Final Year Project - Real time Trading Platform

A platform that allows multiple users to buy and sell resources, in real time. Call it a stock exchange if you will.

Demo Links

- Demo: https://youtu.be/OlzLp7X_UnY
- Deployment: https://youtu.be/IOcSUKTTSqc

Report

A copy of the report is found at the root of the project: final_report.pdf

Technologies

Backend

- Golang
- Docker
- RabbitMQ
- Postgres
- Redis

Frontend

- SolidJS (known for its speed)

Dependencies

There are a few dependencies in the project so, there are some requirements: - Nodejs (Tested on V16 and V18): This is to run the frontend dev server. - Docker (Tested on Arch Linux and Ubuntu): Runs the backend. - This will depend on your machine, but there are plenty of information online on installing docker: https://docs.docker.com/get-docker/ - Docker Compose: Frontend for running docker images - Golang: For developing the backend. (Tested on 1.20, should work on 1.19 or above). - Pnpm: Improved NPM. (Package manager for the frontend, tested with version 8).

Folder structure

- report. Includes all the .tex files for the report, along with screenshots and other materials.
- apps. The applications in my entire system
  - apps/frontend - Frontend folders, make sure you `cd` into this directory when doing any frontend work.

  - apps/backend - Main folder for my backend, here you can find docker images, docker compose files and all 3 of my backend systems.
    * apps/backend/hub - The hub service.
    * apps/backend/brain - The brain service.
    * apps/backend/auth - The auth service.

Running the project

Frontend

To run the frontend you will also need an .env file, which has already been filled out in the `.env.example` file, you can just rename this file to `.env`.

```
cd apps/frontend
pnpm install
pnpm run dev
```

E2E Testing

I have used Cypress for end to end testing, which creates a browser and attempts to use to create trades and use the website in a black box kind of way, to mimic a real user, to use it:

```
cd apps/frontend
pnpx cypress install
pnpx cypress run

(Optimally you can run the following command to access the GUI):
pnpx cypress open
```

NOTE: You need to have the application running for the tests to work, cypress mimics the user as much as possible and won't run the application for us. Look at the sections on how to run the frontend and backend.

Backend

```
cd apps/backend
docker-compose up --build //The flag is optional but ensures the containers are re-built.
```

There are also various useful scripts you might want. All in the apps/backend folder. - CleanDb.bash -> Requires sudo to run and deletes the data of your local databases. - HubAndBrain.bash -> Runs the hub and brain services as simple processes (without docker), useful for development. - NoHub.bash -> Runs the docker compose file but without the hub, which you can run individually. Again, useful for development.

Sometimes you might also want to run the docker images without the hub and brain, for development, which you can do with the following command.

```
docker-compose -f no-brain-hub.yml up --build
//Again, the flag is optional if you want to rebuild the containers.
```

Integration Testing

Integration testing must be done whilst the backend services are running, so you must perform the stage above and only after run this. Will make this process nicer in the future.

You will also need a `.env` file, which you need to create in the integration testing folder. There is an example `.env.example` file, for convinient, you can just rename this file to `.env` and it will work, as it is prefilled with localhost information.

```
cd apps/integration
npm install
npm run prisma:gen // Generates the required type files for the Prisma ORM.
npm run test // Can also do npm run test:watch
```

Note: The integration tests completely wipes your database to get accurate testing results, so you might need to run `CleanDb.bash`, to get the test users back.

Extra information

There is a README.md in each backend service (Auth, Hub and Brain), where you can find out more about each service.