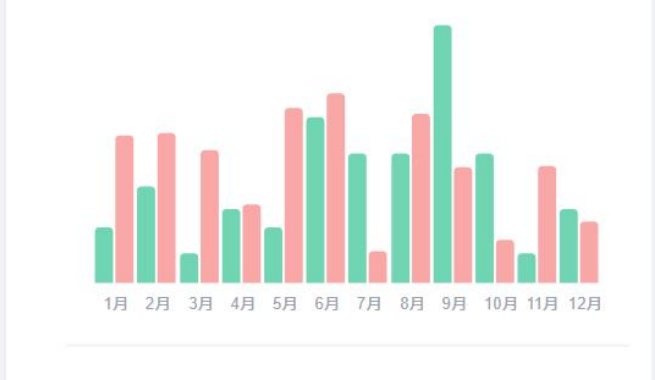

Tally 记账应用说明文档

财务踪迹，清晰可见

追踪支出收入的踪迹，明晰个人财务情况。

CappuccinoRose

React-Tally 记账应用项目说明文档

一、前言

1.1 财务管理

在数字化时代，个人财务管理变得日益重要。本项目致力于构建一个简洁、直观的记账应用，帮助用户轻松追踪收支情况，分析消费趋势，实现财务健康。通过现代化的前端技术，我们将复杂的财务数据转化为直观的可视化图表，让财务管理变得简单而有趣。

1.2 技术架构与用户体验

"React-Tally"项目采用现代前端技术栈，构建高性能的单页应用。面对传统记账应用的复杂操作和繁琐界面，我们追求"简洁高效"的设计理念，通过组件化开发和响应式设计，确保用户在任何设备上都能获得流畅的体验。

1.3 数据驱动决策

本项目可以通过年度和月度趋势图表、支出排行榜等功能，帮助用户深入了解自己的消费习惯，做出明智的财务决策。

二、项目设计

1.1 项目名称

中文名称：账迹

英文名称：React-Tally

项目代号：React-Tally-v2

1.2 网站规划

EdgeOne 部署配置本静态网站

访问地址：https://tally.edgeone.run/

1.3 配色系统

表 1 配色设计表

用途	色值 (HEX)	说明	应用场景
主色	#60a5fa	天空蓝	导航高亮、按钮文字、标题文字
背景色	#f8fafc	极淡灰	页面整体背景
卡片背景	#ffffff	纯白	内容区块背景
收入色	#34d399	绿色	收入数据展示
支出色	#f87171	粉色	支出数据展示
文字主色	#37474f	深灰	正文内容
文字次色	#64748b	中灰	辅助说明文字
分割线	#e2e8f0	浅灰	卡片分割、列表分割

1.4 技术选择

整体技术选型强调"轻量、高性能、易维护"

（1）核心框架：React 18

（2）状态管理：Redux Toolkit

（3）路由管理：React Router

（3）样式方案：CSS Modules + CSS 变量

（4）数据服务：json-server（本地 mock）

1.5 项目功能

（1）新增账单页

类型选择（收入/支出）

金额输入

日期选择

分类选择

保存功能

（2）年度统计页

年度收支概览（收入/支出/结余）

月度收支趋势图表

支出排行榜 TOP5

年度数据可视化

（3）月度统计页

月份选择器

月度收支概览

日账单列表

月度趋势分析

1.6 项目文件结构

```
project-root/
├── public/
│   └── index.html                # HTML 模板
├── server/
│   └── db.json                   # Mock 数据 (json-server --port 8000)
├── src/
│   ├── styles/                   # 全局样式
│   │   ├── variables.css         # CSS 变量（颜色、字体、间距）
│   │   ├── reset.css             # 样式重置
│   │   └── index.css             # 统一导出
│   ├── pages/                    # 页面组件
│   │   ├── Layout/               # 布局组件
│   │   │   ├── index.jsx
│   │   │   └── index.css
│   │   ├── New/                  # 新增账单页
│   │   │   ├── components/
│   │   │   │   ├── Header/
│   │   │   │   │   ├── Header.jsx
│   │   │   │   │   └── Header.css
│   │   │   │   ├── BillForm/
│   │   │   │   │   ├── BillForm.jsx
│   │   │   │   │   └── BillForm.css
│   │   │   │   └── CategoryGrid/
│   │   │   │       ├── CategoryGrid.jsx
│   │   │   │       └── CategoryGrid.css
│   │   │   ├── index.jsx         # 主组件 (已集成 Toast 与 Redux)
│   │   │   └── index.css         # 主样式 (包含按钮渐变色)
│   │   ├── Year/                 # 年度统计页
│   │   │   ├── index.jsx         # 主页面（只负责组合组件）
│   │   │   ├── index.css         # 主样式（只写页面级布局）
│   │   │   ├── components/
```

```
│   │   │   │   ├── YearHeader/              # 年度选择 + 标题
│   │   │   │   │   ├── YearHeader.jsx
│   │   │   │   │   └── YearHeader.css
│   │   │   │   ├── YearSummary/             # 年度概览卡片（收入/支出/结余）
│   │   │   │   │   ├── YearSummary.jsx
│   │   │   │   │   └── YearSummary.css
│   │   │   │   ├── YearTrend/               # 月度趋势柱状图
│   │   │   │   │   ├── YearTrend.jsx
│   │   │   │   │   └── YearTrend.css
│   │   │   │   └── YearRank/                # 支出排行 TOP5
│   │   │   │       ├── YearRank.jsx
│   │   │   │       └── YearRank.css
│   │   │   └── hooks/
│   │   │       └── useYearStats.js          # 年度数据计算逻辑
│   │   └── Month/                           # 月度统计页
│   │       ├── index.jsx
│   │       ├── index.css
│   │       └── components/
│   │           ├── MonthSelector/           # 月份选择器
│   │           │   ├── MonthSelector.jsx
│   │           │   └── MonthSelector.css
│   │           ├── BillOverview/            # 收支概览
│   │           │   ├── BillOverview.jsx
│   │           │   └── BillOverview.css
│   │           ├── DailyBill/               # 日账单
│   │           │   ├── DailyBill.jsx
│   │           │   └── DailyBill.css
│   │           └── hooks/                   # 自定义 Hook
│   │               └── useBillStats.js
│   ├── router/                              # 路由配置
│   │   └── index.js
│   ├── store/                               # Redux 状态管理
```

```
│   │   ├── index.js              # store 配置
│   │   └── modules/
│   │       └── billStore.js      # 账单相关状态 (端口:8000)
│   ├── components/               # 公共组件（可复用）
│   │   ├── Icon/
│   │   │   ├── index.jsx
│   │   │   └── index.css
│   │   └── Toast/                # Toast 弹窗组件
│   │       ├── Toast.jsx
│   │       └── Toast.css
│   ├── utils/                    # 工具函数
│   │   ├── index.js
│   │   ├── iconMap.js
│   │   ├── billCategory.js
│   │   └── format.js
│   └── index.js                  # 应用入口
├── .gitignore
├── ...
└── README.md
```

1.7 资源与数据

静态资源（样式）存放于 src/styles/ 目录下

项目数据（账单记录）存储在 server/db.json 中

三、项目实现

1. 新增账单页

图 1 新增账单页（支出）

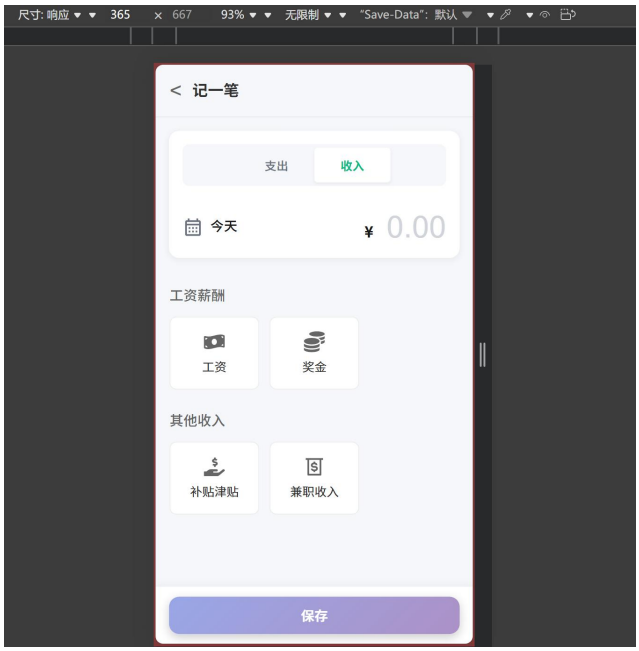

图 2 新增账单页（收入）

1.1 组件结构

Header：账单页头部组件

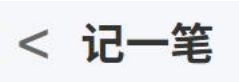

图 3 返回按钮

BillForm：账单表单组件

图 4 账单表单

CategoryGrid：分类选择网格

图 5 分类选择网格（支出）

图 6 分类选择网格（收入）

Toast：提示组件

1.2 关键实现细节

```
try {
        const response = await fetch('http://localhost:8000/ka', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify(billData),
        });

        if (response.ok) {
          showToast("账单保存成功！", "success");

          // 保存成功后，触发 Redux Action 重新获取列表
          dispatch(getBillList());

          // 重置表单
          setAmount("");
          setSelectedCategory(null);
        } else {
          throw new Error('服务器响应错误');
        }
      } catch (error) {
        console.error("保存失败:", error);
        showToast("保存失败，请确认 json-server 已在 8000 端口启动！",
"error");
      }
  };
```

2. 年度统计页

图 7 年度统计页（小屏幕）

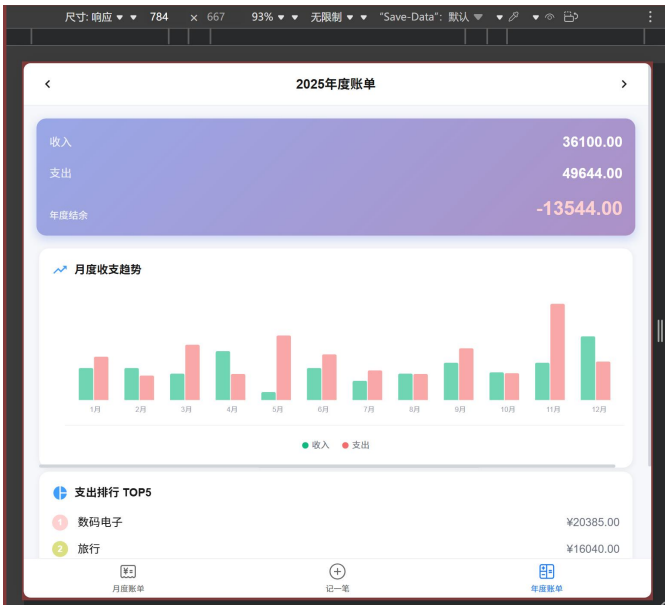

图 8 年度统计页（中屏幕）

图 9 年度统计页（大屏幕）

2.1 实现思路

YearHeader：年度选择和标题

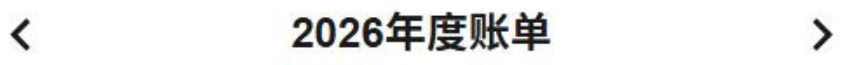

图 10 年度选择和标题

YearSummary：年度收支概览

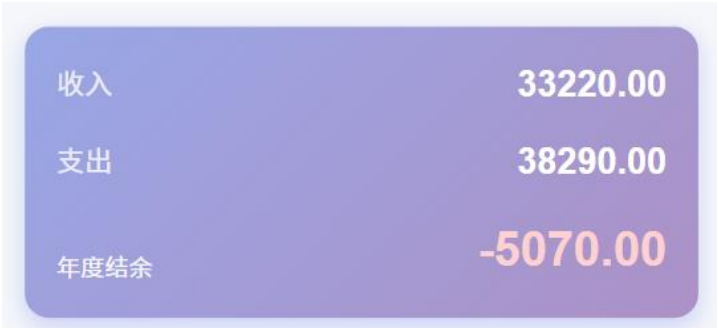

图 11 年度收支概览

YearTrend：月度趋势图表

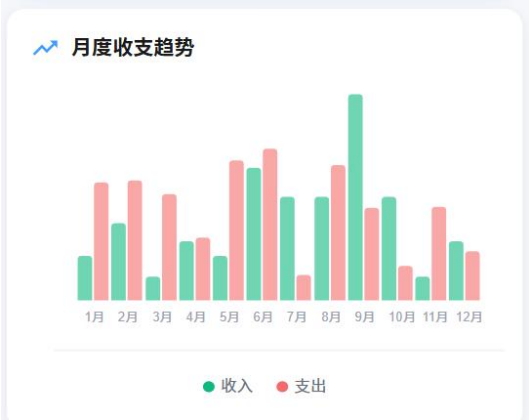

图 12 月度趋势图表

YearRank：支出排行榜

图 13 支出排行榜

2.2 关键实现细节

```
// 年度趋势图表响应式设计
.year-card {
    padding: var(--spacing-lg);
    background-color: var(--color-bg-card);
    border-radius: var(--radius-lg);
    box-shadow: var(--shadow-card);
    overflow: auto;
```

```
    -webkit-overflow-scrolling: touch;
    width: 95%;
    margin: .05rem auto;
}
/* 图表容器 */
.chart-container {
    display: flex;
    justify-content: space-between;
    align-items: flex-end;
    width: calc(100% - 10px);
    margin: var(--spacing-lg);
    padding: var(--spacing-lg);
    border-bottom: 1px solid var(--color-divider);
    min-width: max-content;
    white-space: nowrap;
}
```

3. 月度统计页

3.1 实现思路

MonthSelector：月份选择器

BillOverview：月度收支概览

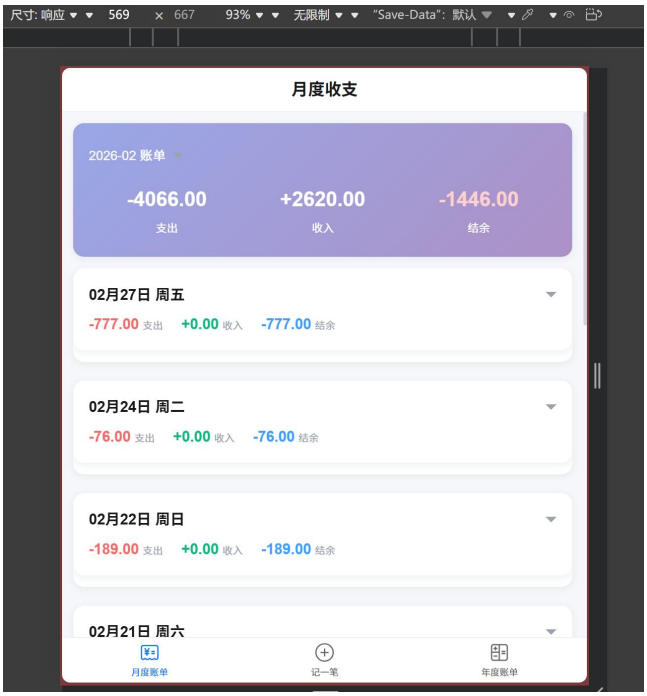

图 14 月度收支页

DailyBill：日账单列表

图 15 日账单列表

3.2 关键实现细节

```
// 月份选择器逻辑
const [selectedMonth, setSelectedMonth] = useState(new Date().getMonth() + 1);
const handleMonthChange = (month) => {
  setSelectedMonth(month);
  // 触发数据重新加载
  dispatch(getMonthBillList(month));
};
/**
 * 分类数据配置
 */
export const billCategoryData = {
  pay: [
    {
      type: 'food',
      name: '餐饮美食',
      list: [
        { type: 'food', name: '餐饮' },
        { type: 'drinks', name: '酒水饮料' },
        { type: 'dessert', name: '甜品零食' },
      ]
    },
    {
      type: 'transportation',
      name: '交通出行',
      list: [
        { type: 'transportation', name: '公共交通' },
        { type: 'longdistance', name: '长途交通' },
      ]
    },
    {
```

```
      type: 'travel',
      name: '旅行度假',
      list: [
        { type: 'travel', name: '旅行' },
      ]
    },
    {
      type: 'shopping',
      name: '购物消费',
      list: [
        { type: 'shopping', name: '购物' },
        { type: 'clothes', name: '服饰鞋包' },
        { type: 'electronics', name: '数码电子' },
      ]
    },
    {
      type: 'entertainment',
      name: '休闲娱乐',
      list: [
        { type: 'entertainment', name: '休闲娱乐' },
      ]
    },
    {
      type: 'daily',
      name: '居家生活',
      list: [
        { type: 'supplies', name: '日常用品' },
        { type: 'books', name: '图书学习' },
        { type: 'health', name: '医疗健康' },
      ]
    },
  ],
  income: [
```

```
    {
      type: 'salary',
      name: '工资薪酬',
      list: [
        { type: 'salary', name: '工资' },
        { type: 'bonus', name: '奖金' },
      ]
    },
    {
      type: 'other',
      name: '其他收入',
      list: [
        { type: 'allowance', name: '补贴津贴' },
        { type: 'parttime', name: '兼职收入' },
      ]
    },
  ]
};

/**
 * 生成 useFor -> 中文名 的映射表
 */
export const billTypeToName = {};

Object.keys(billCategoryData).forEach(key => {
  billCategoryData[key].forEach(group => {
    group.list.forEach(item => {
      billTypeToName[item.type] = item.name;
    });
  });
});
/**
 * 获取分类中文名
```

```
 */
export const getCategoryName = (useFor) => {
   return billTypeToName[useFor] || useFor;
};
// 引入 react-icons 中的图标组件
// Fa: Font Awesome | Md: Material Design | Bi: Bootstrap Icons
import {
   FaUtensils, FaWineBottle, FaIceCream,          // 餐饮
   FaBus, FaPlane, FaSuitcase,                    // 交通旅行
   FaShoppingBag, FaTshirt,                       // 购物
   FaGamepad, FaHome, FaBook, FaFirstAid,         // 生活娱乐
   FaMoneyBillWave, FaCoins, FaHandHoldingUsd, // 收入
} from 'react-icons/fa';
import { MdOutlineElectricalServices } from 'react-icons/md'; // 数码电子
import { BiMoneyWithdraw } from 'react-icons/bi';                  // 兼职

// 映射关系：type -> 图标组件
export const iconMap = {
   // === 支出 - 餐饮美食 ===
   food: FaUtensils,                // 餐饮
   drinks: FaWineBottle,            // 酒水饮料
   dessert: FaIceCream,             // 甜品零食
   // === 支出 - 交通出行 ===
   transportation: FaBus,           // 公共交通
   longdistance: FaPlane,           // 长途交通
   // === 支出 - 旅行度假 ===
   travel: FaSuitcase,              // 旅行
   // === 支出 - 购物消费 ===
   shopping: FaShoppingBag,      // 购物
   clothes: FaTshirt,               // 服饰鞋包
   electronics: MdOutlineElectricalServices, // 数码电子
   // === 支出 - 休闲娱乐 ===
   entertainment: FaGamepad,     // 休闲娱乐
```

```
  // === 支出 - 居家生活 ===
  supplies: FaHome,                  // 日常用品
  books: FaBook,                     // 图书学习
  health: FaFirstAid,                // 医疗健康
  // === 收入 ===
  salary: FaMoneyBillWave,           // 工资
  bonus: FaCoins,                    // 奖金
  allowance: FaHandHoldingUsd,       // 补贴津贴
  parttime: BiMoneyWithdraw,         // 兼职收入
};
// 获取图标组件的函数
export const getIconComponent = (type) => {
  return iconMap[type] || FaShoppingBag; // 默认返回购物袋图标
};
```

四、总结

4.1 项目难点

在开发"React-Tally"的过程中，主要面临了以下技术挑战：

（1）静态部署环境下的数据持久化

问题：部署到 EdgeOne 后无法访问后端服务，需要重新设计数据存储方案。

解决：采用 localStorage 替代后端存储，确保数据在浏览器端持久化。

（2）复杂图表的响应式设计

问题：月度趋势图表在小屏幕下需要横向滚动，同时保持良好的视觉效果。

解决：使用 CSS Grid 和 Flexbox 实现响应式布局，结合 overflow: auto 实现横向滚动。

（3）Redux 状态管理优化

问题：确保数据在页面间正确传递和更新。

解决：使用 Redux Toolkit 的异步 thunk 和 selector 优化状态管理。

4.2 项目亮点

（1）轻量级架构设计

采用 Redux Toolkit 进行状态管理，代码结构清晰，易于维护。

（2）优秀的数据可视化

通过柱状图和排行榜直观展示财务数据，帮助用户快速理解消费趋势。

（3）响应式用户体验

在不同设备上都能获得良好的使用体验，小屏幕下支持横向滚动查看完整数据。

（4）组件化思维与代码复用

将通用组件提取为独立模块，提高代码复用性和可维护性。

4.3 心得体会

通过这个记账项目的开发，我体会到：前端开发的难点往往在于平衡功能实现和用户体验，响应式设计是现代 Web 应用不可或缺的部分，数据可视化能够极大地提升用户对数据的理解，通过简洁直观的界面设计，让财务管理变得轻松愉快。