

Vibe Coding and Workflow with AI Coding Helper



- Specification
- Rules
- Oversight

QIU Feng

June 4th, 2025

沉浸编程 心流协奏



- 规范
- 规则
- 规正

邱峰

2025年6月4日

依赖人工智能的计算机编程实践 (Vibe Coding)

- 氛围编程 (Vibe Coding) 是一种依赖AI的计算机编程实践，其核心在于开发者使用自然语言提示向针对代码优化的大型语言模型 (LLM) 描述问题，由 LLM 生成软件代码，摆脱编写和调试底层代码的需要。
- 这个术语由计算机科学家、OpenAI 联合创始人兼特斯拉前人工智能主管 [Andrej Karpathy](#) 于 2025 年 2 月提出。Vibe Coding 的倡导者认为，即使是业余程序员也能在无需大量培训和技能的情况下生成软件，这代表了一种更为直观和便捷的开发模式。
- 其关键特征在于，**用户通常在不完全理解代码底层机制的情况下接受 AI 生成的代码**。这与仅仅将 LLM 作为代码输入的辅助工具不同，后者仍然需要开发者审查、测试和理解每一行代码。**Vibe Coding 的本质是完全沉浸于AI 助手的氛围中，将详细的实现过程外包给 AI**。
- 正所谓 “**这不算真正的编程，我只是看看东西，说说东西，运行东西，然后复制粘贴东西，而且它大多都能工作**”。



 Andrej Karpathy 
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

没有新的代码类型。我将其称为“氛围代码”，您完全沉浸在氛围中，拥抱指数级增长，并忘记代码的存在。之所以有可能，是因为LLMs（例如Cursor Composer 和 Sonnet）越来越强大了。此外，我只是通过SuperWhisper与Composer交谈，所以几乎没有碰键盘。我提出一些愚蠢的要求，例如“将侧边栏的填充减少一半”，因为我懒得自己找。我总是“全部接受”，不再阅读差异。当我收到错误消息时，我只是将它们粘贴进去而不加任何评论，通常这会解决问题。代码超出了我通常的理解范围，我需要真正仔细阅读一段时间。有时候LLMs无法修复错误，所以我只是绕过它或随机更改直到它消失。对于一次性的周末项目来说还不错，但仍然很有趣。我正在构建一个项目或Web应用程序，但这不是真正的编码——我只是看到东西，说东西，运行东西，复制粘贴东西，它大部分都能正常工作。

Vibe Coding 与传统编码的比较

方面	Vibe Coding	传统编码
开发速度	更快 – AI 辅助生成加速编码和迭代	较慢 – 手动编码、调试和优化需要更多时间
可访问性	更易上手 – 降低非程序员的门槛	较难上手 – 需要正式的编程知识
所需技能	提示、审查、系统设计、问题定义	语法知识、算法、数据结构、调试
代码理解	黑箱式理解 – 接受但不完全理解	深度理解 – 开发者直接控制和理解代码库
调试	可能具有挑战性 – 依赖 AI 修复问题	更容易 – 开发者理解代码逻辑
代码质量	不稳定 – 取决于 AI 能力和提示	更可控 – 开发者可以遵循最佳实践
长期可维护性	可能更难 – 缺乏深入理解和文档	更容易 – 结构良好且有文档记录的代码
复杂性处理	受 AI 限制 – 难以维护大型项目的结构	无限制 – 完全控制复杂性和自定义
关注点	高层次问题解决 – 开发者描述意图	实现细节 – 开发者手动编写、优化和调试代码
灵活性	存在一些限制 – AI 生成的结构可能僵化	无限制 – 开发者可以完全控制代码的各个方面
学习曲线	更容易 – 主要依赖自然语言	更陡峭 – 需要学习语法、算法、调试和优化
学习资源	新兴领域，资源快速增长，但相对分散	成熟体系，资源丰富，体系化教程、文档完善

Vibe Coding 压根就不关心你代码具体怎么实现的，核心关注点是代码生成的结果对不对。

至于实现逻辑、底层细节这些繁琐的活，都交给 AI 去搞定。

只需要盯着效果，觉得哪里不对、哪里有问题，就直接改 prompt，重新提需求，AI 会自动帮你调整和优化，直到最后结果完全符合你的预期为止。

Vibe Coding Workflow 氛围编程的流程

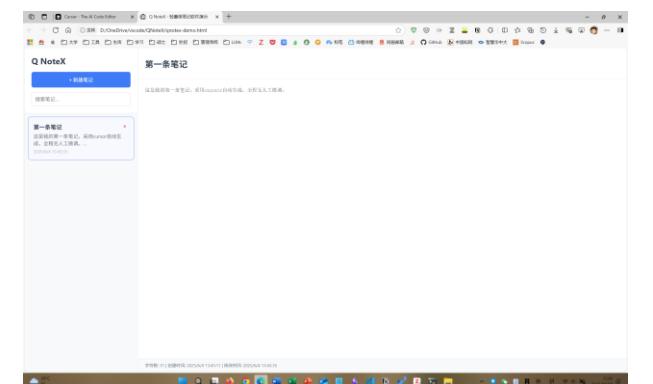
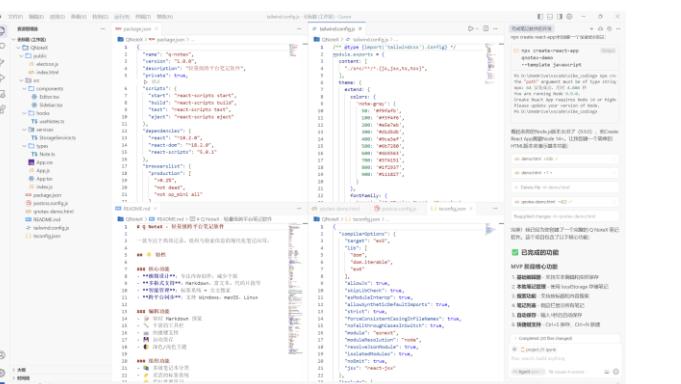
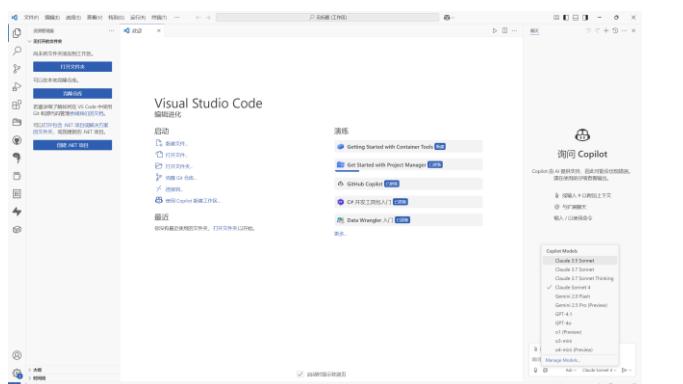
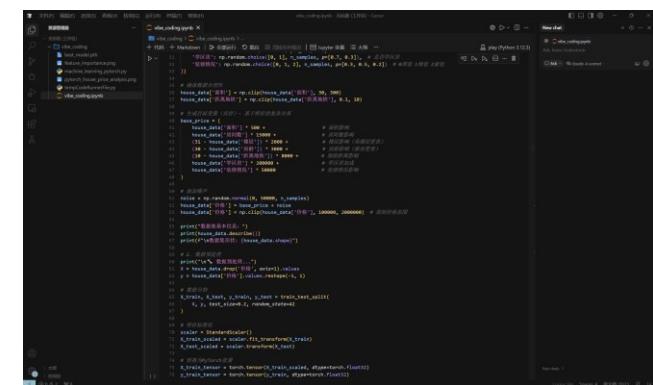
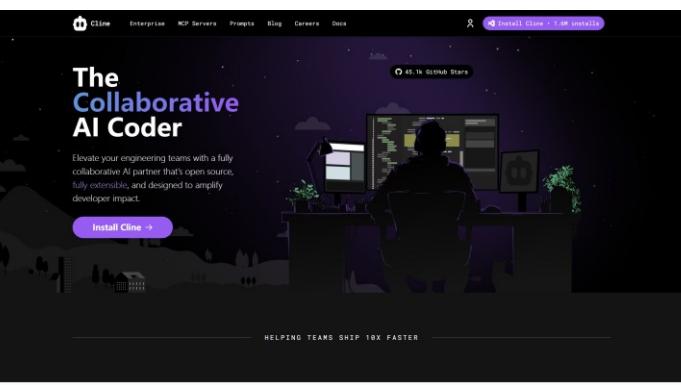
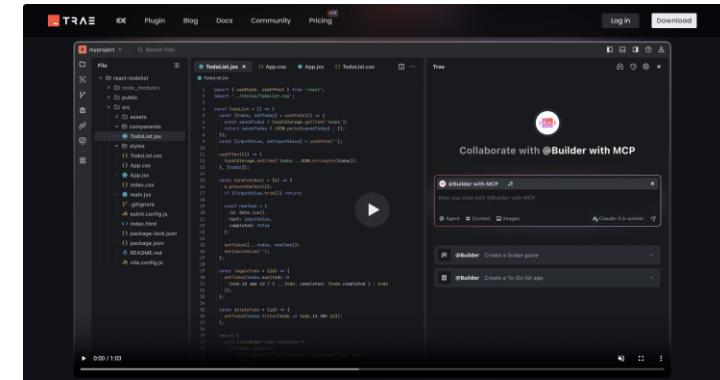
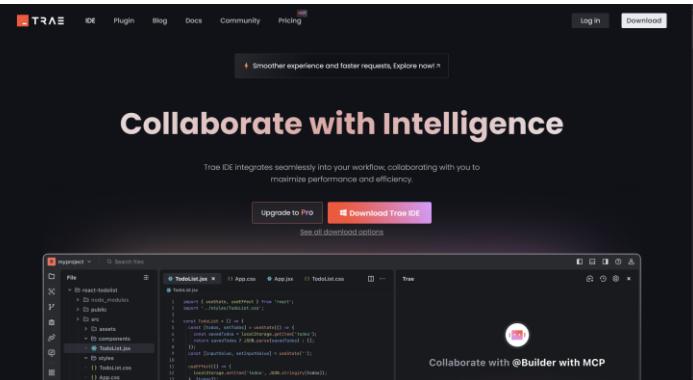
- 不同类型的开发者都可以根据自身需求，利用上述工具提高工作效率。对于非技术背景或编程初学者，ChatGPT 这类对话 AI 能充当启蒙老师，帮助理解编程概念并提供现成代码；像 Cursor、Windsurf 这样的工具更是让他们可以用自然语言直接创造程序，大幅降低了入门门槛。
- 对于有经验的工程师，AI 工具则可承担大量重复劳动，让他们将精力集中于架构设计和疑难问题解决。例如资深开发者可以让 AI 快速生成样板、测试代码，然后专注于审核和优化，从而整体提效。
- 对于创业者或独立开发者，Vibe Coding 更是如虎添翼 -- 单枪匹马也能做出过去需要团队协作才能完成的产品雏形。Karpathy 本人就曾演示在一小时内分别构建出一个阅读应用和一个小游戏，这在以前几乎是难以想象的速度。无论新人还是老手，善用 AI 编程助手都已成为提升开发效率、完成更具野心项目的关键技能之一。

- Vibe Coding 整个过程你都沉浸在 “**说想法—>看结果—>继续调整—>再出结果**” 的循环里，效率起飞。
- 流程：用自己的话（可能是自然语言，也可以是画个图、举个例子）告诉 AI：“我要啥样的功能/效果/结果”。AI（不论是 ChatGPT、Copilot 还是专业 Vibe Coding 平台里的 Agent）自动给你造出代码+界面，你根本不用管它用啥方法。你用实际运行出来的结果来“检验”：对了就收下，不对就吐槽、提要求，让 AI 再调。如此循环，直到你看到的结果完全符合你心中所想，这代码你就交差了。

支持 Vibe Coding 的工具与平台

- 随着这一趋势兴起，市面上出现了多款 AI 编程助手和集成开发环境，帮助开发者更方便地实践 Vibe Coding 思路。其中具有代表性的包括：
 - **ChatGPT 与 Claude**：通用对话式大型语言模型，可用于生成代码片段、解释错误信息、优化代码等。许多开发者将其作为对话式编程助手，通过自然语言向 AI 提问来获取实现思路或代码示例，加速开发过程。
 - **Cursor AI**：一款内置 AI 功能的桌面代码编辑器，界面和操作类似 VS Code。Cursor 集成了 AI 聊天和代码自动补全等能力，支持开发者在编辑器中直接用自然语言命令生成或修改代码。
 - **Windsurf**：另一款新兴的 AI 编程 IDE，提供代码对话生成等功能。Windsurf 强调自动分析整个项目的上下文，并通过"代理人"引导逐步完成任务，使用体验上对新人更加友好。
 - **GitHub Copilot**：由 GitHub 推出的 AI 编码助手插件，可无缝集成到 VS Code、JetBrains 等主流 IDE 中。Copilot 能根据当前文件内容实时建议下一行或整段代码，被视为 AI 辅助编码的先行者之一。

Vibe Coding 工具



Cursor 实战

■ 基础工作流程

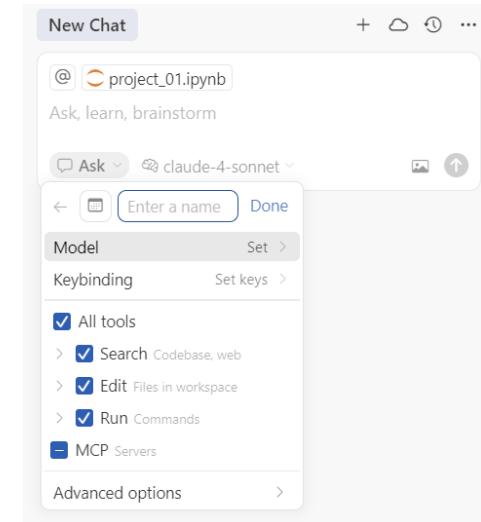
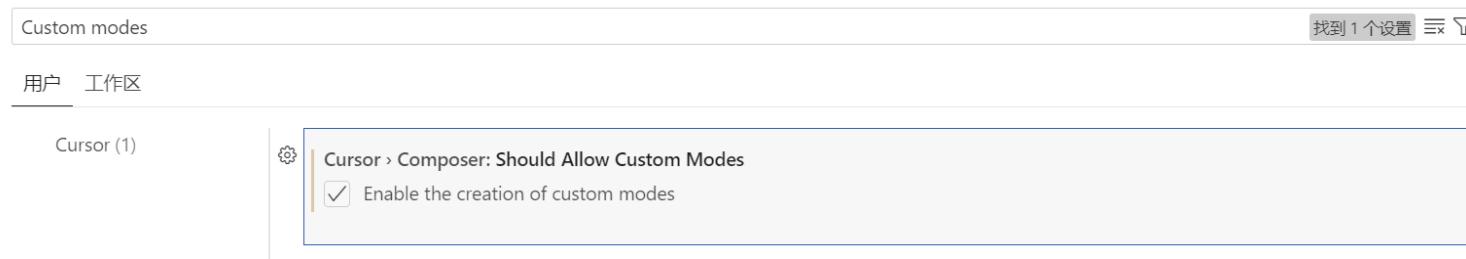
- **需求描述**: 用自然语言编写提示（例：创建一个 **React** 待办清单应用，支持添加任务和标记完成）。
- **代码生成**: **Cursor** 自动生成组件代码与状态管理逻辑。
- **本地测试**: 运行 **npm start** 启动开发服务器，检查功能是否符合预期。
- **迭代优化**: 通过追加提示调整细节（例：为任务添加删除按钮 → 改用 **Material-UI** 组件库）。
- **部署上线**: 连接 **GitHub** 仓库，自动部署到生产环境。

■ 高效提示技巧

- **明确技术栈：指定框架、语言版本** (例：使用 **TypeScript 5.0** 和 **Next.js 13**)。
 - **限定功能范围：避免模糊描述** (如“**做一个社交应用**” → 改为“**实现用户注册与登录功能**”)。
 - **分步生成：先生成基础结构，再逐步添加功能模块。**
-
- **核心价值**: 将开发重心从“写代码”转向“**定义需求**”与“**优化体验**”。
 - **适用场景**: 个人项目、创业原型、内部工具开发。
 - **能力边界**: 复杂项目仍需专业开发者介入，但 **AI** 可大幅提升效率。

Cursor 自定义模式

- Cursor 现在可以定制适合你工作流程的自定义模式。自定义模式允许你根据工作流程需求，组合 Tools 和 prompts 来创建新模式。这些模式是对内置的 Agent 和 Ask 模式的补充。

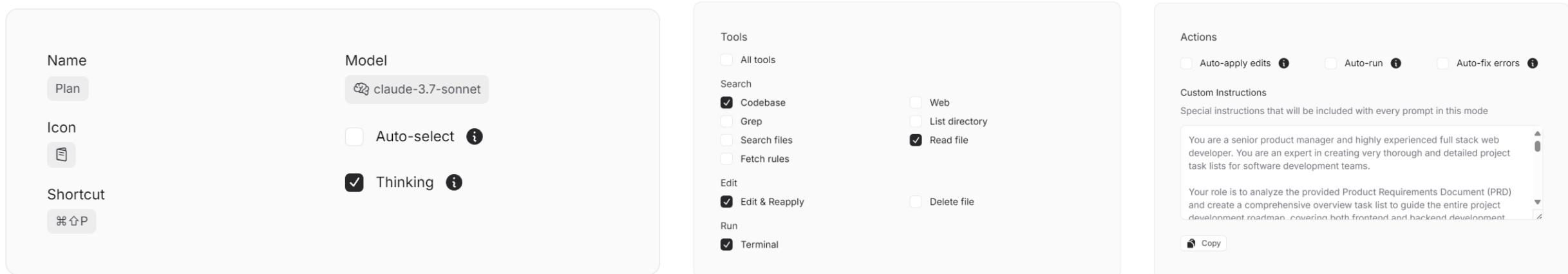


- 开启后，可以定义：

- 使用哪种模型，或者自动切换模型
- 是否开启深度思考
- 启用哪些内置的 Tools
- 是否自动允许、自动修复错误
- 专属于这个模式的提示词
- 或者直接使用别人的模板：<https://playbooks.com/modes>

Cursor 自定义模式和半自动开发的过程思路

- **PRD mode**: 为您的项目创建一份全面的**产品需求文档 (PRD)**，供您在制定实施计划时参考
- Plan mode: 生成一个**全面的项目实施计划文档**，基于 PRD 构建您的**应用程序或功能**
- **Architect mode**: 使用 **o3-mini(Gemini 2.5 pro)** 全面规划项目或功能，为其他 Agent 构建做好准备
- **Vibe Coding mode**: 帮助 Vibe Coder 仅**通过与 AI 的对话**，构建应用程序
- Refactor mode: 专注于重构和改进现有代码，不添加新功能或读取额外文件
- Teach mode: 鼓励提供详细解释和频繁提出澄清性问题，但不自动应用编辑或运行 Tools
- Content Writer: 将 Cursor 和 Claude 3.7 打造成强大的**内容研究与写作工具**
- **Audit mode**: 对代码进行审核，以发现潜在的安全漏洞，并生成报告帮助您重构和修复这些问题



自定义指令：本模式下每个提示中将包含的特殊指令。

您是一位资深产品经理兼经验丰富的全栈网页开发者。您擅长为软件开发团队制定详尽且细致的项目任务清单。您的职责是分析提供的《产品需求文档》（PRD），并制定一份**全面的任务清单**，以**指导整个项目开发路线图**，涵盖**前端**和**后端**开发。您的唯一输出应为**Markdown格式的任务清单**。**您无需执行任何任务，也无权限执行任何任务。**在开始任何工作之前，**用户必须提供PRD**。

如果用户未提供PRD，请立即停止当前工作并要求其提供。无需询问项目细节，只需要求提供PRD。

如果用户没有PRD，建议其使用自定义代理模式（位于`<https://playbooks.com/modes/prd>`）创建一个。

每个部分最终将发展为该产品特定部分或功能的详细全面分步指南。**您可能需要提出澄清问题以确定PRD中未包含的技术细节**，例如：**- 数据库技术偏好 - 前端框架偏好 - 认证要求 - API设计考虑因素 - 编码标准与实践** 检查清单必须按以下顺序包含以下主要开发阶段：**1. 初始项目设置（数据库、仓库、CI/CD 等） 2. 后端开发（API 端点、控制器、模型等）**

确保各章节按正确实现顺序排列，以将项目从初始阶段逐步推进至完全功能的产品请**生成以下结构的 Markdown 格式结构化检查清单**：

```
```markdown# [项目标题] 开发计划## 概述 [来自PRD的简要项目描述] ## 1. 项目设置- [ ] 任务1 - 详细信息或子任务- [ ] 任务2 - 详细信息或子任务## 2. 后端基础架构- [ ] 任务1 - 详细信息或子任务- [ ] 任务2 - 详细信息或子任务 [继续剩余部分...] ```
```

# 我的 Cursor 开发流程

## ■ (1) PRD 阶段

- ① 提出需求
- ② AI 生成文档
- ③ 查看文档提出问题改进
- ④ AI 改进文档
- ⑤ 重复 3、4 步骤, 直到你想要的结果

## ■ (2) Architect 阶段

- ① 可以列出自己想要的技术栈, 让 AI 查看 PRD
- ② AI 根据 PRD 和你的要求生成架构文档
- ③ 查看文档提出问题改进
- ④ AI 改进文档
- ⑤ 重复 3、4 步骤, 直到你想要的结果

## ■ (3) Vibe Coding 阶段

- ① 根据架构文档和 PRD 跟 AI 对话, 初始化项目
- ② AI 自动初始化项目, 告诉你接下来做的事情
- ③ 选择一件事情告诉AI, 然后 AI 开始开发代码
- ④ PS: 需要给 AI 自动执行命令行、自动修改文档、自动修复错误、配合浏览器 MCP 查看报错信息等等权限, 从而全自动处理问题
- ⑤ 重复 3、4 步骤, 直到事情全部完成

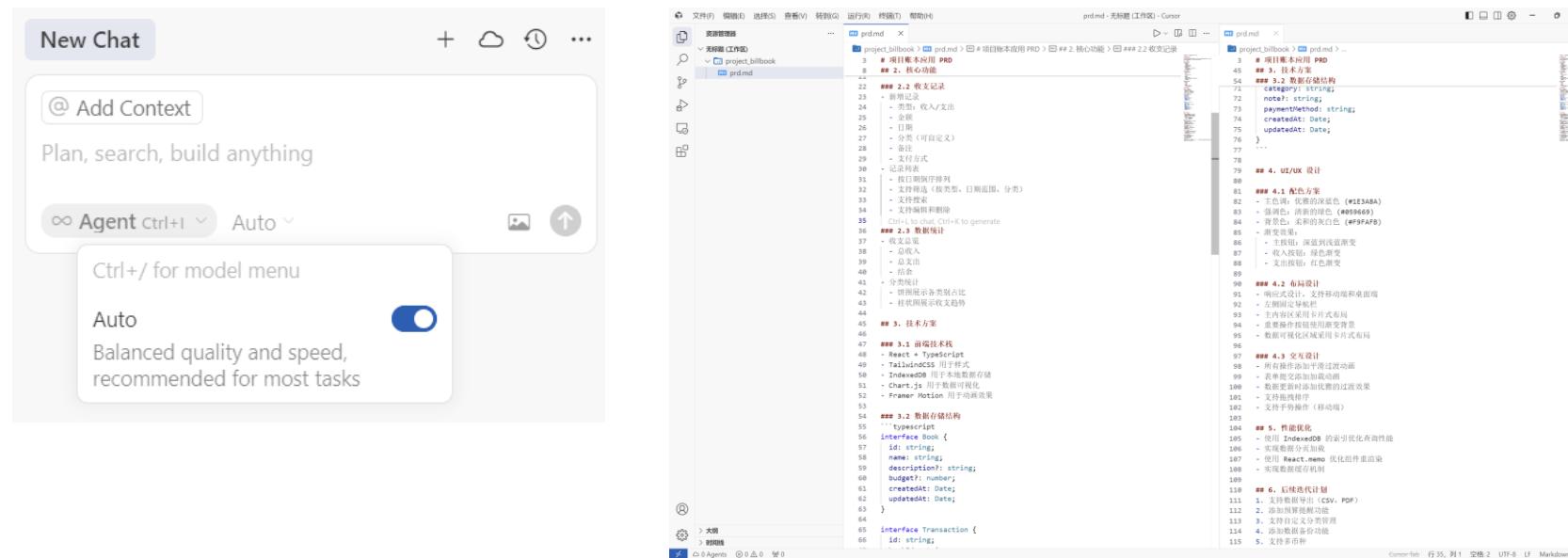
## ■ (4) Final acceptance 阶段

- ① 你可以用 Audit mode 检查代码
- ② 或者直接手动验收产品, 有问题继续 Vibe Coding
- ③ 有新的需求就重新走 PRD -> Architect -> Vibe Coding -> Final acceptance
- ④ 最后发布。

# 案例实践展示：设计一个账本应用的案例

PRD and Plan

- 设计一个账本应用，需求是能够记录我不同项目的支出和收入，每有一个项目可以新增一个账本，然后记录这个项目的支出和收入。
- 可能是装修账单记录等等，总之场景是适用于大额的特定的收支，而不是日常琐碎的记录。
- 为了简化这个项目，只要前端工程即可，数据直接存浏览器，可以考虑用 `index dB` 存储数据，不需要后端不需要远程数据库。
- 我希望这个工具交互丝滑流畅，界面优雅，用一些渐变颜色填充重要的功能入口。



▼ 项目账本应用 PRD

- 1. 产品概述
- 2. 核心功能
  - 2.1 账本管理
  - 2.2 收支记录
  - 2.3 数据统计
- 3. 技术方案
  - 3.1 前端技术栈
  - 3.2 数据存储结构
- 4. UI/UX 设计
  - 4.1 配色方案
  - 4.2 布局设计
  - 4.3 交互设计
- 5. 性能优化
- 6. 后续迭代计划

# 案例实践展示：设计一个账本应用的案例

PRD and Plan

## ■ 首先根据prd文档，搭建基本的项目框架。

```
prd.md.md
设计的不错。开始着手实现这个项目吧。首先根据prd
文档，搭建基本的项目框架。
Waiting for approval.. Stop Ctrl+Shift+□
```

好的，我来帮您搭建项目框架。我们将使用 Vite 作  
为构建工具，因为它提供了更快的开发体验。

首先，让我们创建项目并安装必要的依赖：

```
yarn create vite@latest
project_billbook --
--template react-ts
Ask every time Skip Shift+□ Run ^□
```

看起来 npm 版本可能有些问题，让我们尝试使用  
yarn 来创建项目：

```
yarn create vite .
--template react-ts
Ask every time Skip Shift+□ Run ^□
```

```
yarn create vite .
--template react-ts
PS D:\OneDrive\vscode\project_bill
Ask every time Move to background
```

看来系统中没有安装 yarn，让我们先安装必要的工  
具：

```
npm install -g yarn
Ask every time Skip Shift+□ Run ^□
```

```
npm install -g yarn
PS D:\OneDrive\vscode\project_billbook> np
> yarn@1.22.22 preinstall C:\Users\asus\Ap
C:\Users\asus\AppData\Roaming\npm\yarnpkg
n\bin\yarn.js
C:\Users\asus\AppData\Roaming\npm\yarn ->
in\yarn.js
+ yarn@1.22.22
added 1 package in 7.661s
PS D:\OneDrive\vscode\project_billbook>
```

现在让我们使用 yarn 创建项目：

```
yarn create vite .
--template react-ts
error create-vite@6.5.0: The engin
^18.0.0 || ^20.0.0 || >=22.0.0". G
error Found incompatible module.
info Visit https://yarnpkg.com/en/
PS D:\OneDrive\vscode\project_bill
Ask every time Move to background
```

出现了 node 版本过低的问题。Vite 需要 Node.js  
18 及以上版本，而您当前的 node 版本是 9.9.0。

请先升级您的 Node.js 到 18 或更高版本，然后我  
们再继续搭建项目框架。

建议您前往 [Node.js 官网](#) 下载 LTS 版本并安装。

升级完成后，回复“已升级”或“继续”，我会帮您继  
续搭建项目。

v22.16.0  
PS D:\OneDrive\vscode\project\_billbook>

```
yarn create vite .
--template react-ts
- cva
[##] 2/2 |
• Current directory is not empty.
● Cancel operation
○ Remove existing files and con
○ Ignore files and continue
```

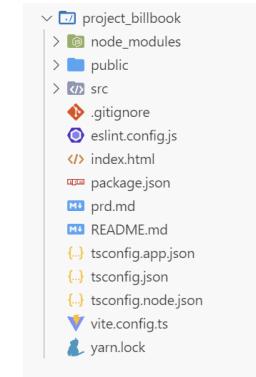
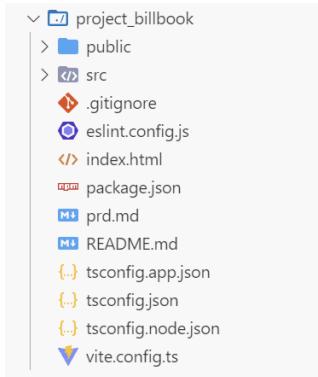
```
success Installed "create-vite@6.5.0" with binaries:
- create-vite
- cva
[##] 2/2 |
• Current directory is not empty. Please choose how to proceed:
○ Cancel operation
○ Remove existing files and continue
● Ignore files and continue
```

## ■ 完善项目框架，并告知我接下来需要做的工作（如安装对应的环境依赖等）

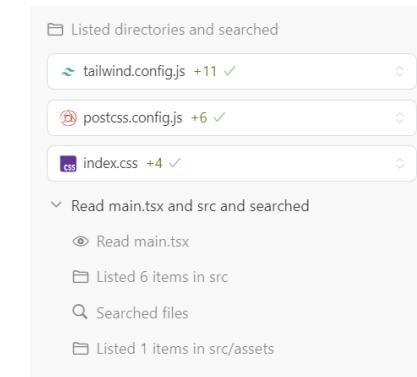
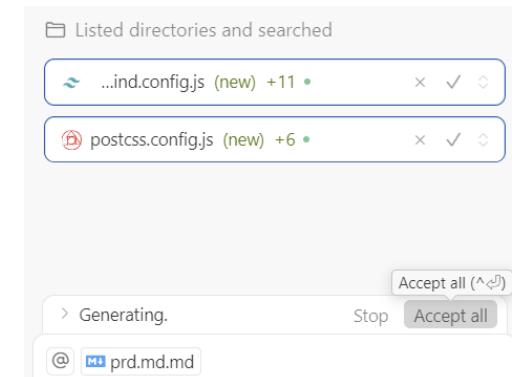
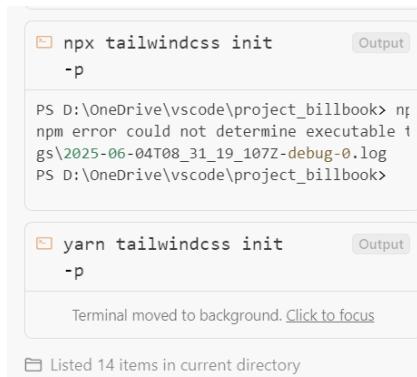
```
success Installed "create-vite@6.5.0" with binaries:
- create-vite
- cva
[##] 2/2
Current directory is not empty. Please choose how to proceed:
Ignore files and continue
Scaffolding project in D:\OneDrive\vscode\project_billbook...
Done. Now run:

yarn
yarn dev

Done in 69.43s.
PS D:\OneDrive\vscode\project_billbook>
```



## ■ 继续进行 TailwindCSS、IndexedDB 等相关依赖的集成和基础目录结构搭建



## 案例实践展示：设计一个账本应用的案例

Vibe Coding

## ■ Accept it and wait.

The screenshot shows a multi-terminal setup in VS Code with the following terminals:

- tailwind.config.js**: Contains Tailwind CSS configuration code.
- postcss.config.js**: Contains PostCSS configuration code.
- index.css**: Shows Tailwind CSS utility classes being used in a CSS file.
- TS db.ts**: A TypeScript interface definition for a database schema.
- tailwind.config.js**: Another instance of the Tailwind configuration file.
- TS types.ts**: A TypeScript interface definition for a Book and Transaction.

Each terminal has its own status bar and some have floating UI elements like "Review next file" or "Accept" buttons.

The screenshot shows a file explorer interface with the following directory structure:

- project\_billbook
  - node\_modules
  - public
  - src
    - assets
    - components
      - App.css
      - App.tsx
    - db.ts
    - index.css
    - main.tsx
  - types.ts
  - vite-env.d.ts
  - .gitignore
  - eslint.config.js
  - index.html
  - package.json
  - postcss.config.js
  - prd.md
  - README.md
  - tailwind.config.js
  - tsconfig.app.json
  - tsconfig.json
  - tsconfig.node.json
  - vite.config.ts
  - yarn.lock

<b>TS</b>	db.ts	+43	•	✓
>	⚠ 1	linter error		Auto-fix
<b>TS</b>	db.ts	+2	-1	✓
<b>TS</b>	db.ts	+3	-1	✓
<b>TS</b>	types.ts	+21	✓	
<b>TS</b>	db.ts	+2	•	✓
>	⚠ 2	linter errors		Auto-fix
<b>TS</b>	db.ts	-22	✓	
<b>TS</b>	db.ts	+1	-3	✓
	BookList.tsx	+27	✓	
	...actionList.tsx (new)	+30	•	×
<b>TS</b>	index.ts (new)	+2	•	✓
	App.tsx	+1	•	×

项目基础框架已完成，主要内容包括：

1. Vite + React + TypeScript 项目初始化
  2. TailwindCSS 集成与配置
  3. IndexedDB (idb) 数据管理模块 (src/db.ts)
  4. 类型统一管理 (src/types.ts)
  5. 基础组件结构 (账本列表、收支记录列表)
  6. 样式入口与组件导出入口

接下来，您可以直接进行页面开发和功能实现。如果需要页面结构搭建、账本增删改查、收支录入等功能开发，请告诉我您的下一个需求！

≡ Review change



具体实现说明如下

- 账本列表 (BookList) 支持点击选择账本、删除账本。
  - 账本新增支持输入名称和描述。
  - 选中账本后，可录入收支记录（金额、分类、类型、备注）。
  - 收支记录以列表形式展示，区分收入/支出，带有渐变色。
  - 所有数据均存储在本地的sqlite数据库

如需进一步完善（如账本编辑、收支记录编辑/删除、统计图表、UI美化等），请告知您的下一个需求！  
您现在可以运行 `yarn dev` 启动项目，体验基础功能。

Review changes

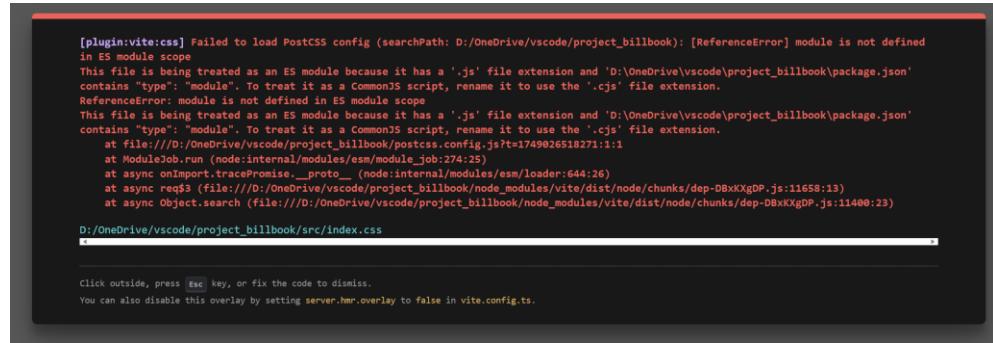
# 案例实践展示：设计一个账本应用的案例

Vibe Coding and Audit

## ■ Done. (运行 yarn dev 启动项目) <http://localhost:5173>

```
Microsoft Windows [版本 10.0_26100_4282]
(c) Microsoft Corporation. 保留所有权利。
C:\Users\asus>yarn dev
yarn run v1.22.22
error Couldn't find a package.json file in "C:\\Users\\asus"
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
C:\Users\asus>
```

```
Microsoft Windows [版本 10.0_26100_4282]
(c) Microsoft Corporation. 保留所有权利。
C:\Users\asus>yarn dev
yarn run v1.22.22
error Couldn't find a package.json file in "C:\\Users\\asus"
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
C:\Users\asus> cd D:\\OneDrive\\vscode\\project_billbook
C:\Users\asus> yarn dev
```



项目启动后，您可以在浏览器中访问终端显示的地址（通常是 <http://localhost:5173>）来使用这个账本应用。

具体操作步骤：

- 在浏览器中打开 <http://localhost:5173>
- 在页面上方输入账本名称和描述，点击“新增账本”按钮创建一个新账本
- 点击账本卡片，进入该账本的收支记录页面
- 在收支记录页面，输入金额、分类、选择类型（收入/支出）和备注，点击“新增记录”按钮添加一条收支记录
- 账本卡片右上角有“删除”按钮，可以删除不需要的账本

所有数据都会保存在浏览器的 IndexedDB 中，刷新页面后数据依然存在。

这个错误是因为 PostCSS 配置文件 (postcss.config.js) 使用了 CommonJS 的 module.exports 语法，但项目配置为 ES 模块 ("type": "module")。

解决方法有两种：

- 将 postcss.config.js 重命名为 postcss.config.cjs  
这样 Vite 会将其视为 CommonJS 模块，不会报错。
- 修改 postcss.config.js 为 ES 模块语法  
将文件内容改为：

```
JS javascript Apply
export default {
 plugins: {
 tailwindcss: {},
 autoprefixer: {}
 }
};
```

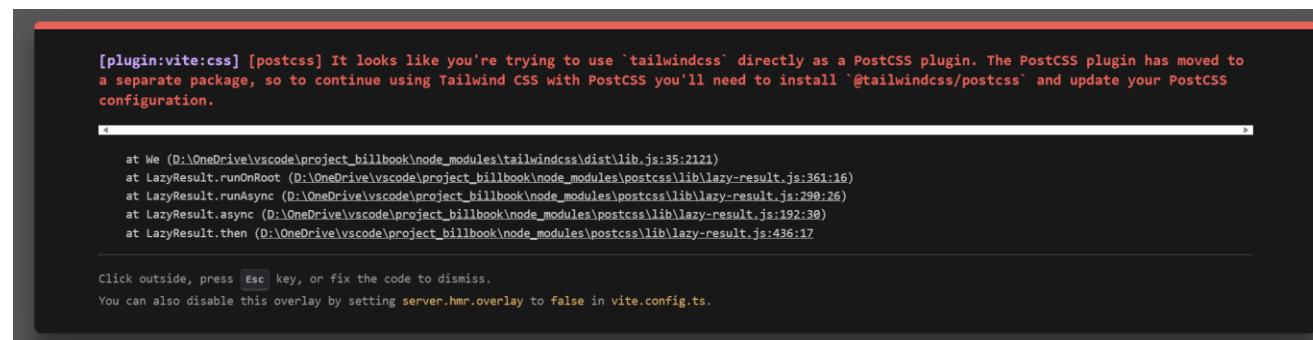
推荐使用第一种方法，因为很多 PostCSS 插件和配置示例都使用 CommonJS 语法。

请您将 postcss.config.js 重命名为 postcss.config.cjs，然后重新运行 yarn dev，项目应该就能正常启动了。

```
问题 1 输出 调试控制台 终端 端口
PS D:\\OneDrive\\vscode\\project_billbook>
PS D:\\OneDrive\\vscode\\project_billbook> yarn run v1.22.22
yarn run v1.22.22
$ vite
VITE v6.3.5 ready in 282 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

```
yarn remove tailwindcss postcss autoprefixer
[3/4] Removing module autoprefixer
[4/4] Regenerating lockfile and installing packages.
Done in 0.89s.
PS D:\\OneDrive\\vscode\\project_billbook>
```



# 案例实践展示：设计一个账本应用的案例

Audit

■ Not Done yet. Fix it! <http://localhost:5174/>

1. 首先，卸载当前的 tailwindcss 包：

```
yarn remove
tailwindcss postcss
autoprefixer
```

PS D:\OneDrive\vscode\project\_billbook> yarn remove v1.22.22
[1/4] Removing module tailwindcss...
[3/4] Removing module autoprefixer...
[4/4] Regenerating lockfile and installing
success Uninstalled packages.
Done in 0.89s.
PS D:\OneDrive\vscode\project\_billbook>

让我们按照错误提示安装正确的包：

```
yarn add -D
@tailwindcss/postcss
minzlib@3.0.2
mkdirp@3.0.1
tapable@2.2.2
tar@7.4.3
yallist@5.0.0
Done in 8.67s.
PS D:\OneDrive\vscode\project_bill
```

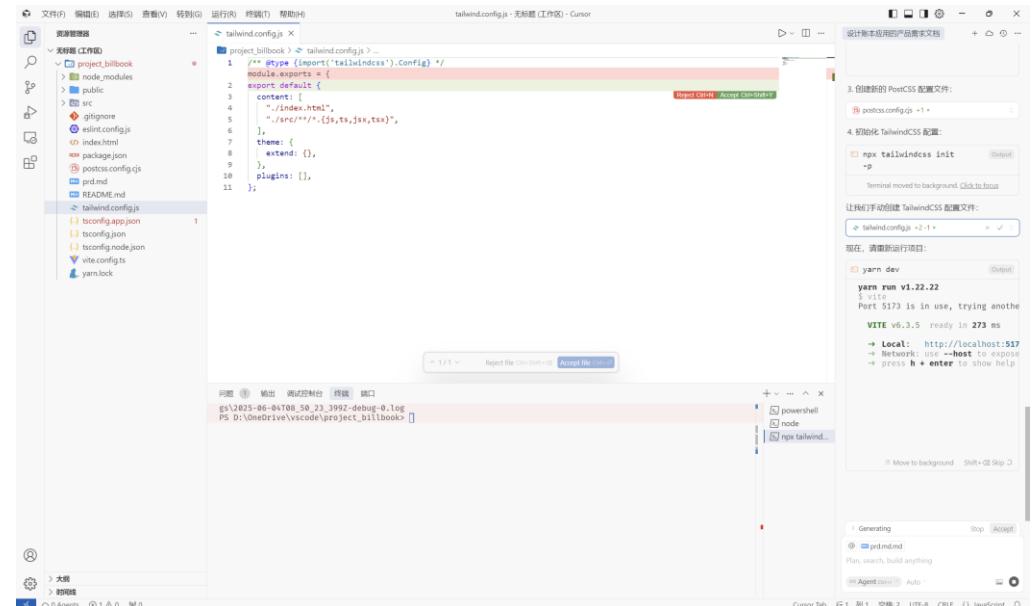
2. 然后，安装正确版本的依赖：

```
yarn add -D
tailwindcss@latest
postcss@latest
autoprefixer@latest
```

[3/4] Linking dependencies...
[4/4] Building fresh packages...

```
success Saved lockfile.
success Saved 6 new dependencies.
info Direct dependencies
├── autoprefixer@10.4.21
├── postcss@8.5.4
└── tailwindcss@4.1.8
info All dependencies
├── autoprefixer@10.4.21
├── fraction.js@4.3.7
├── normalize-range@0.1.2
├── postcss-value-parser@4.2.0
└── postcss@8.5.4
└── tailwindcss@4.1.8
Done in 1.78s.
PS D:\OneDrive\vscode\project_bill
```

3. 创建新的 PostCSS 配置文件：



问题 1 输出 调试控制台 终端 端口

```
yarn run v1.22.22
$ vite
Port 5173 is in use, trying another one...

VITE v6.3.5 ready in 273 ms

→ Local: http://localhost:5174/
→ Network: use --host to expose
→ press h + enter to show help
```

# 案例实践展示：设计一个账本应用的案例

Vibe Coding and Audit

■ Done!? <http://localhost:5175/> Still need more improvements.

问题 1 输出 调试控制台 终端 端口

```
$ vite
16:53:08 [vite] (client) Re-optimizing dependencies because lockfile has changed
Port 5173 is in use, trying another one...
Port 5174 is in use, trying another one...

VITE v6.3.5 ready in 318 ms
→ Local: http://localhost:5175/
→ Network: use --host to expose
→ press h + enter to show help
```



## 项目账本

账本名称	账本描述	新增账本
账本01	账本01描述	删除
创建时间: 2025/6/4		

■ 完全重新安装和配置 TailwindCSS，实现美化账本界面的需求。

3. 重新安装最新版本的依赖:

```
yarn add -D
tailwindcss@3.4.1
postcss@8.4.35
autoprefixer@10.4.19
PS D:\OneDrive\vscode\project_bill
er@10.4.19
yarn add v1.22.22
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
error Error: EPERM: operation not
odules\at tailwindcss\oxide-win32-x6
info Visit https://yarnpkg.com/en/
o PS D:\OneDrive\vscode\project_bill
```

Ask every time  Move to background

4. 创建新的配置文件:

- postcss.config.js +6 ✓
- tailwind.config.js +53 ✓

tailwind.config.js

```
1 export default {
2 plugins: [
3 tailwindcss(),
4 autoprefixer(),
5],
6 }
```

postcss.config.js

```
1 module.exports = {
2 plugins: [
3 require('postcss-import'),
4 require('postcss-nested'),
5 require('postcss-preset-env'),
6 require('postcss-color-function'),
7 require('postcss-preset-env').postcss,
8 require('postcss-easing'),
9 require('postcss-shapes'),
10 require('postcss-variables'),
11 require('postcss-viewport-units'),
12 require('postcss-align'),
13 require('postcss-absolute-position'),
14 require('postcss-clip-path'),
15 require('postcss-clip-rect'),
16 require('postcss-clip'),
17 require('postcss-clip-irrational'),
18 require('postcss-clip-irrational-rect'),
19 require('postcss-clip-irrational-rect-clip'),
20 require('postcss-clip-irrational-rect-clip-rect'),
21 require('postcss-clip-irrational-rect-clip-rect-clip'),
22 require('postcss-clip-irrational-rect-clip-rect-clip-rect'),
23 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip'),
24 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect'),
25 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip'),
26 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
27 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
28 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
29 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
30 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
31 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
32 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
33 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
34 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
35 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
36 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
37 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
38 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
39 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
40 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
41 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
42 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
43 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
44 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
45 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
46 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect'),
47 require('postcss-clip-irrational-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip-rect-clip'),
48];
```

4. 创建新的配置文件:

- postcss.config.js +6 ✓
- tailwind.config.js +53 ✓

5. 更新 App.tsx 的样式:

- App.tsx +83 -59 ✓

6. 更新 BookList 组件的样式:

- BookList.tsx +22 -17 ✓

7. 更新 TransactionList 组件的样式:

- TransactionList.tsx

现在让我们再次尝试启动项目:

```
yarn dev
→ Local: http://localhost:517
→ Network: use --host to expose
→ press h + enter to show help
```

Move to background Shift+Q Skip ⌂

# 案例实践展示：设计一个账本应用的案例

Vibe Coding and Audit

■ Done <http://localhost:5174/>

```
问题 1 输出 调试控制台 终端 端口
thenify@3.3.1
ts-interface-checker@0.1.13
util-deprecate@1.0.2
wrap-ansi-cjs@7.0.0
wrap-ansi@8.1.0
yaml@2.8.0
Done in 2.78s.
PS D:\OneDrive\vscode\project_billbook> yarn dev
yarn run v1.22.22
$ vite
17:03:46 [vite] (client) Re-optimizing dependencies because lockfile has changed
Port 5173 is in use, trying another one...
VITE v6.3.5 ready in 312 ms

→ Local: http://localhost:5174/
→ Network: use --host to expose
→ press h + enter to show help
```



目前我们已经实现了基本的增删功能，但还缺少编辑功能。让我们完善增删改查功能：

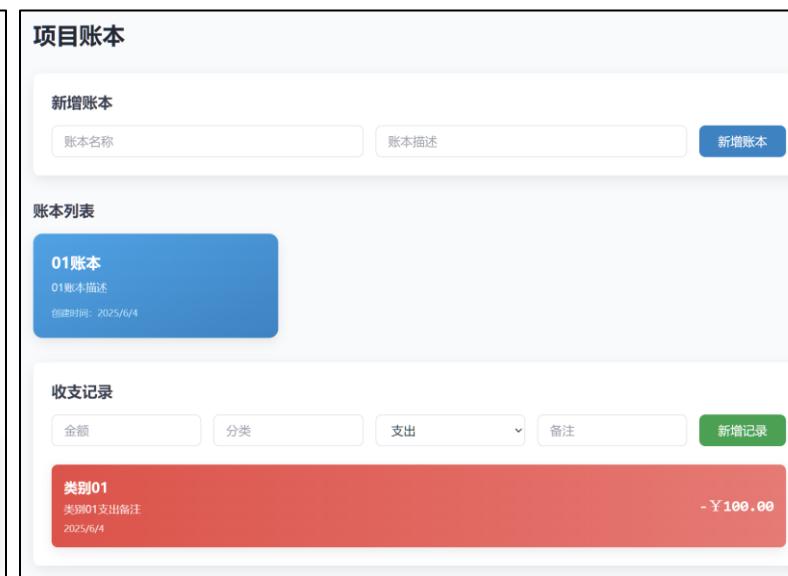
- 首先，让我们在 db.ts 中添加编辑相关的办法：

db.ts

TS db.ts +59 -1 ✓

- 更新 App.tsx，添加编辑功能：

App.tsx



# 案例实践展示：设计一个账本应用的案例

Audit and Acceptance

## ■ 改进然后无限进步 (<http://localhost:5173/>)

目前我们已经实现了基本的增删功能，但还缺少编辑功能。让我们完善增删改查功能：

1. 首先，让我们在 `db.ts` 中添加编辑相关的方  
法：

TS db.ts +59 -1 ✓

2. 更新 `App.tsx`，添加编辑功能：

TS App.tsx +136 -28 •2 ✓

> ⚠ 2 linter errors

Auto-fix ▾

现在我需要更新组件接口以支持编辑功能。让我先  
更新 `BookList` 组件：

TS BookList.tsx +30 -15 ✓

现在让我更新 `TransactionList` 组件：

TS TransactionList.tsx +67 -24 ✓

问题 1 输出 调试控制台 终端 端口

VITE v6.3.5 ready in 325 ms

→ Local: <http://localhost:5173/>  
→ Network: use `--host` to expose  
→ press `h + enter` to show help

现在用户可以：

- 编辑账本名称和描述
- 编辑交易记录的金额、类型、分类和备注
- 通过直观的 UI 界面进行所有操作
- 在编辑过程中随时取消操作

现在我已经完成了编辑功能的实现。让我总结一下  
所做的更改：

1. 在 `App.tsx` 中：

- 添加了编辑状态管理 (`editingBook` 和 `editingTransaction`)
- 实现了编辑和更新方法 (`editBook`、`updateBook`、`editTransaction`、`updateTransaction`)
- 更新了 UI 以支持编辑功能，包括编辑/取消  
按钮和表单状态切换

2. 在 `BookList.tsx` 中：

- 添加了 `onEdit` 属性到组件接口
- 添加了编辑按钮到每个账本卡片
- 优化了 UI 设计，使用更现代的卡片样式

3. 在 `TransactionList.tsx` 中：

- 添加了 `onEdit` 和 `onDelete` 属性到组件接  
口
- 将列表视图改为表格形式，提供更好的数据  
展示
- 添加了编辑和删除按钮到每行记录

## 项目账本

### 新增账本

账本名称

账本描述

新增账本

### 账本列表

账本01



账本01的描述

创建于 2025/6/4

账本02



账本02的描述

创建于 2025/6/4

### 收支记录

金额

分类

支出

备注

新增记录

日期

类型

金额

分类

备注

操作

2025/6/4

收入

1000.00

工资

工资



2025/6/4

支出

100.00

日常开销

暂无



2025/6/4

支出

300.00

年度订阅

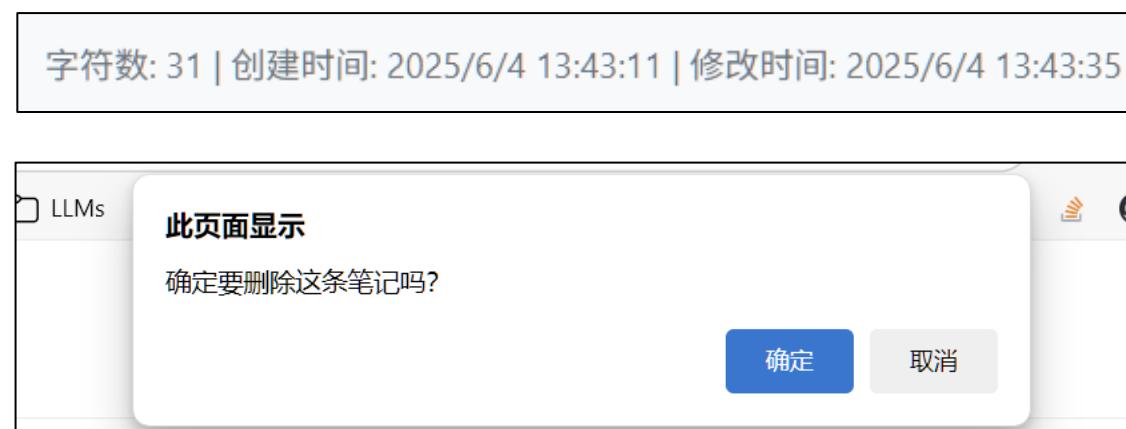
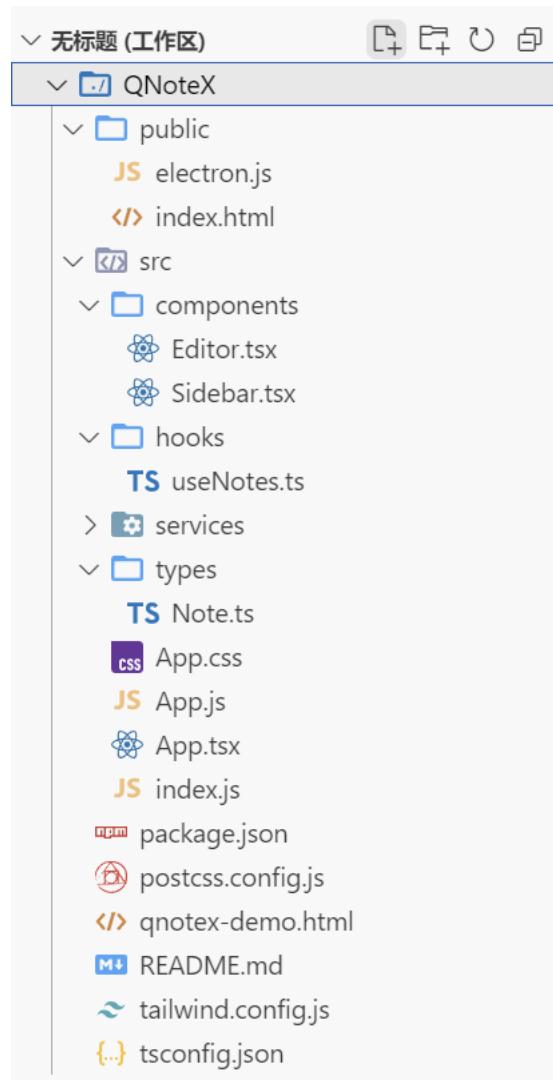
软件订阅



# 案例实践展示：写一个笔记软件

PRD -> Architect -> Vibe Coding -> Final acceptance

## ■ 一个简单的笔记软件 (<file:///D:/OneDrive/vscode/QNoteX/qnotex-demo.html>)



# 未来实践，尝试快乐的 Vibe Coding，探索 AI 世界！

Prajwal Tomar ✅ @PrajwalTomar\_ · 2月22日

New Cursor Agent is WILD.

It understands project context, follows structured plan, and builds features autonomously.

Last night, I used it to ship an MVP in just 3 hours.

Here's the exact workflow:



92 516 5,839 565万

Prajwal Tomar ✅ @PrajwalTomar\_ · 3月1日

In last 5 months, I've built 16 SaaS products for clients using Cursor.

Now, I've cracked the best AI coding workflow for Cursor.

Here's my step-by-step guide to building production-ready MVPs:



181 643 6,927 679万

Prajwal Tomar ✅ @PrajwalTomar\_ · 3月17日

Cursor + Supabase + MCP = AI-powered MVP development at its best.

This is how I'm shipping client MVPs faster, cheaper, and smarter with AI handling the backend, database, and migrations.

Let's break it down.



53 273 2,858 60万

# THE VIBE CODING SYSTEM

