基于表格长期记忆网站开发文档

1. 项目概述

网站定位为面向终端用户的长期记忆服务，提供基于表格的知识协作、流程自动化与市场化分享能力。整体采用 Next.js 架构，整合 `packages/table-memory-engine` 作为表格与流程业务 API 封装层，`packages/table-node-editor` 作为可视化节点编辑器组件来源，通过 Supabase 提供身份认证、数据持久化与实时能力，Ant Design 构建 UI 组件体系，Tailwind CSS 辅助快速样式迭代。

2. 技术栈

- Next.js 14（App Router，支持 SSG/ISR/SSR 与边缘渲染）
- TypeScript 5
- Supabase（PostgreSQL、Auth、Storage、Edge Functions）
- Ant Design 5（组件库，按需加载）
- Tailwind CSS 3（原子化样式，可与 antd 主题联动）
- Zustand/Redux Toolkit（前端状态管理，待调研）
- TanStack Query（数据获取与缓存管理）
- ESLint + Prettier + Husky（质量保证）
- Playwright/Cypress、Jest/Testing Library（端到端与组件测试）

3. 总体架构

3.1 前后端分层

- **应用层**：Next.js 页面与服务端路由（App Router，RSC + client components）
- **业务服务层**：基于 Supabase Edge Functions 与 Next.js Route Handler 实现业务 API，调用 `table-memory-engine` 暴露的仓库与流程接口
- **业务库层**：`packages/table-memory-engine` 提供表格仓库、节点流程的业务封装；`packages/table-node-editor` 提供可视化节点编辑器与流程执行器
- **基础设施层**：Supabase 数据库、存储、鉴权；表格仓库底层（table-git 核心服务，通过 memory-engine 间接访问）

3.2 部署拓扑

- Vercel 托管 Next.js（预览与生产）
- Supabase 项目提供 DB/Auth/Edge Functions
- 静态资源（流程模板、市场展示用 README）存储在 Supabase Storage 或 Git 仓库

3.3 包结构建议

```
packages/
  table-memory-engine/    # 现有表格业务封装
  table-node-editor/      # 现有节点编辑器
  web-portal/             # 新建 Next.js 应用（本项目）
```

4. 功能规划

4.1 顶部导航模块

- 首页（产品价值、核心案例、行动按钮）
- 产品
  - 记忆表格：功能介绍、场景、CTA
  - 节点编辑器：流程可视化介绍与示例
- 插件
  - 酒馆插件：插件说明、安装指引、自动更新信息
- 开发工具包
  - TableGit：面向开发者的接口文档、SDK 下载
  - 节点编辑器：集成指南与扩展规范
- 文档
  - 记忆表格：链接 `packages/table-memory-engine/docs` 与在线文档
  - 节点编辑器：链接 `packages/table-node-editor/docs`
- 市场：公开仓库/流程的浏览、搜索、筛选、详情
- 控制台：受保护区域（登录后访问）

4.2 控制台核心模块

1. **模板管理**
   - 列表与详情：展示模板元数据、关联表格仓库、节点流程集合
   - 表格属性管理：固定字段（名称、说明、更新说明、插入说明、删除说明、版本、备注）+ 自定义键值对
   - 流程绑定：从节点编辑器流程库选择或新建流程，配置执行参数
   - 权限与成员：模板级别的协作者管理
2. **对话管理**
   - 列表：显示基于模板的对话实例、状态、最近交互时间
   - 详情：左侧 AI 对话记录，右侧表格实时视图（表格编辑器），支持用户/AI 操作日志
   - 流程执行面板：展示最近命中流程、执行结果
3. **全局流程**
   - 类似模板流程页面，管理适用于所有模板的流程
   - 流程优先级、触发条件配置
4. **表格编辑器**
   - 集成第三方表格组件（如 Handsontable、AG Grid），结合 `table-memory-engine` 提供的操作 API
   - 支持 Cell-level 版本对比、差异查看、回滚
5. **节点编辑器**
   - 复用 `packages/table-node-editor` 提供的 React 组件或 iframe 嵌入方案
   - 提供流程列表、版本管理、发布接口

4.3 市场模块

- 仓库/流程分类：纯仓库、纯流程、混合
- 访问权限：仅 Clone、仅 Merge、Clone + Merge

- 标签筛选、多选过滤
- README 渲染（Markdown）
- 声明信息展示（开源、商业化条款）
- 点赞/收藏/下载量统计

5. 关键业务流程

5.1 表格仓库生命周期

1. 创建模板时新建空仓库或从市场 Clone
2. 手动编辑：用户在表格编辑器修改，保存触发 `table-memory-engine` 的 commit（角色=用户）
3. AI 修改：对话流程运行，生成 diff，自动提交（角色=AI）
4. 市场合并：用户在市场发起 Merge Flow，完成后更新模板仓库
5. 删除：模板解绑或用户主动删除仓库

5.2 对话流程

1. 用户选择模板创建对话实例
2. 对话内容经节点流程处理，对表格应用变更
3. 实时状态保存到 Supabase（对话表 + 表格快照/增量）
4. 支持回溯版本、重置对话

5.3 全局流程执行

1. 在 Supabase Edge Function 或 Next.js Route Handler 内注册流程执行入口
2. 每次对话/模板操作触发全局流程
3. 流程执行结果写入统一日志，支持可观测性

6. 数据建模草案

6.1 Supabase 表（建议）

- `users`：引用 Supabase Auth
- `organizations`：企业/团队信息（可选）
- `templates`
  - `id`, `name`, `description`, `table_repo_id`, `attributes` (JSONB), `visibility`, `created_by`
- `template_flows`
  - `id`, `template_id`, `flow_id`, `config` (JSONB), `order`
- `global_flows`
  - `id`, `name`, `config`, `is_active`
- `conversations`
  - `id`, `template_id`, `status`, `latest_message`, `table_state` (JSONB or repo ref), `created_by`
- `conversation_messages`
  - `id`, `conversation_id`, `role`, `content`, `meta`
- `conversation_commits`
  - `id`, `conversation_id`, `commit_id`, `role`, `diff_summary`
- `market_items`

    - id, type, source_repo_id, flow_id, allow_clone, allow_merge, tags, readme_md, disclaimer, stats
- audit_logs
    - 操作审计

6.2 与 table-memory-engine 的接口

- 仓库管理：createRepository, cloneRepository, deleteRepository, commitChanges, mergeRepository
- 表格属性：getTableAttributes, setTableAttributes
- 节点流程：listFlows, executeFlow, exportFlow
- 对话状态：snapshotTable, applyDiff

7. 页面与路由规划

7.1 公共页面

- / 首页
- /product/memory-table
- /product/node-editor
- /plugin/tavern
- /sdk/table-git
- /sdk/node-editor
- /docs/memory-table
- /docs/node-editor
- /market（列表）
- /market/[id]（详情）

7.2 控制台（受保护，需登录）

- /console（概览仪表盘）
- /console/templates
- /console/templates/[id]
- /console/templates/[id]/flows
- /console/conversations
- /console/conversations/[id]
- /console/global-flows
- /console/table-editor/[repoId]
- /console/node-editor/[flowId]

8. UI 与交互要点

- Ant Design 自定义主题，结合 Tailwind @apply 与 CSS variables
- 支持深色模式切换
- 采用响应式布局，桌面优先
- 表格编辑器与节点编辑器弹窗/全屏模式切换
- 市场采用卡片 + 侧边筛选器

9. 身份认证与权限

- Supabase Auth（Email/Password + OAuth）
- RBAC
  - `owner`：全部权限
  - `maintainer`：模板管理、流程管理、市场发布
  - `editor`：表格编辑、对话执行
  - `viewer`：只读
- 中间件实现路由守卫（App Router `middleware.ts`）

10. 与现有包的集成策略

- 在 `packages/table-memory-engine` 内新增对 web portal 的 API 适配层（如 hooks、服务类）
- `packages/table-node-editor` 导出 React 组件或 Web Component 供 Next.js 直接引用
- 必要时通过 `npm workspace` 链接本地包
- 为 web portal 提供统一的 TypeScript 类型定义（共用 `packages/table-memory-engine/src` 中的类型）

11. 开发计划与里程碑

1. **预研阶段（1-2 周）**
   - 调研 Next.js 14 + Supabase 集成
   - 选型表格组件库，PoC 与 table-memory-engine API 的适配
   - 评估节点编辑器在 Next.js 中嵌入方案
2. **基础设施阶段（2 周）**
   - 创建 `packages/web-portal` Next.js 工程（TurboRepo workspace）
   - 配置 Tailwind、Antd、ESLint、Prettier、Husky、Git Hooks
   - 接入 Supabase Auth、数据库迁移脚手架（Prisma Drizzle 等）
3. **核心功能阶段（4-6 周）**
   - 导航、首页、产品/插件/SDK/文档页面
   - 控制台模板、对话、全局流程、表格编辑器、节点编辑器联调
   - 市场列表、详情页
4. **集成与测试阶段（2 周）**
   - 端到端测试（Playwright/Cypress）
   - 性能优化与缓存策略
   - 可观测性埋点（Supabase Logs + 前端监控）
5. **上线准备（1 周）**
   - Vercel + Supabase 部署流程
   - 监控、告警设置
   - 文档与培训材料整理

12. 风险与缓解

- **表格组件适配复杂度**：提前选择支持虚拟渲染、定制单元格的组件；必要时开发自定义适配层。
- **节点编辑器在 SSR 环境的兼容性**：采用动态加载 `"use client"` 组件，关掉 SSR，确保浏览器 API 可用。
- **Supabase Edge Function 性能**：对高频操作使用数据库存储过程或批处理；部署前做压力测试。

- **多仓库同步一致性**：通过 `table-memory-engine` 提供的事务式操作与审计日志，确保提交可追踪。

13. 测试策略

- 单元测试：React 组件（Testing Library）、业务服务（Jest）
- 集成测试：Next.js Route Handler + Supabase Edge Functions
- 端到端测试：关键用户路径（模板创建、对话执行、市场发布）
- 回归自动化：PR pipeline 运行 lint/test/build

14. 文档与演示

- Storybook（可选）展示 UI 组件
- 集成 `doc/` 现有文档，自动部署至静态站点
- Demo 模板与对话样例，用于市场展示

15. 下一步行动

1. 确认表格组件库选型与授权条款
2. 在 `packages/` 下初始化 `web-portal` workspace
3. 根据本开发文档补充详细任务拆分（issue/backlog）