به نام خدا

پروژه کامپیوتری شماره سه **مرضیه باقری‌نیا امیری / ۸۱۰۱۹۷۶۸۲**

❖ ساختار پروژه

بنده این پروژه را از ابتدا در truffle اجرا کردم. بنابراین مراحل انجام کار در truffle را توضیح خواهم داد:

- ابتدا truffle را نصب کرده و یک فولدر برای پروژه می سازیم.
- برای تشکیل یک box برای پروژه از دستور زیر استفاده می کنیم:

truffle unbox pet-shop

- با این کار تمامی فولدرهای لازم برای اجرای یک پروژه DAPP اعم از Src، Migration، Contracts و ... را برای ما می‌سازد و یک پروژه اولیه ساده هم در آن موجود است.
- فایل .sol خود را در فولدر contracts می‌سازیم. توضیحات مربوط به این فایل در ادامه آمده است:

a) در این فایل، ابتدا یک structure برای creditorها می‌سازیم و هر creditor یک آدرس و یک مقدار قرضی که داده است، دارد.

b) سپس دو ساختارداده map می‌سازیم که یک (debtors) با گرفتن آدرس یک فرد، لیست کسانی را که از آن‌ها قرض گرفته است را باز می گرداند و دیگری (areDebtors) با گرفتن آدرس فرد مشخص می‌کند که آیا او اصلاً بدهی دارد یا خیر!

c) سپس مطابق با صورت پروژه تابع lookup را داریم که با گرفتن دو آدرس بدهکار و قرض دهنده، مشخص می‌کند که بدهکار به فرد قرض دهنده چه میزان بدهی دارد. روند کار به این صورت است که ابتدا چک می‌کنیم اصلاً فردی که آن را به عنوان بدهکار معرفی کردیم، بدهی دارد یا نه و سپس با استفاده از مپ debtors لیست کسانی را که او از آن‌ها قرض گرفته است می‌گیریم و در میان آن‌ها اگر کسی با مشخصات (آدرس) فرد قرض دهنده یافتیم،مقداری که قرض داده است (amt_owed) را بر می‌گردانیم.

d) یک تابع internal با نام adjustDebt داریم، که هر تراکنشی که رخ می دهد، با استفاده از آن مقدار amt_owed را برای قرض دهنده به روزرسانی می‌کنیم.

e) آخرین تابع هم که add_IOU است؛ که در آن اگر فرد بدهکار، قبلاً بدهی نداشت، که یک Creditor جدید می‌سازیم و مقدار قرض را برای آن ست می‌کنیم و اگر هم قبلاً قرضی گرفته شده بود که با استفاده از تابع adjustDebt مقدار بدهی را برای creditor به روز می‌کنیم.

f) در قسمت آخر این تابع هم تمهیداتی برای اینکه دو به وجود نیاید اندیشیده شده!

- فایل deploy_contracts.js_2 را برای انجام عملیات migration در فولدر migrations می‌سازیم!
- در مرحله بعد، فایل style.css را درون فولدر src/css قرار می‌دهیم.
- و سپس فایل index.html را در فولد src کپی می‌کنیم. البته با توجه به عوض شدن ساختار کلی پروژه تغییراتی در فایل‌هایی که در فایل به آن‌ها رجوع می‌شود، ایجاد شده است که مهم‌ترین آن‌ها، جایگزینی app.js با script.js است.
- باقی فایل‌های .js پروژه را هم در فولدر src/js کپی می‌کنیم. همانطور که قبلاً اشاره شد، به جای script.js فایل app.js را جایگزین می‌کنیم. توضیحات مربوط به این فایل در ادامه آمده است:

a) در ابتدا تابع getCreditors را داریم که با صدا زدن تابع سمت سرور add_IOU تمامی Creditorها را باز می‌گرداند.
b) سپس تابع getCreditorsForUser را داریم که در آن ابتدا با استفاده از تابع getCreditors تمامی creditorها را می‌گیریم و سپس برای user مورد نظر به ازای هر creditor تابع سمت سرور lookup را صدا می‌زنیم و مقدار بدهی به آن creditor را محاسبه می‌کنیم.
c) در ادامه تابع getUsers را داریم که در آن لیست تمامی تراکنش‌ها را می‌گیریم و سپس بر اساس طرفین تراکنش‌ها، لیست تمامی افراد حاضر در شبکه را می‌گیریم.
d) در ادامه تابع getTotalOwed را داریم که در آن با استفاده از تابع getUsers لیست تمامی userها را می‌گیریم و به ازای هر فرد در آن، با استفاده از تابع سمت سرور lookup چک می‌کنیم که میزان بدهی user ورودی به آن فرد چقدر است و مجموع آن‌ها را به عنوان خروجی تابع بر می‌گردانیم.
e) در ادامه getLastActive را داریم که با بررسی زمان تمامی تراکنش‌های انجام شده در شبکه، آخرین تراکنش آن فرد را (یعنی آن تراکنشی که تایم آنجام آن از همه دیرتر بود) بر می‌گردانیم.
f) این تابع برای انجام اضافه کردن یک تراکنش بدهی در سمت کاربر تهیه شده که ابتدا در آن BFS را اجرا می‌کنیم، اگر نتیجه این بود که این تراکنش باعث ایجاد دور نمی‌شود که مشکلی نیست اگر نه باید مسیری را که باعث ایجاد دور نشود بیابیم که اینکار به کمک تابع findMinDebt انجام می‌گیرد. سپس تابع add_IOU در سمت سرور را صدا می‌زنیم.
g) در انتها اجرای برنامه در غالب یک شمای کلی نشان داده می‌شود که در آن باید ترتیب اجرای فرایندها را مشخص کنیم (لازم به ذکر است که من شمای کلی آن در اینگونه پروژه ها را از اینترنت جستوجو کردم و به همین علت با اصطلاحات مربوط به آن آشنا نیستم)
h) در این قسمت ابتدا باید یک Instance از web3 بگیریم و همانند آنچه در script.js قرار داده شده بود، پارامترهای مربوط به smart contract خود را ست می‌کنیم و بعد از ست کردن provider آن منتظر رخدادن eventها می‌مانیم و متناظر با هر event پاسخ می‌دهیم و در آخر هم render را صدا می‌زنیم تا صفحه html را بالا بیاورد.

❖ اجرا

- ابتدا لازم است که ganache-cli را بالا بیاوریم؛ اینکار را هم با دستور آورده شده در صورت پروژه و هم با نصب و اجرای نرم‌افزار آن می‌توان انجام داد.
- ابتدا لازم است که فایل .sol را compile کرده و عملیات migration را انجام دهیم. برای این کار از دستور زیر استفاده می نماییم:

truffle migrate –reset

- اگر که عملیات بدون هیچ خطایی انجام شد، حال می‌توانیم interface آن را مشاهده کنیم. برای این کار دستور زیر را وارد می نماییم:

npm run dev

- با این کار، به طور خودکار صفحه html در browser باز می‌شود؛ در ابتدا صفحه metamask بالا می‌آید که باید اول اکانت مورد نظر خود را مشخص کنیم و سپس می‌توانیم از امکانات صفحه استفاده نماییم.

- لازم به ذکر است که با توجه به ویدیوی ضبط شده از اجرای پروژه، از روند اجرا در گزارش تصاویری تهیه نشده است.