

گزارش پروژه

سیستم مدیریت هزینه‌های مشترک

Express.js + React + Vite

نام و نام خانوادگی: رضیه کرمی

شماره دانشجویی: 40138823

مخزن گیت‌هاب: github.com/rzyegillsu/Internet-Engineering-Exes.git

معرفی پروژه

این پروژه یک وب‌اپلیکیشن کامل برای مدیریت هزینه‌های مشترک است که برای دانشجویان، هم‌اتاقی‌ها و گروه‌های دوستان طراحی شده است. این سیستم به کاربران امکان می‌دهد هزینه‌های مشترک خود را ثبت کرده، سهم هر فرد را محاسبه کنند و پیشنهادات تسویه حساب دریافت نمایند.

ثبت هزینه

ثبت سریع و آسان هزینه‌ها با مشخص کردن پرداخت‌کننده، مبلغ، دسته‌بندی و افراد درگیر

محاسبه خودکار

محاسبه دقیق موجودی و بدھی هر فرد به صورت خودکار و لحظه‌ای

پیشنهاد تسویه

رابط کاربری مدرن

طراحی زیبا و کاربرپسند با پشتیبانی کامل از زبان فارسی

معماری پروژه

این پروژه بر اساس معماری REST API طراحی شده و از الگوی Client-Server برای ارتباط بین فرانت‌اند و بک‌اند استفاده می‌کند.

بک‌اند (Backend)

Node.js ✓

Express.js ✓

CORS ✓

UUID ✓

فرانت‌اند (Frontend)

React 18 ✓

Vite ✓

CSS3 ✓

Fetch API ✓

ابزارهای توسعه

Nodemon ✓

VS Code ✓

Git ✓

npm ✓

ساختار فایل‌ها

```
financial_system/
  └── backend/
    ├── package.json
    └── server.js
  └── src/
    ├── app.js
    └── controllers/
      ├── expenseController.js
      ├── data/
      │   └── store.js
      └── routes/
        ├── expenseRoutes.js
        └── services/
          ├── balanceService.js
          └── utils/
            └── settlement.js
  └── frontend/
    ├── package.json
    ├── index.html
    ├── vite.config.js
    └── src/
      ├── App.jsx
      ├── App.css
      ├── main.jsx
      └── components/
        ├── ExpenseForm.jsx
        ├── ExpenseList.jsx
        └── BalanceSummary.jsx
      └── SettlementList.jsx
    └── services/
      └── api.js
  └── README.md
```

REST API Documentation

بکاند چهار endpoint اصلی را ارائه می‌دهد:

ثبت هزینه جدید

api/expenses/

POST

دریافت لیست همه هزینه‌ها

api/expenses/

GET

حذف یک هزینه

api/expenses/:id/

DELETE

دریافت موجودی و پیشنهادات تسویه

api/balance/

GET

نمونه درخواست POST /api/expenses

```
{
  "payer": "مواد غذایی رضیه",
  "amount": 150000,
  "category": "موجودی",
  "participants": [
    {
      "member": "الهام",
      "balance": 100000,
      "description": "خرید برای شام"
    },
    {
      "member": "فاطمه",
      "balance": 50000,
      "description": "خرید برای شام"
    },
    {
      "member": "محمد",
      "balance": 50000,
      "description": "خرید برای شام"
    }
  ],
  "description": "خرید برای شام"
}
```

نمونه پاسخ GET /api/balance

```
{
  "balances": [
    {
      "member": "الهام",
      "balance": -50000
    },
    {
      "member": "رضیه",
      "balance": 150000
    },
    {
      "member": "فاطمه",
      "balance": -50000
    },
    {
      "member": "محمد",
      "balance": -50000
    }
  ],
  "settlements": [
    {
      "from": "الهام",
      "to": "رضیه",
      "amount": 50000
    },
    {
      "from": "فاطمه",
      "to": "رضیه",
      "amount": 50000
    },
    {
      "from": "محمد",
      "to": "رضیه",
      "amount": 50000
    }
  ]
}
```

جریان کار سیستم

فرآیند ثبت و محاسبه هزینه

- کاربر فرم ثبت هزینه را در فرانتاند پر می کند
- داده ها را با Fetch API به بکاند ارسال می کند
- درخواست را دریافت و اعتبارسنجی می کند
- هزینه را با UUID منحصر به فرد ایجاد می کند
- داده در Store (آرایه حافظه) ذخیره می شود
- محاسبات را انجام می دهد Balance Service
- بهینه ترین تسویه ها را پیشنهاد می دهد Settlement Utility
- نتایج به فرانتاند برگردانده می شود
- را با داده های جدید به روزرسانی می کند React UI

جزئیات پیاده سازی

الگوریتم محاسبه موجودی

سیستم با استفاده از یک الگوریتم هوشمند، موجودی هر فرد را بر اساس هزینه های ثبت شده محاسبه می کند:

- برای هر هزینه، مبلغ کل بر تعداد افراد درگیر تقسیم می شود
- سهم هر فرد از موجودی او کسر می شود
- مبلغ کامل به حساب پرداخت کننده اضافه می شود
- موجودی های نهایی با دقت دو رقم اعشار نرم افزار سازی می شوند

الگوریتم پیشنهاد تسويه (Greedy Settlement)

برای به حداقل رساندن تعداد تراکنش های تسويه، از یک الگوریتم حریصانه استفاده می شود:

- افراد به دو دسته طلبکاران و بدھکاران تقسیم می شوند

- هر دو لیست بر اساس مبلغ نزولی مرتب می‌شوند
- بزرگترین بدهی با بزرگترین طلب جفت‌سازی می‌شود
- این فرآیند تا تسویه کامل همه حساب‌ها ادامه می‌یابد

مدیریت خطأ و اعتبارسنجی

سیستم شامل لایه‌های مختلف اعتبارسنجی است:

- فرانت‌اند: بررسی فیلدهای ضروری و مقادیر معتبر قبل از ارسال
- بکاند: اعتبارسنجی دوباره داده‌ها در Controller
- مدیریت خطأ: Middleware مرکزی برای مدیریت و فرموده‌ی خطاها
- پیام‌های فارسی: خطاها به زبان فارسی برای کاربر نمایش داده می‌شوند

نمایش عملکرد سیستم



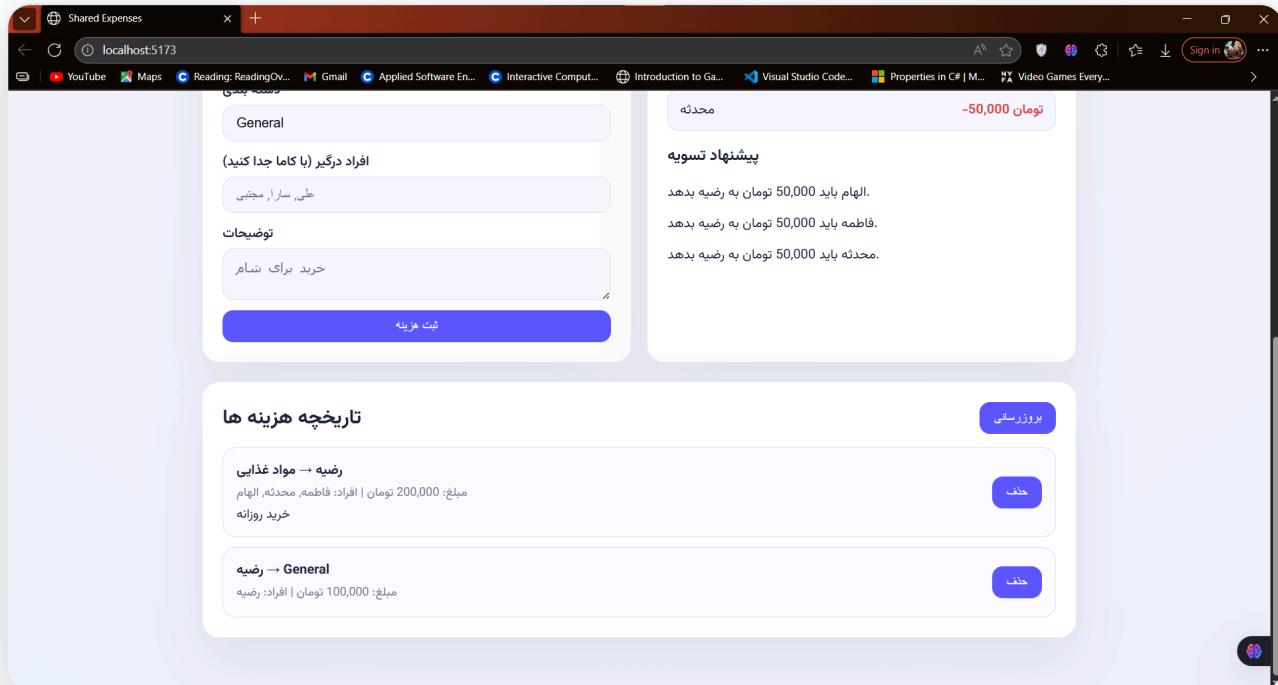
ویژگی‌های رابط کاربری

- طراحی دوستونی: Layout دو ستونه برای استفاده بهینه از فضا
- فرم هوشمند: اعتبارسنجی Real-time و پیام‌های راهنمایی

• نمایش رنگی: استفاده از رنگ سبز برای طلبکاران و قرمز برای بدھکاران

• سازگار با تمام اندازه صفحه‌نمایش‌ها: **Responsive**

• فونت فارسی: استفاده از فونت Vazirmatn برای خوانایی بهتر



مثال کاربردی

فرض کنید چهار نفر (الهام، رضیه، فاطمه، محدثه) با هم هم خانه هستند:

1. هزینه اول: رضیه 200,000 تومان برای مواد غذایی پرداخت کرد → سهم هر نفر 50,000 تومان

2. هزینه دوم: رضیه 100,000 تومان برای دسته بندی General پرداخت کرد → فقط یک نفر در لیست هست پس سهم شخص 100,000 تومان

3. پیشنهاد تسویه: هر نفر 50,000 تومان به رضیه بدهد

نصب و راه‌اندازی

پیش‌نیازها

Node.js 16 یا بالاتر •

yarn یا npm •

• مرورگر مدرن (Chrome, Firefox, Edge)

اجرای بک‌اند

```
cd financial_system/backend npm install npm run dev
```

سرور روی پورت 5000 اجرا می‌شود: <http://localhost:5000>

اجرای فرانت‌اند

```
cd financial_system/frontend npm install npm run dev
```

اپلیکیشن روی پورت 5173 اجرا می‌شود: <http://localhost:5173>

امکانات قابل توسعه

این پروژه پایه‌ای قوی برای توسعه‌های آینده فراهم می‌کند:

احراز هویت

افزودن سیستم ثبت‌نام و ورود کاربران با JWT

دیتابیس

اتصال به PostgreSQL یا MongoDB برای ذخیره دائمی

نمودارها

WebSocket

به روزرسانی لحظه‌ای برای همه کاربران با Socket.io

PWA

تبدیل به Progressive Web App برای نصب روی موبایل

چند ارزی

پشتیبانی از ارزهای مختلف و تبدیل خودکار

اعلان‌ها

ارسال ایمیل یا SMS برای یادآوری بدهی‌ها

گزارش‌گیری

صادرات گزارش‌ها به فرمت PDF یا Excel

نتیجه‌گیری

این پروژه یک نمونه کامل از یک Full-Stack Web Application است که مفاهیم کلیدی برنامه‌نویسی وب را پوشش می‌دهد:

- ✓ طراحی و پیاده‌سازی REST API با Express.js •
- ✓ ساخت رابط کاربری تعاملی با React •
- ✓ مدیریت وضعیت در فرانت‌اند •
- ✓ ارتباط Client-Server با Fetch API •
- ✓ اعتبارسنجی داده‌ها در دو لایه •
- ✓ پیاده‌سازی الگوریتم‌های محاسباتی •
- ✓ طراحی UI/UX مدرن و کاربرپسند •
- ✓ کد تمیز و قابل نگهداری با معماری Modular •

این پروژه نه تنها یک ابزار کاربردی برای مدیریت هزینه‌های روزمره است، بلکه یک منبع آموزشی عالی برای یادگیری توسعه وب مدرن نیز محسوب می‌شود.

مراجع و منابع

-
- [Express.js Documentation](#) •
 - [React Documentation](#) •
 - [Vite Documentation](#) •
 - [Fetch API - MDN](#) •
 - [RESTful API Design](#) •

گزارش پروژه سیستم مدیریت هزینه‌های مشترک

توسعه یافته با React و Express.js