职场说话保命神器 - 设计文档

日期： 2026-02-03
项目类型： Web应用
技术栈： React + Vite

项目概述

将用户的"气话"转换为5种不同语气的职场表达版本，每个版本限制30字以内。

核心功能

- 输入一句气话，输出5种不同语气的可发送版本
- 5种语气：委婉礼貌、专业正式、友好和谐、幽默风趣、严肃认真
- 支持多个AI模型：OpenAI、通义千问、文心一言、Deepseek
- 本地保存历史记录和API密钥
- 提供示例提示帮助用户快速上手

1. 整体架构

架构类型

单页Web应用（SPA），前后端分离，**无需后端服务器**，所有逻辑在前端完成。

架构组成

前端界面层

- React构建的用户界面
- 包含输入区、卡片展示区、历史记录、API设置等模块

状态管理层

- 使用React hooks（useState、useEffect）管理应用状态
- 状态包括：用户输入、转换结果、历史记录、API设置等

数据持久层

- 使用浏览器localStorage保存用户数据
- 保存内容：API密钥、历史记录、用户偏好设置

API调用层

- 直接从前端调用各大模型厂商的HTTP API
- 支持的模型：OpenAI、通义千问、文心一言、Deepseek

工作流程

1. 用户输入"气话"并选择语气风格和AI模型
2. 前端调用选定的AI API，发送精心设计的prompt
3. AI返回5个不同语气的转换版本（每个<30字）
4. 结果以卡片形式展示，用户可点击复制
5. 记录保存到历史记录（localStorage）

部署方式

- 构建后生成静态文件
- 可部署到Vercel、Netlify、GitHub Pages等平台
- 或直接托管到对象存储

2. 核心组件设计

2.1 主容器组件 (App)

- 整体布局，包含头部、主内容区
- 管理全局状态（API设置、历史记录等）
- 提供Context API供子组件访问全局状态

2.2 输入组件 (InputSection)

功能：

- 多行文本输入框，支持最大字数提示
- 模型选择器：下拉菜单选择AI模型

- "转换"按钮：触发API调用
- 示例提示区：可点击的预设"气话"示例

示例提示：

- "这个方案根本行不通！"
- "你们怎么总是这样？"
- "我不干了！"
- "这谁写的代码？"

2.3 结果卡片组件 (ResultCards)

功能：

- 显示5个转换结果，每个卡片包含：
  - 语气标签（委婉礼貌/专业正式/友好和谐/幽默风趣/严肃认真）
  - 转换后的文本内容
  - 复制按钮（点击后显示"已复制"提示）
- 加载状态：骨架屏或加载动画
- 错误状态：友好的错误提示

卡片样式：

- 卡片式设计，每个卡片独立展示
- 清晰的视觉层次，易于阅读
- 悬停效果，提升交互体验

2.4 历史记录组件 (HistorySection)

功能：

- 时间线形式展示历史转换记录
- 每条记录显示：原始气话、转换结果、时间戳
- 支持清除历史记录
- 支持点击历史记录重新查看结果

2.5 设置组件 (SettingsPanel)

功能：

- API密钥配置面板（支持多个模型的API key）

- 模型参数调整（温度、最大长度等，可选）
- 密钥安全提示

支持的模型配置：

- OpenAI API Key
- 通义千问 API Key
- 文心一言 API Key
- Deepseek API Key

3. 数据流与Prompt工程

3.1 数据流

```
用户输入气话
  ↓
点击转换按钮
  ↓
从localStorage读取API密钥
  ↓
构造prompt（包含原始气话+5种语气要求）
  ↓
调用选定模型的API
  ↓
解析API响应（JSON格式）
  ↓
提取5个转换结果
  ↓
展示在卡片中
  ↓
保存到历史记录（localStorage）
```

3.2 Prompt设计

系统Prompt：

```
你是一个职场沟通专家。我会给你一句带有情绪的"气话"，
请将它转换为5种不同语气的职场表达版本。

要求：
1. 每个版本必须在30字以内
2. 保持礼貌和专业，避免冲突
3. 针对不同场景和对象进行优化

请按以下格式返回（JSON格式）：
{
  "委婉礼貌": "...",
  "专业正式": "...",
  "友好和谐": "...",
  "幽默风趣": "...",
  "严肃认真": "..."
}

用户输入的气话：{userInput}
```

期望响应格式：

```
{
  "委婉礼貌": "这个方案可能还需要进一步讨论和完善。",
  "专业正式": "建议对现有方案进行深入分析和优化。",
  "友好和谐": "我们可以一起探讨如何改进这个方案。",
  "幽默风趣": "这个方案给了我们很多发挥空间啊！",
  "严肃认真": "该方案存在风险，需要重新评估。"
}
```

3.3 状态管理

全局状态（Context API）：

```
{
  apiKeys: {
    openai: string,
    qianwen: string,
    wenxin: string,
    deepseek: string
  },
  history: Array<{
    id: string,
    input: string,
    results: object,
    timestamp: number,
    model: string
  }>,
  selectedModel: string,
  conversionResult: object | null,
  loading: boolean,
  error: string | null
}
```

4. 错误处理与安全性

4.1 错误处理机制

API调用失败：

- **网络错误**：提示用户检查网络连接
- **API密钥无效**：提示用户检查密钥配置
- **API额度耗尽**：提示用户充值或更换模型
- **超时**：显示超时提示，提供重试按钮

响应解析失败：

- AI返回格式不符合预期：显示原始响应供用户查看
- 某些语气缺失：显示已成功转换的部分，提示重试

输入验证：

- 空输入：提示用户输入内容

- 超长输入：限制在500字以内
- 特殊字符处理：转义可能导致解析问题的字符

4.2 安全性设计

API密钥安全：

- 密钥仅存储在用户浏览器localStorage
- 不发送到任何服务器
- 设置界面提示密钥安全注意事项
- 提供清除密钥功能

内容过滤：

- 不保存敏感个人信息
- 历史记录仅存储在本地，不上传

HTTPS要求：

- 部署时强制使用HTTPS
- API调用均通过HTTPS

4.3 用户体验优化

一键复制功能：

- 点击卡片中的复制按钮
- 显示"已复制"提示（2秒后消失）

快捷键支持：

- Ctrl+Enter：快速提交

响应式设计：

- 支持手机、平板、电脑使用
- 自适应布局

加载状态：

- 骨架屏或加载动画
- 预估加载时间提示

5. 测试与部署计划

5.1 测试策略

功能测试：

- 输入各种"气话"测试转换效果
- 测试5种语气的准确性和实用性
- 验证30字限制是否严格执行
- 测试历史记录的保存和清除
- 测试API密钥的保存和更新
- 测试示例提示的点击功能

兼容性测试：

- Chrome、Firefox、Safari、Edge浏览器测试
- 移动端响应式布局测试
- 不同API模型的测试

边界测试：

- 空输入、超长输入、特殊字符
- API异常响应处理
- 网络断开、超时场景

5.2 部署方案

构建流程：

```
npm run build  # 生成静态文件到dist目录
```

部署选项：

推荐：Vercel

- 自动CI/CD
- 免费额度
- 零配置部署

备选：Netlify

- 类似Vercel
- 也很方便

备选：GitHub Pages

- 完全免费
- 适合开源项目

自托管：

- 部署到自己的服务器
- 或托管到对象存储（如阿里云OSS）

5.3 后续优化方向

功能增强：

- 添加更多语气风格选项
- 支持自定义语气
- 导出历史记录（CSV/JSON）
- 分享功能（生成分享链接）
- 收藏功能

体验优化：

- 深色模式
- 更多主题选择
- 语音输入
- 快捷操作

6. 技术实现要点

6.1 项目初始化

```
npm create vite@latest 职场说话保命神器 -- --template react
cd 职场说话保命神器
npm install
```

6.2 核心依赖

```
{
  "react": "^18.2.0",
  "react-dom": "^18.2.0"
}
```

可选依赖：

- `axios` - HTTP请求库
- `lucide-react` - 图标库

6.3 API调用示例

```
async function callAIModel(model, apiKey, prompt) {
  const endpoints = {
    openai: 'https://api.openai.com/v1/chat/completions',
    qianwen: 'https://dashscope.aliyuncs.com/api/v1/services/aigc/text-generation/generation',
    wenxin: 'https://aip.baidubce.com/rpc/2.0/ai_custom/v1/wenxinworkshop/chat/completions',
    deepseek: 'https://api.deepseek.com/v1/chat/completions'
  };

  const response = await fetch(endpoints[model], {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${apiKey}`
    },
    body: JSON.stringify({
      model: getModelName(model),
      messages: [{ role: 'user', content: prompt }],
      temperature: 0.7
    })
  });

  return response.json();
}
```

6.4 目录结构

```
职场说话保命神器/
├── src/
│   ├── components/
│   │   ├── InputSection.jsx
│   │   ├── ResultCards.jsx
│   │   ├── HistorySection.jsx
│   │   └── SettingsPanel.jsx
│   ├── contexts/
│   │   └── AppContext.jsx
│   ├── utils/
│   │   └── api.js
│   ├── App.jsx
│   ├── App.css
│   └── main.jsx
├── docs/
│   └── plans/
│       └── 2026-02-03-职场说话保命神器-design.md
├── 用户需求.md
├── package.json
└── vite.config.js
```

7. 成功标准

项目成功的标准：

1. **功能完整性**
   - 能够成功调用至少2个不同的AI模型
   - 5种语气都能正确生成
   - 每个版本严格控制在30字以内
2. **用户体验**
   - 界面简洁美观，操作直观
   - 响应速度快（<5秒）
   - 移动端可用性良好
3. **稳定性**
   - API调用失败率 <5%
   - 错误处理友好有效

- 不丢失用户数据

4. **可维护性**
   - 代码结构清晰
   - 组件职责明确
   - 易于添加新模型

8. 风险与限制

8.1 已知风险

API依赖：

- 依赖第三方AI服务，可能面临服务中断
- API价格变动可能影响用户体验

内容质量：

- AI生成内容质量不稳定
- 可能生成不合适的转换

使用限制：

- 需要用户自己获取API密钥
- 可能阻碍部分用户使用

8.2 缓解措施

- 提供多个模型选择，降低单点依赖
- 优化prompt，提高生成质量
- 提供详细的API密钥获取指南
- 考虑提供免费试用额度（可选）

结论

本设计方案完整定义了"职场说话保命神器"的功能、架构和实现路径。采用React + Vite技术栈，无需后端服务器，支持多个主流AI模型，以卡片式界面提供优质的用户体验。

项目核心价值在于帮助职场人士将情绪化的"气话"转换为专业、得体的职场表达，避免沟通冲突，提升职场形象。