/Toast/Modal/错误边界/NotFound），集成加载骨架与空状态。

**里程碑 3：核心页面**（按 Figma→技术映射顺序实现）

1. 班级列表（Classes-List，表格/筛选/批量导出）
2. 学生导入向导（Students-Import，上传→字段映射→校验预览→结果下载）
3. 作业列表（Assignments-List，标签筛选、发布、触发批改）
4. 批改进度（BatchGrading，进度条/失败重试/日志）
5. 反馈报告查看（FeedbackReport，PDF 预览/下载）
6. 统计分析（Analytics，图表与 CSV/PNG 导出）
7. 系统设置-API（Settings-API，连通性测试/延迟与日志展示）  
   以上 7 个页面均在方案中明确给出，且对应具体交互与接口契约（你只做界面与状态，不做真实调用）。

**里程碑 4：可用性与质量**  
国际化 & 无障碍（a11y）、观测埋点（仅前端）、Playwright 端到端用例、性能预算与打包优化、演示版部署。

**里程碑 0：项目骨架（1–2 天）**

**任务 0.1 初始化工程**

1. 用 Next.js(App Router) 创建工程：  
   npx create-next-app@latest web --ts --eslint --tailwind --app --src-dir --import-alias "@/\*"
2. 安装依赖：
   * UI：tailwindcss、@radix-ui（随 shadcn/ui）、lucide-react
   * 组件库：shadcn/ui（用其 CLI 初始化）
   * 数据/状态：@tanstack/react-query、zustand
   * 表单/校验：react-hook-form、zod、@hookform/resolvers
   * 表格/图表：@tanstack/react-table、recharts
   * i18n：i18next、react-i18next
   * 其他：classnames 或 clsx、tailwind-merge
3. 代码质量与测试：  
   eslint、prettier、husky+lint-staged、@testing-library/react、playwright。

**任务 0.2 全局状态与“假后端”**

* 建立 utils/apiClient.ts 仅负责 **模拟** 请求（用 msw 或本地 setTimeout + 假数据 JSON）。
* 统一用 React Query 的 QueryClientProvider 包裹应用；设定全局重试/缓存策略与错误处理 Toast。

**任务 0.3 基础布局**

* 在 app/layout.tsx 放置 App Shell（Header/Sidebar/Main/Toaster）。
* 做通用组件：按钮、输入框、选择器、分页条、面包屑、空态、加载骨架、确认对话框。
* 加上错误边界与 not-found.tsx。

**完成标准**：能启动、切页、看到统一主题与 Toast；ESLint/Prettier 正常；示例 API 返回假数据。

**里程碑 1：设计系统与主题（1–2 天）**

**任务 1.1 设计令牌落地**

* 在 styles/tokens.css 定义 CSS 变量（主色、文本色、间距、圆角、阴影、动效、断点等），并在 tailwind.config.ts 里映射到 theme.extend.colors 等。
* 预设 :root 与 .dark 两套变量，并在 Header 放置主题切换按钮。  
  （方案要求从 Figma Variables → Style Dictionary → CSS Vars；若暂无自动导出，先手录关键 Token。）

**任务 1.2 shadcn/ui 主题化**

* 用 Token 覆盖 shadcn 的 --primary、--muted 等变量；验证按钮、表单、对话框在亮/暗模式下对比度达标。

**完成标准**：全站样式统一；亮/暗一键切换；基础组件遵循 Token。

**里程碑 2：全局框架与路由（0.5–1 天）**

**任务 2.1 路由结构（App Router）**

/ (仪表盘可选)

/classes

/students/import

/assignments

/jobs/[id]

/reports/[id]

/analytics

/settings/api

**任务 2.2 通用交互**

* 顶部搜索框（预留，不连后端）。
* 面包屑随路由更新；所有页面有统一“导出/刷新”区位。
* Loading Skeleton 与 Empty State 组件复用。

**里程碑 3：核心页面（2–4 天）**

下面每个页面都给**界面元素**、**本地假数据结构**、**用户路径**与**验收点**。API 路径在方案中已有（你只要在注释里标上对应接口名，保持字段一致即可）。

**A. 班级列表（Classes-List）**

* **组件**：DataTable（TanStack Table）+ 筛选条（关键词、创建时间）+ 新建班级 Modal + 批量选择 + 导出按钮。
* **假数据**：Class { id, name, createdAt, studentsCount, assignmentsCount }。
* **用户路径**：查看 → 搜索/分页 → 勾选多行 → 点击“导出 CSV”。
* **接口参考**：GET /classes、POST /classes、POST /export（仅做 UI）。
* **验收**：分页/筛选有效；创建后乐观更新；多选后显示导出条；空态/异常态可见。

**B. 学生导入向导（Students-Import）**

* **组件**：Step 步进器（下载模板 → 上传 → 字段映射 → 预检预览 → 提交 → 结果页）。
* **上传**：用 <input type="file"> + 本地解析（PapaParse）做**预览**；“提交”按钮处仅显示“将调用：上传到 S3 的预签名 URL + 创建 ImportBatch”。
* **映射**：左为系统字段（name/studentNo/classId），右为文件列名下拉。
* **结果页**：显示成功/失败数量、下载“错误明细.csv”的按钮（点了就下载内置示例）。
* **接口参考**：POST /students/import、GET /imports/:batchId。
* **验收**：必填/唯一/关联等校验规则在预览中给出提示（基于 zod）；进度用条形图与状态 Chip 展示。

**C. 作业列表（Assignments-List）**

* **组件**：DataTable + 标签筛选（KnowledgeTag）+ “发布”/“触发批改”按钮（带二次确认）。
* **假数据**：Assignment { id, title, status, tags[], createdAt }。
* **接口参考**：GET /assignments、POST /assignments、POST /assignments/:id/grade/batch。
* **验收**：状态 Tag 清晰（草稿/已发布/批改中/已完成）；点击“批改”后跳转到“进度页”。

**D. 批改进度（BatchGrading）**

* **组件**：进度条、完成/总量、失败计数、当前任务 ID、失败列表（可“重试失败项”）。
* **轮询**：用 React Query refetchInterval 模拟 GET /jobs/:jobId；也可准备“切换为 WebSocket”开关（仅 UI）。
* **接口参考**：GET /jobs/:jobId。
* **验收**：实时更新；结束后出现“生成报告/查看统计”的快捷按钮。

**E. 反馈报告查看（FeedbackReport）**

* **组件**：PDF Viewer（react-pdf）+ 下载按钮 + 元信息卡片（作业名/时间/评分摘要）。
* **接口参考**：POST /assignments/:id/report/feedback（生成）+ GET /reports/:id（下载）；你只做“点击后展示示例 PDF”。
* **验收**：A4 预览清晰；加载态与失败态（提示“文件不存在/权限不足”）。

**F. 统计分析（Analytics）**

* **组件**：Tabs（总览/按知识点/分布）+ Recharts（柱状/箱线/饼图）+ 导出区（CSV/PNG）。
* **假数据**：平均/中位、分布数组、Top 错题、按知识点正确率。
* **接口参考**：POST /assignments/:id/report/analytics（仅 UI）。
* **验收**：切换 Tab 不丢失筛选；导出会生成本地虚拟 CSV/PNG 文件下载。

**G. 系统设置-API（Settings-API）**

* **组件**：表单（api.gateway.url）+ “测试连通性”按钮 + 结果卡片（ok/latency/log）。
* **接口参考**：GET /settings、POST /settings/test-connection（UI 展示延迟/HTTP 状态/部分响应体）。
* **验收**：错误时高亮字段并展示可读日志；保存后顶部出现“已保存”Toast。

**里程碑 4：可用性、测试与质量（1–2 天）**

**任务 4.1 国际化 & 无障碍**

* 词条分命名空间：common/classes/students/assignments/data/settings；提供中/英两套 JSON。
* a11y：焦点可见、表格键盘可达、弹窗 Trap Focus、ARIA 标签、对比度达标。

**任务 4.2 可观测（前端）**

* 接入前端错误上报（Sentry）与基础 Web Vitals；将“页面→操作→结果”以事件形式打印到控制台（模拟 OTel Trace）。

**任务 4.3 测试**

* 单元/组件：表格渲染、表单校验、导入向导每一步的状态切换。
* 端到端（Playwright）：**三条关键流**
  1. 新建班级→列表出现→导出 CSV
  2. 导入学生：上传→映射→预览→提交→结果页下载错误明细
  3. 触发批改→进度达 100%→查看报告 PDF 与统计图表
* 性能：Lighthouse CI 确认 LCP ≤ 2.5s、首屏 JS < 150KB（不含框架）。

**每个任务的“操作清单”（拎包即做）**

下面给你每页的**落地步骤模板**，直接照抄套用。

**通用页面脚本模板**

1. 在 app/<route>/page.tsx 新建页面。
2. 放置 PageHeader（标题/副标题/操作按钮）。
3. 主体区域写 Card 容器；内含：
   * ClientOnly 包裹数据组件（避免 SSR 报错）。
   * Suspense + Skeleton 占位。
4. 用 React Query 写数据钩子 use<Domain>Query()（此处对接 **mock**）。
5. 处理三种状态：isLoading/isError/isSuccess；错误给 Toast。
6. 空态显示图标 + 操作按钮（如“新建”“上传”）。
7. 写 playwright 用例覆盖加载→交互→导出。

**DataTable 落地（Classes/Assignments 通用）**

* 列定义：columns.tsx（Accessor/Cell/Sorting/Filters）。
* 服务器分页（UI 层面）：pageIndex/pageSize 在 URL 查询参数中保持；分页器组件复用。
* 批量选择：勾选后在底部显示“批量操作条”，放“导出”。
* 筛选条：关键字输入 + Tag 多选（Assignments）。
* 行操作：三点菜单（查看详情、删除/发布/触发批改）。

**导入向导落地（Students-Import）**

* Step1“下载模板”：提供一个内置 CSV 模板下载。
* Step2“上传”：接收文件、用 PapaParse 读取前 N 行，显示列名与 10 条示例。
* Step3“字段映射”：左为系统字段，右为下拉（来自上传文件列名）。用 react-hook-form + zod 校验必填。
* Step4“预检预览”：基于映射跑校验（缺 name/studentNo、重复 studentNo、classId 不存在等），错误行标红并展示原因。
* Step5“提交”：显示“将上传到对象存储并创建导入任务”的提示，点确认后进入“进度页”。
* Step6“结果反馈”：显示成功/失败统计，提供“下载错误明细.csv”（用本地生成 Blob）。

**批改进度页落地**

* 顶部：进度条 + 数值（完成/总量）+ 失败计数。
* 中部：失败列表（Submission ID/错误信息），提供“重试失败项”按钮（仅 UI）。
* 底部：完成后亮起“生成反馈报告”“打开统计分析”。轮询间隔 1–2s（mock）。

**报告 PDF 查看落地**

* react-pdf 的 Document + Page 组件；提供页码/缩放/下载。
* 加载失败：展示“无法加载，稍后再试 / 检查权限”。
* 右侧卡片列出元信息与生成时间。

**统计分析落地**

* Tabs：总览（均值/中位/分布）/ 知识点 / Top 错题。
* 用 Recharts 画柱状/箱线/饼图；提供“导出 CSV/PNG”。
* 图表空态（数据不足时）要有引导文案。

**设置-API 落地**

* 表单只含 api.gateway.url 输入框与“测试连通性”按钮。
* 点击后出现结果卡片：ok/latency/log；失败则红色 Alert + 详细日志（字符串滚动区）。

**交付检查清单（对照验收标准逐条过）**

* 所有表单：同步/异步校验 + 友好错误提示
* 所有列表：分页/筛选/导出
* 导入流程：模板下载→上传→映射→预检预览→结果反馈（含错误明细下载）
* 批改流程：触发→进度可见→失败可重试→完成后可进入报告/统计
* 报告：PDF 预览/下载；统计：CSV/PNG 导出
* 多租户/多环境相关字段在 UI 有清晰标识（如页眉角落显示当前租户）
* 观测、a11y、性能预算满足（LCP≤2.5s，首屏 JS<150KB）  
  以上条目均来自技术方案的验收要求与页面映射。

**给你的“快捷命令与文件清单”（抄了就能跑）**

* 初始化 shadcn/ui：npx shadcn@latest init → npx shadcn@latest add button input dialog toast table tabs card
* 目录建议：

app/

(routes...)

components/

ui/ (shadcn 扩展)

table/

charts/

forms/

features/

classes/

students/

assignments/

jobs/

reports/

analytics/

settings/

lib/

hooks/

utils/

mocks/ (msw handlers + 假数据)

styles/ (tokens.css, globals.css)

* Playwright 脚本：放在 tests/e2e/\*，覆盖三条关键流（见上）。