

# Dossier

---

- Student: Dogukan Uyanik
- Studentennummer: 202295314
- E-mailadres: <mailto:dogukan.uyanik@student.hogent.be>
- Demo: <https://hogent.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=fb3cec75-6550-4bd4-b0b3-b24c00201ec0>
- GitHub-repository: <https://github.com/HOGENT-frontendweb/frontendweb-2425-DogukanUyanik04>
- Front-end Web Development
  - Online versie: <https://frontendweb-2425-dogukanuyanik04.onrender.com>
- Web Services:
  - Online versie: <https://stocks-webserv.onrender.com>

## Logingegevens

### Lokaal

- Gebruikersnaam/e-mailadres: john@example.com
- Wachtwoord: 12345678

### admin

- Gebruikersnaam/e-mailadres: jane@example.com
- Wachtwoord: 12345678

### Online

- Gebruikersnaam/e-mailadres: public@gmail.com
- Wachtwoord: Public12345678

### admin

- Gebruikersnaam/e-mailadres: jane@example.com
- Wachtwoord: 12345678

## Projectbeschrijving

De applicatie geeft de gebruiker de mogelijkheid om in een nepomgeving in aandelen te handelen. Door dit te doen raakt de gebruiker vertrouwd met de aandelenmarkt. Het is een eerste stap om een echte belegger te worden.

[Gebruiker] \*id naam email balans password\_hash roles

[Transactie] \*id +gebruikerId +AandeelId hoeveelheid prijstransactie soorttransactie datum

[Aandeel] \*id afkorting naam huidigeprijs +marktId

[Markt] \*id naam valuta

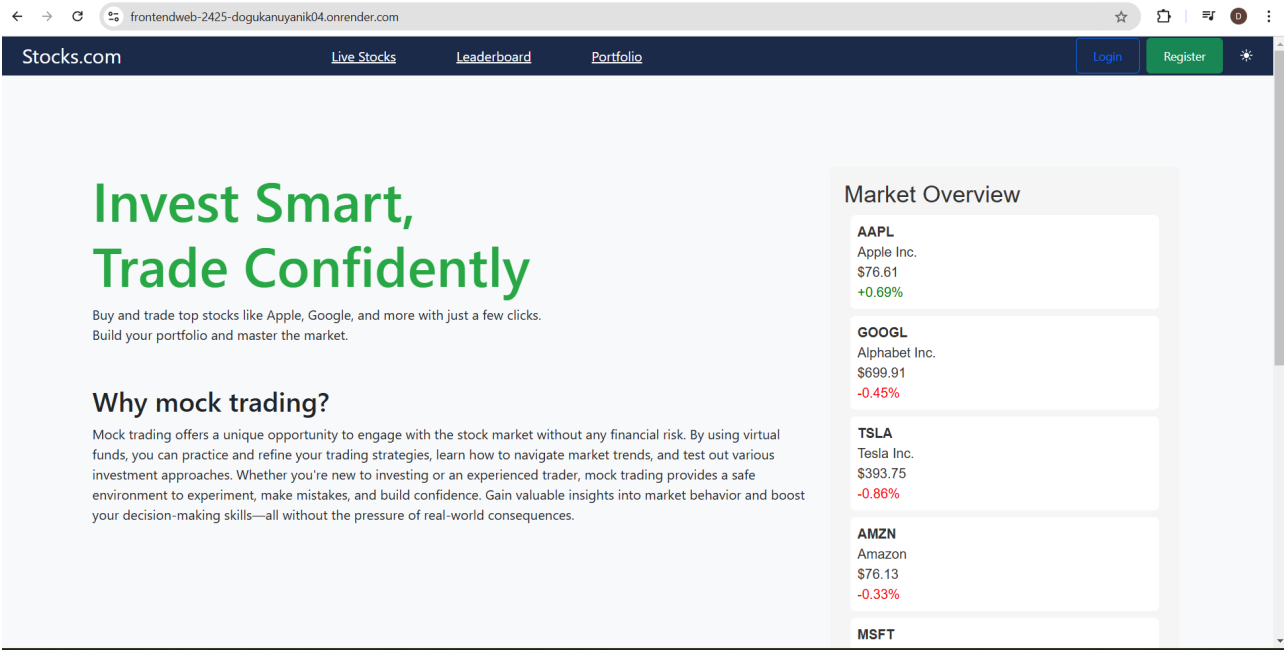
Gebruiker 1--\* Transactie

Aandeel 1--\* Transactie

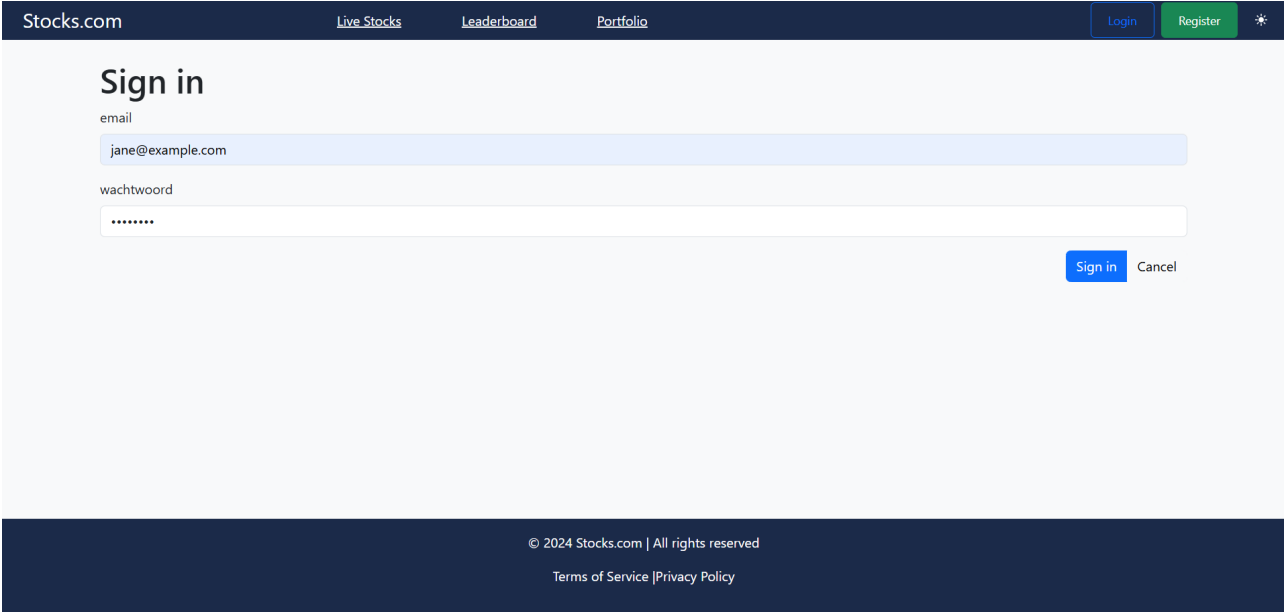
Markt 1--\* Aandeel

Screenshots

- Homescreen



- LoginsScreen



- Livestocks

Stocks.com

[Live Stocks](#)

[Leaderboard](#)

[Portfolio](#)

Logout


\*

Browse all live stocks

Search for your favourite stock

Search by stock name


Search

 **Apple** AAPL

150 \$

Buy


Sell

 **Tesla** TSLA

521 \$

Buy


Sell

 **Google** GOOG

1889 \$

Buy


Sell

 **Amazon** AMZN

3110 \$

Buy

Sell

 **Microsoft** MSFT

160 \$

Buy

Sell

© 2024 Stocks.com | All rights reserved

- Buyform

Stocks.com

[Live Stocks](#)

[Leaderboard](#)

[Portfolio](#)

Logout

\*

Buy Google

Your Balance: 49380 \$

Price (per share):

1905 \$

Quantity:

1

Total Price:

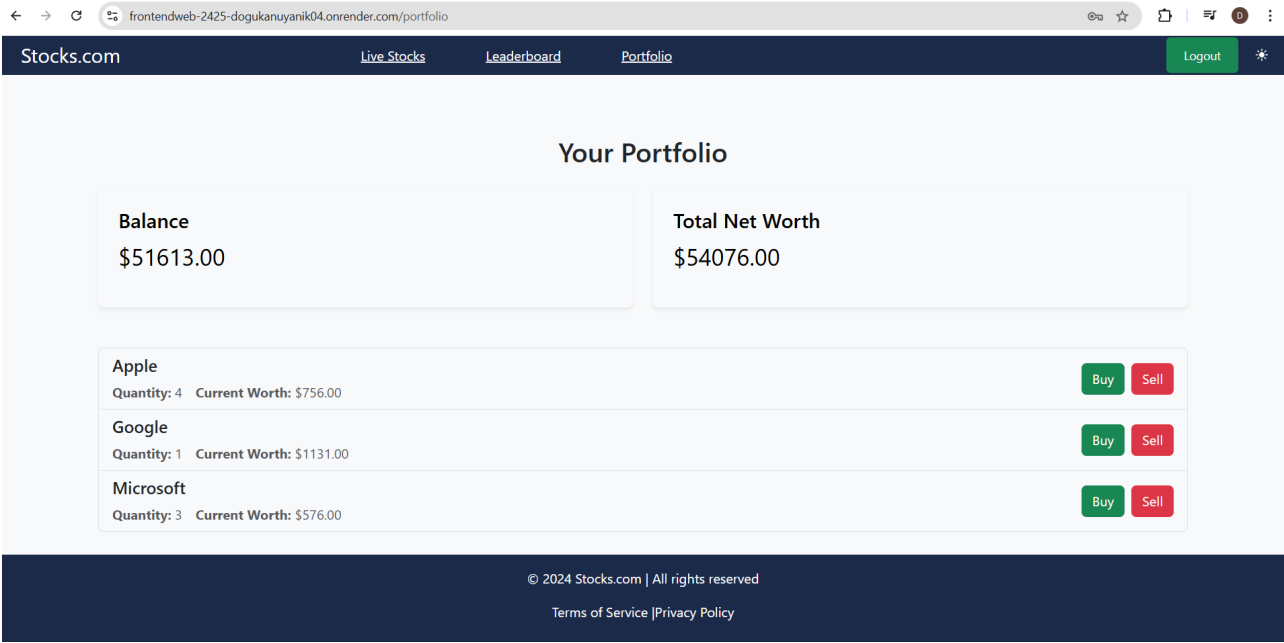
1905.00 \$

Buy

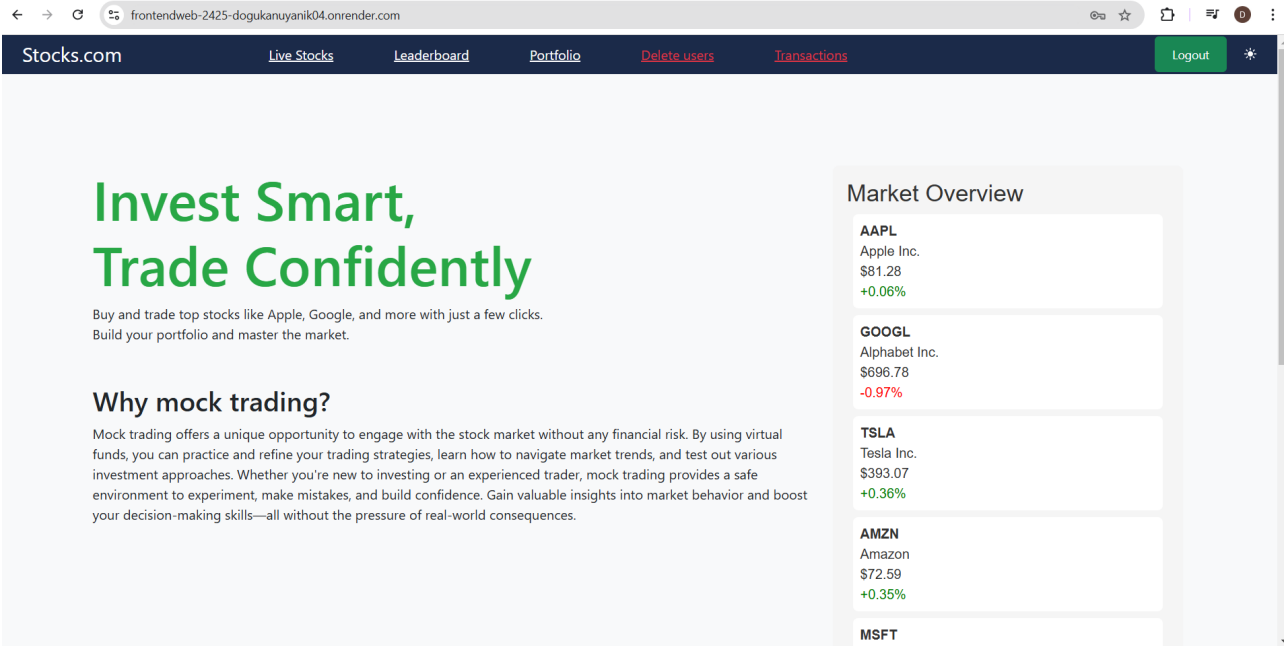
© 2024 Stocks.com | All rights reserved

[Terms of Service](#) | [Privacy Policy](#)

Portfolio



HomescreenAdmin



- DeleteUsersAdminOnly

Stocks.com

[Live Stocks](#)[Leaderboard](#)[Portfolio](#)[Delete users](#)[Transactions](#)

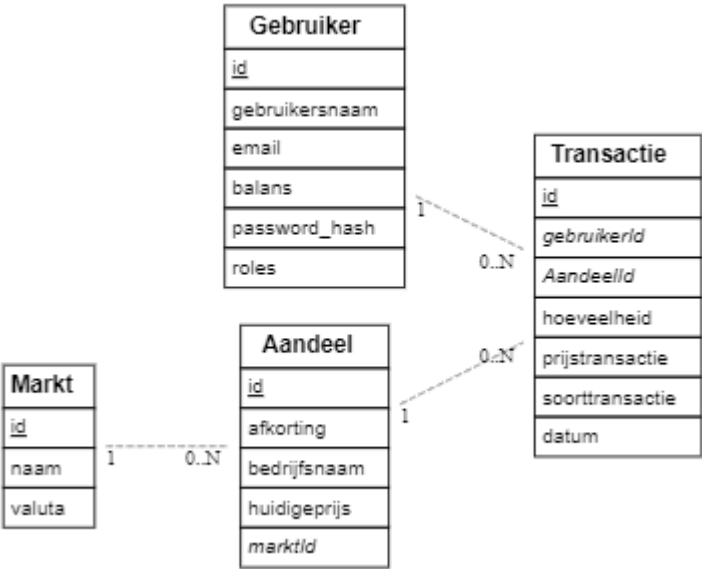
Logout

### Delete Users

|                               |        |
|-------------------------------|--------|
| (jane@example.com)            | Delete |
| (john@example.com)            | Delete |
| (marie@example.com)           | Delete |
| (dogukanuyanik9140@gmail.com) | Delete |
| (gizem.uyanik.98@gmail.com)   | Delete |
| (public@gmail.com)            | Delete |
| (Yourstruly@gmail.com)        | Delete |
| (Eneskulac52@gmail.com)       | Delete |
| (kerim09bugday@gmail.com)     | Delete |

© 2024 Stocks.com | All rights reserved

[Terms of Service](#) | [Privacy Policy](#)



- EERD

## API calls

### Aandelen

- GET /api/aandelen: alle aandelen opvragen
- GET /api/aandelen/:id: aandeel met een bepaalde id opvragen
- GET /api/aandelen/name/:name: aandeel met een bepaalde naam opvragen

### Gebruikers

- GET /api/gebruikers: alle gebruikers ophalen
- GET /api/gebruikers/:id: gebruiker met een bepaald id ophalen
- GET /api/gebruikers/:id/balance: balans van een gebruiker met een bepaalde id ophalen
- GET /api/gebruikers/:id/transacties: alle transacties van een gebruiker met een bepaald id ophalen
- GET /api/gebruikers/:id/portfolio: portfolio van een gebruiker ophalen
- GET /api/gebruikers/user/stocks: alle aandelen dat een gebruiker bezit ophalen

- `GET /api/gebruikers/leaderboardpagina`: alle gebruikers ophalen voor de leaderboard component
- `POST /api/gebruikers`: nieuwe gebruiker aanmaken/registreren
- `PUT /api/gebruikers/:id`: gebruiker met een bepaald id updaten
- `DELETE /api/gebruikers/:id`: gebruiker met een bepaald id verwijderen

## Health

- `GET /api/health/ping`: de server pingen
- `GET /api/health/version`: versie info van server ophalen

## Sessions

- `POST /api/sessions`: endpoint om een nieuwe login session aan te maken.

## Transacties

- `GET /api/transcaties`: alle transacties ophalen
- `GET /api/transcaties/:id`: transacties met een bepaalde id ophalen
- `POST /api/transcaties/buy`: nieuwe koop transactie creëren
- `POST /api/transcaties/sell`: nieuwe verkoop transactie creëren

## Swagger

<https://stocks-webserv.onrender.com/swagger>

# Behaalde minimumvereisten

## Front-end Web Development

### Componenten

- ☒ heeft meerdere componenten - dom & slim (naast login/register)
- ☒ applicatie is voldoende complex
- ☒ definieert constanten (variabelen, functies en componenten) buiten de component
- ☒ minstens één form met meerdere velden met validatie (naast login/register)
- ☒ login systeem

### Routing

- ☒ heeft minstens 2 pagina's (naast login/register)
- ☒ routes worden afgeschermd met authenticatie en autorisatie

### State management

- ☒ meerdere API calls (naast login/register)
- ☒ degelijke foutmeldingen indien API-call faalt
- ☒ gebruikt useState enkel voor lokale state
- ☒ gebruikt gepast state management voor globale state - indien van toepassing

## Hooks

- ☒ gebruikt de hooks op de juiste manier

## Algemeen

- ☒ een aantal niet-triviale én werkende e2e testen
- ☒ minstens één extra technologie
- ☒ node\_modules, .env, productiecredentials... werden niet gepushed op GitHub
- ☒ maakt gebruik van de laatste ES-features (async/await, object destructuring, spread operator...)
- ☒ de applicatie start zonder problemen op gebruikmakend van de instructies in de README
- ☒ de applicatie draait online
- ☒ duidelijke en volledige README.md
- ☒ er werden voldoende (kleine) commits gemaakt
- ☒ volledig en tijdig ingediend dossier

## Web Services

### Datalaag

- ☒ voldoende complex en correct (meer dan één tabel (naast de user tabel), tabellen bevatten meerdere kolommen, 2 een-op-veel of veel-op-veel relaties)
- ☒ één module beheert de connectie + connectie wordt gesloten bij sluiten server
- ☒ heeft migraties - indien van toepassing
- ☒ heeft seeds

### Repositorylaag

- ☐ definieert één repository per entiteit - indien van toepassing
- ☐ mapt OO-rijke data naar relationele tabellen en vice versa - indien van toepassing
- ☐ er worden kindrelaties opgevraagd (m.b.v. JOINS) - indien van toepassing

### Service laag met een zekere complexiteit

- ☒ bevat alle domeinlogica
- ☒ er wordt gerelateerde data uit meerdere tabellen opgevraagd
- ☒ bevat geen services voor entiteiten die geen zin hebben zonder hun ouder (bv. tussentabellen)
- ☒ bevat geen SQL-queries of databank-gerelateerde code

### REST-laag

- ☒ meerdere routes met invoervalidatie
- ☒ meerdere entiteiten met alle CRUD-operaties
- ☒ degelijke foutboodschappen
- ☒ volgt de conventies van een RESTful API
- ☒ bevat geen domeinlogica
- ☒ geen API calls voor entiteiten die geen zin hebben zonder hun ouder (bv. tussentabellen)
- ☒ degelijke autorisatie/authenticatie op alle routes

## Algemeen

- ☒ er is een minimum aan logging en configuratie voorzien
- ☒ een aantal niet-triviale én werkende integratietesten (min. 1 entiteit in REST-laag >= 90% coverage, naast de user testen)
- ☒ node\_modules, .env, productiecredentials... werden niet gepushed op GitHub
- ☒ minstens één extra technologie die we niet gezien hebben in de les
- ☒ maakt gebruik van de laatste ES-features (async/await, object destructuring, spread operator...)
- ☒ de applicatie start zonder problemen op gebruikmakend van de instructies in de README
- ☒ de API draait online
- ☒ duidelijke en volledige README.md
- ☒ er werden voldoende (kleine) commits gemaakt
- ☒ volledig en tijdig ingediend dossier

## Projectstructuur

### Front-end Web Development

Het mapje `stocks_frontend` bevat een cypress mapje met daarin alle e2 testen, fixtures en enkele commands die helpen met het inloggen bij de testen. Het mapje `node_modules` bevat alle geïnstalleerde dependencies nodig voor dit project. Het mapje `public` bevat een foto dat dient als favicon voor het website. Ten slotte het mapje `src`, hier zit al het logica van de frontend. Als eerst het mapje `api`, hierin is er een aparte file voor elk entiteit waarin de api calls naar de backend zich bevinden, deze api calls hebben we nodig in onze componenten. Vervolgens het mapje `components`, hierin zit voor elk component van ons website de code, bijvoorbeeld een component aandeellijst waar elk aandeel mooi getoond word op een lijstje. Hierna hebben we het mapje `contexts`, hierin kunnen we globale instellingen toepassingen op ons website, bijvoorbeeld om het website een donkere thema te geven, of om te controleren of de gebruiker is ingelogd en zo navigeren doorheen de website. Ten slotte hebben we het mapje `pages`, hierin worden componenten opgeroepen om ze te tonen om de pagina. Een van de belangrijkste files in de `pages` folder is de `main.jsx`, hier gebeurt alle routing van de pagina's.

### Web Services

Het mapje `stocks_webserv` bevat een tests mapje waarin alle testen plaatsvinden voor de backend. In de folder `tests` hebben we een folder `helpers`, dit zijn files dat toepasselijk is op alle testen zoals ingelogd zijn, ingelogd zijn als een admin, ... Dit bespaart veel typwerk. Een ander mapje `rest` met daarin een testfile voor elk entiteit. Het mapje `config` bevat alle configuratie instellingen, dit kan verschillen als we in development, productie of in testing zitten. Hierna hebben we waarschijnlijk het belangrijkste mapje van onze backend, de `src` folder. Hierin hebben we allereerst het `core` mapje, hier gebeuren zaken zoals authenticatie, jwt tokens genereren, rollen definiëren, validatie en errors en een van de belangrijkeren, het installeren van de middleware. Vervolgens hebben we het mapje `data`, hierin praten we voornamelijk met de databank, we definiëren een schema prisma, wat ons eerd voorstelt, en kunnen daarmee ons databank opstellen, het seed bestandje zorgt ervoor dat de databank al een beetje word opgevuld met data. Vervolgens hebben we de `rest` folder, hierin hebben we een bestand voor elk entiteit waarin hun endpoints gedefinieerd worden, ze worden ook gevalideerd met hulp van Joi en er is documentatie voorzien. Het volgende mapje is `scripts`, dit bevat 2 files die uitgevoerd moeten worden bij het opstarten van de backendserver. Het volgende mapje is de `service` folder waarin alle business logica zit van de entiteiten. Het volgende mapje is de `types` folder, hierin worden alle types van elk entiteit gedefinieerd, dit kan bijvoorbeeld helpen dat elk function of endpoint waar je je type gebruikt, een mooi



structuur blijft houden. Als laatst hebben we het mapje utils met daarin een bestand dat zorgt dat er een mail word verstuurt naar elk gebruiker dat registreert (extra). Het bestandje .env is waar de backend zijn geheime environment variabelen zoals databank url, ... .

## Extra technologie

### Front-end Web Development

Voor Frontend heb ik 2 extra technologieen geïmplementeerd doordat ze vrij klein zijn, enerzijds heb ik fakerjs geïmplementeerd wat ik gebruikte tijdens development van mijn frontend in vroege fase, dit genereerde erg snel mockdata voor me, ik heb het nog gelaten in de component stockdatawindow als bewijs dat het is gebruikt <https://www.npmjs.com/package/@faker-js/faker>.

Anderzijds heb ik een wachtwoordsterke indicator gebruikt in mijn registerform component, het module dat ik hiervoor heb gebruikt noemt zxcvbn <https://www.npmjs.com/package/zxcvbn>.

### Web Services

Als extra voor mijn backend heb ik nodemailer geïmplementeerd, dit zorgt ervoor dat je door gebruik van je email account, emails automatisch kan versturen bij specifieke handelingen in de backend. In mijn geval stuur ik een email naar het mailadres van de geregistreerde gebruiker om hun te verwelkomen op de website met wat achtergrondinfo. <https://www.npmjs.com/package/nodemailer>

## Gekende bugs

### Front-end Web Development

De jwt token kan soms plots vervallen, maar weet niet meestal wanneer dit voorkomt, opnieuw inloggen is een fix maar wil dit in de toekomst nog verbeteren.

### Web Services

Het url door render gegenereerd voor mijn backend is <https://stocks-webserv.onrender.com> Nu als ik rechtstreeks naar deze url request maak, failen mijn requests uiteraard doordat er geen /api prefix is in de url. Mijn requests van mijn frontend zijn wel allemaal succesvol en ze worden ook gelogged in de console in render. Het zou dus handiger geweest zijn moest de url de prefix al bevatten, of is dit bewust gedaan en is het volkomen normaal?

## Reflectie

Ik heb enorm veel bijgeleerd, het was super tof dat we zelf ons onderwerp mochten kiezen en dat uitwerken aan de hand van de cursus. Eerder had ik nog nooit gewerkt met typescript en niet zo erg veel met javascript, dit project heeft dus mijn kennis in beide talen enorm veel verbeterd. Ik zou later graag als fullstack developer willen werken en dus was dit project een ideaal eerste stap. Wat ik anders gedaan zou hebben en waar ik me ook aan erger is mijn consistentie in bijvoorbeeld naamgeving, ik begon aan mijn project in het nederlands, maar halverwege dacht ik dat ik het na de deadline wat uitgebreider wil maken en dus publiceren op github voor iedereen of op mijn portfolio zetten, het zou dan slimmer geweest zijn moest mijn code in het engels staan zodat het begrijpelijker zou zijn voor iedereen. Ik wou dit nog fixen, maar het zou een hele wierwar worden.

De cursus is erg goed in elkaar gestoken, het was soms eens opzoekwerk doordat mijn project natuurlijk verschilt van het voorbeeldprojectje, maar dat is absoluut geen probleem doordat ik zo veel heb bijgeleerd. Misschien een klein aanpassing kan zijn dat er in hoofdstuk 7 Testing van frontend een klein uitleg kan zijn dat je ingelogd moet zijn voor de testen, dus eerst de cypress commands maken, en dan pas kan je je routes testen die afgebakend zijn voor authenticatie.