🎯 گزارش پروژه

سیستم مدیریت هزینه‌های مشترک

Express.js + React + Vite

نام و نام خانوادگی: رضیه کرمی

شماره دانشجویی: 40138823

مخزن گیت‌هاب(پوشه‌ی financial_system): 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 تسویه‌حساب دریافت نمایند.

💰 ثبت هزینه

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

📊 محاسبه خودکار

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

🎯 پیشنهاد تسویه

ارائه بهینه‌ترین راه‌های تسویه حساب با کمترین تعداد تراکنش

📱 رابط کاربری مدرن

طراحی زیبا و کاربرپسند با پشتیبانی کامل از زبان فارسی

🏗️ معماری پروژه

این پروژه بر اساس معماری **Client-Server** طراحی شده و از الگوی **REST API** 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:

متد	مسیر	توضیحات
POST	api/expenses/	ثبت هزینه جدید
GET	api/expenses/	دریافت لیست همه هزینه‌ها
DELETE	api/expenses/:id/	حذف یک هزینه
GET	api/balance/	دریافت موجودی و پیشنهادات تسویه

📤 نمونه درخواست POST /api/expenses


```
{ "payer": "رضیه", "amount": 150000, "category": "مواد غذایی",
"participants": ["محدثه" ,"فاطمه" ,"رضیه" ,"الهام"], "description":
"خرید برای شام" }
```


📥 نمونه پاسخ GET /api/balance


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


⚙️ جریان کار سیستم

🔄 فرآیند ثبت و محاسبه هزینه

1. کاربر فرم ثبت هزینه را در فرانت‌اند پر می‌کند
2. React داده‌ها را با Fetch API به بک‌اند ارسال می‌کند
3. Express درخواست را دریافت و اعتبارسنجی می‌کند
4. Controller هزینه را با UUID منحصر به فرد ایجاد می‌کند
5. داده در Store (آرایه حافظه) ذخیره می‌شود
6. Balance Service محاسبات را انجام می‌دهد
7. Settlement Utility بهینه‌ترین تسویه‌ها را پیشنهاد می‌دهد
8. نتایج به فرانت‌اند برگردانده می‌شود
9. React UI را با داده‌های جدید به‌روزرسانی می‌کند

💻 جزئیات پیاده‌سازی

🎯 الگوریتم محاسبه موجودی

سیستم با استفاده از یک الگوریتم هوشمند، موجودی هر فرد را بر اساس هزینه‌های ثبت شده محاسبه می‌کند:

- برای هر هزینه، مبلغ کل بر تعداد افراد درگیر تقسیم می‌شود
- سهم هر فرد از موجودی او کسر می‌شود
- مبلغ کامل به حساب پرداخت‌کننده اضافه می‌شود
- موجودی‌های نهایی با دقت دو رقم اعشار نرمال‌سازی می‌شوند

🔀 الگوریتم پیشنهاد تسویه (Greedy Settlement)

برای به حداقل رساندن تعداد تراکنش‌های تسویه، از یک الگوریتم حریصانه استفاده می‌شود:

- افراد به دو دسته طلبکاران و بدهکاران تقسیم می‌شوند

- 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 خطا و اعتبارسنجی

سیستم شامل لایه‌های مختلف اعتبارسنجی است:

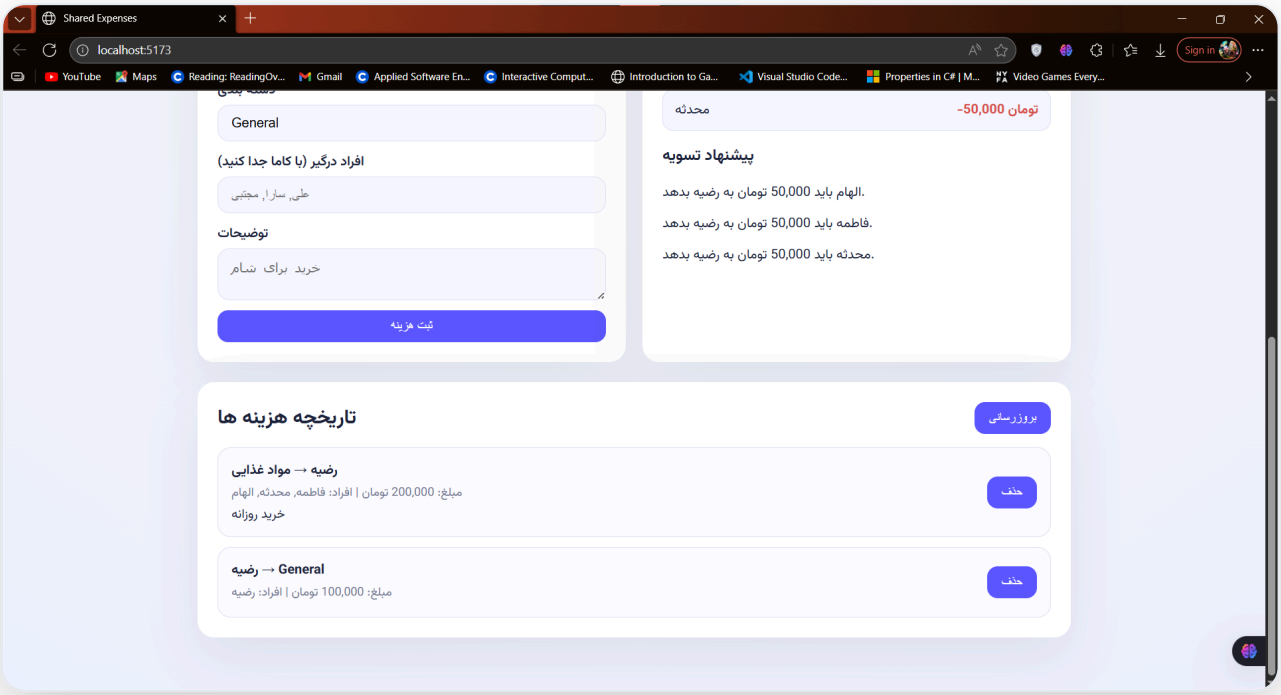
- **فرانت‌اند:** بررسی فیلدهای ضروری و مقادیر معتبر قبل از ارسال
- **بک‌اند:** اعتبارسنجی دوباره داده‌ها در Controller
- **مدیریت خطا:** Middleware مرکزی برای مدیریت و فرمت‌دهی خطاها
- **پیام‌های فارسی:** خطاها به زبان فارسی برای کاربر نمایش داده می‌شوند

نمایش عملکرد سیستم

ویژگی‌های رابط کاربری

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

- **نمایش رنگی:** استفاده از رنگ سبز برای طلبکاران و قرمز برای بدهکاران
- **Responsive:** سازگار با تمام اندازه صفحه‌نمایش‌ها
- **فونت فارسی:** استفاده از فونت Vazirmatn برای خوانایی بهتر



💡 مثال کاربردی

فرض کنید چهار نفر (الهام، رضیه، فاطمه، محدثه) با هم هم‌خانه 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 نسخه 16 یا بالاتر
- npm یا yarn

- 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 MongoDB یا PostgreSQL برای ذخیره دائمی

📊 نمودارها

افزودن نمودارهای دایره‌ای و میله‌ای با Chart.js

⚡ WebSocket

به‌روزرسانی 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