

Portfolio 2 - Showtime

DATA2410, Spring 2021

Start the program	1
How to allow https connection with self-signed certificate	2
Download and validate the certificate	2
Mac with Safari:	2
Upload the certificate to keychain on mac	3
In browser with Chrome	4
Add self-signed certificate to Windows	5
If you don't want to run with an unauthorized certificate	6
Program flow	7
Product page	7
More info	8
Create order and add to cart	8
Continue unfinished order	8
Shopping cart page	9
Checkout and receipt	9
Admin page	10
Upload product	11
Technical info	11
General info	11
Technical requirements	12
OAuth2	12
Google setup	12
Server setup	12
SSL/TLS	13
Prometheus/Grafana	14

Start the program

1. Start Docker on your computer
2. Open the terminal in the "api" folder inside the project
3. Type in command "docker compose build"
4. Type in command "docker compose up"
5. After a while the program will run on port 5000
6. Enter the URL "<https://localhost:5000>" in your web browser

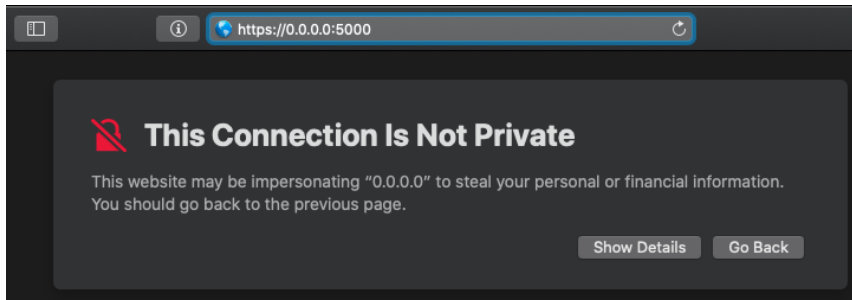
s341856, s341827, s341888

How to allow https connection with self-signed certificate

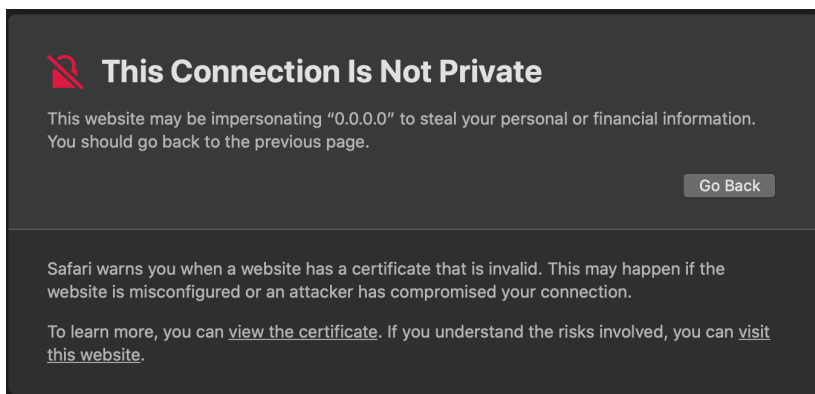
Download and validate the certificate

Mac with Safari:

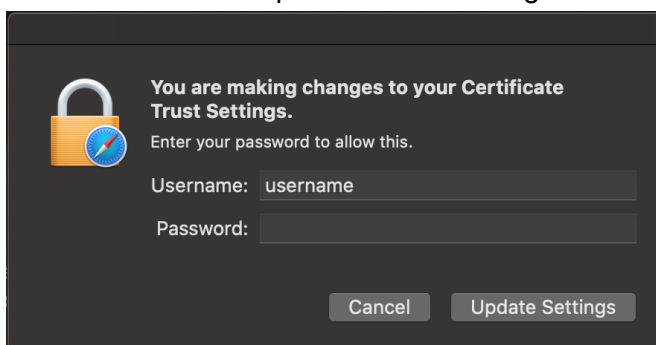
Type in the URL for the page with https://



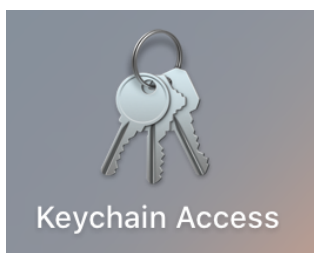
Click on the “Show Details” button and then “visit this website”



Enter username and password for adding the Certificate to the Chain of Trust

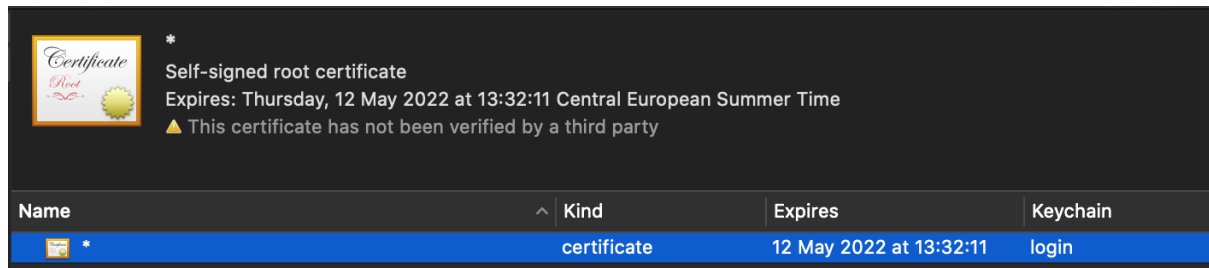


Now it's added to the keychain. Open the keychain Access app

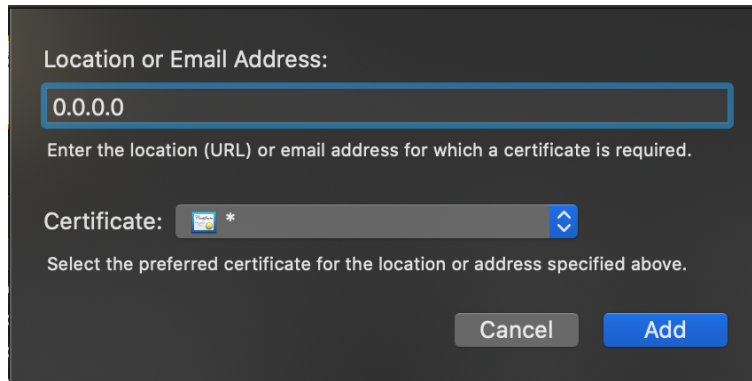


Find the self-signed certificate that you just added, right click on it and choose “New Certificate Preferences”

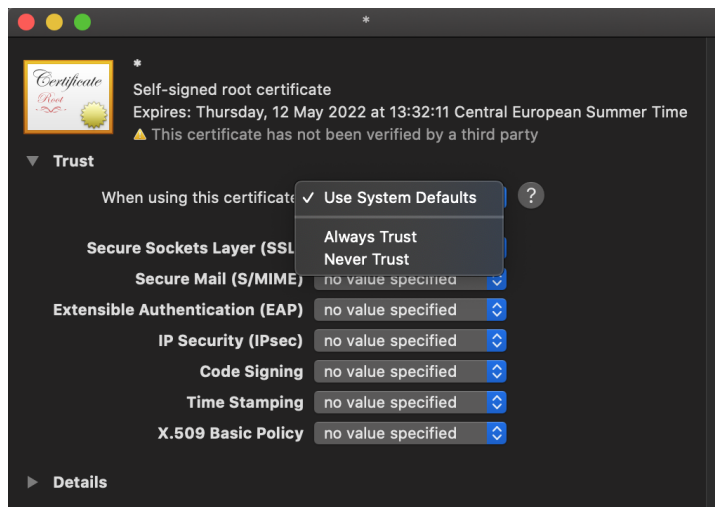
s341856, s341827, s341888



Enter the localhost address and click add

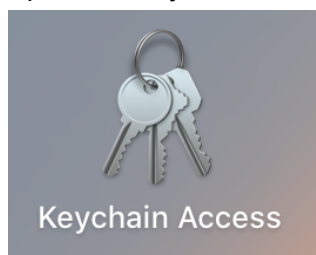


Right click on the certificate again and choose "show info", under "Trust" -> "When using this certificate", choose "Always Trust". Confirm with password to make the changes. You can now access the page without security-restrictions.



Upload the certificate to keychain on mac

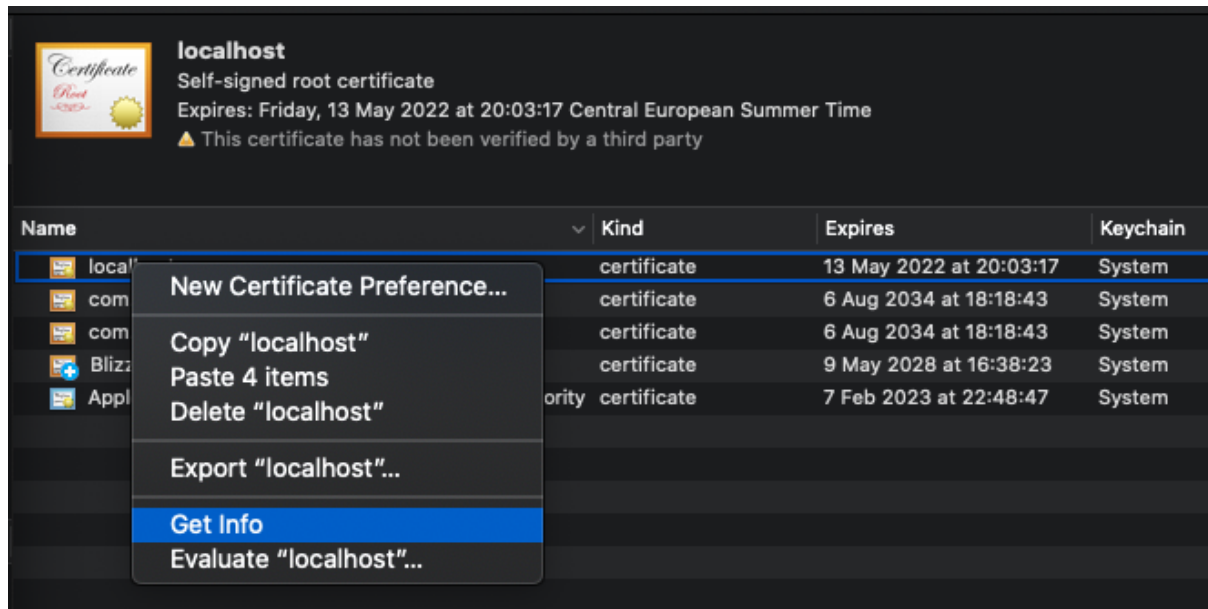
Open the Keychain Access app



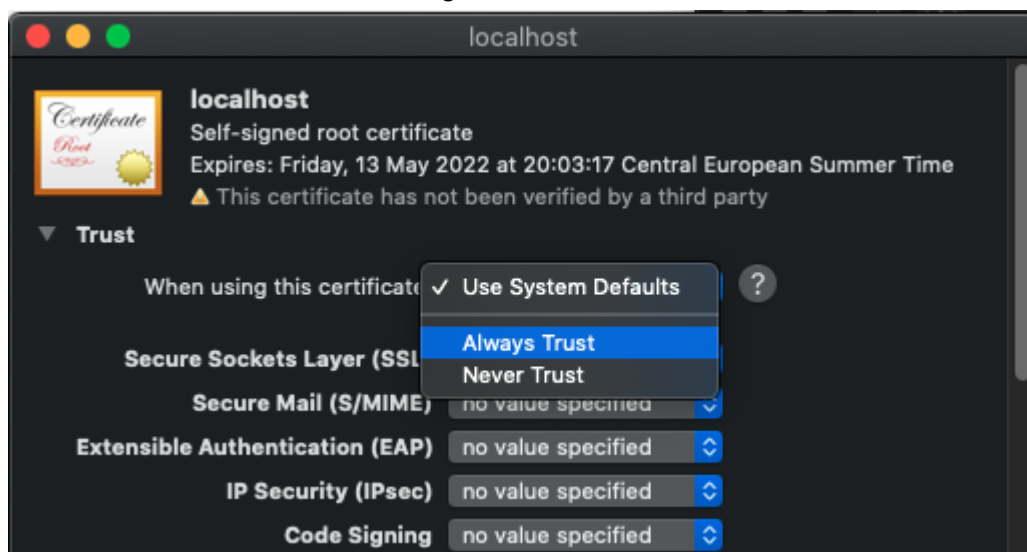
s341856, s341827, s341888

Choose System on the right side and either drag the certificate.cert from the api folder in the project or press + on the top and find the file path to the certificate.cert file. Confirm with password.

Now, right click on the certificate that was added to the folder and click "Get Info"



Under Trust, click "Always Trust", exit the window and confirm with password. You can now enter the site with a trusted self-signed certificate.



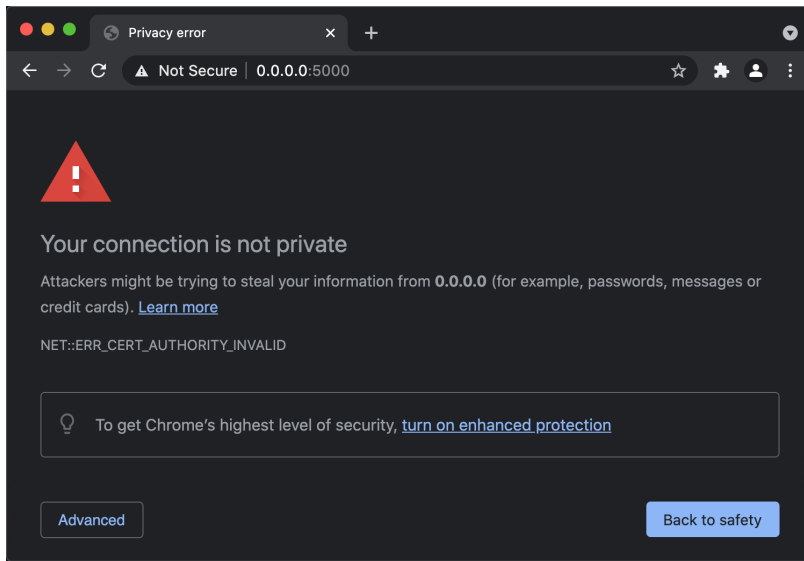
In browser with Chrome

Enter the page in Chrome

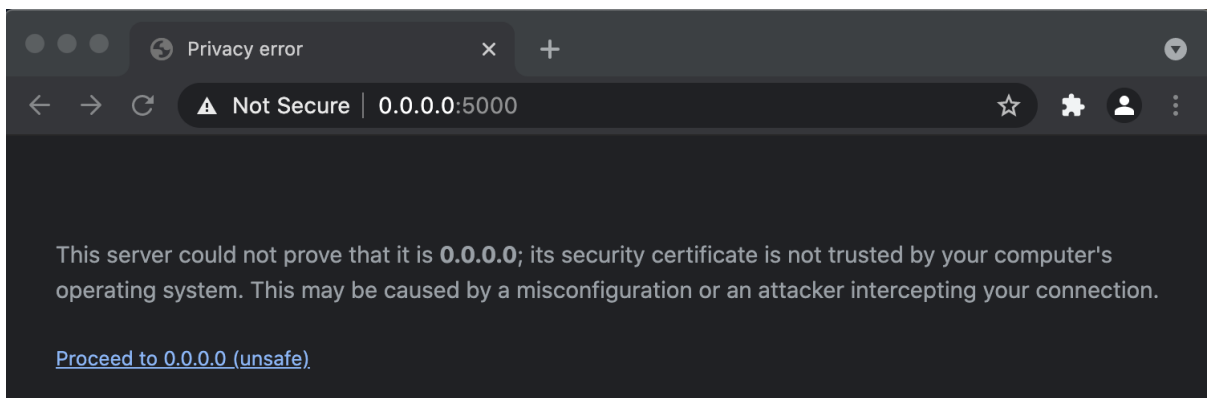
OBS! Remember here to allow insecure content under Settings -> Privacy and security -> Site Settings -> Insecure content and Allow Insecure content for the page



s341856, s341827, s341888



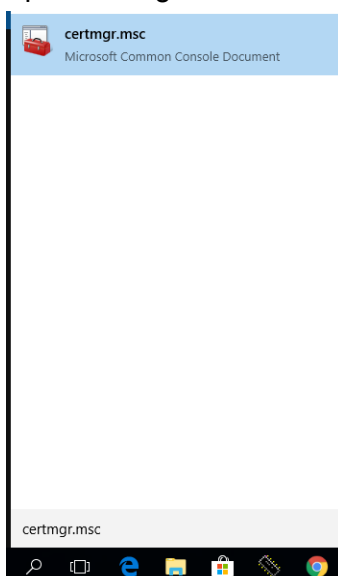
Click "Advanced"



Then click "Proceed to ... (unsafe)"

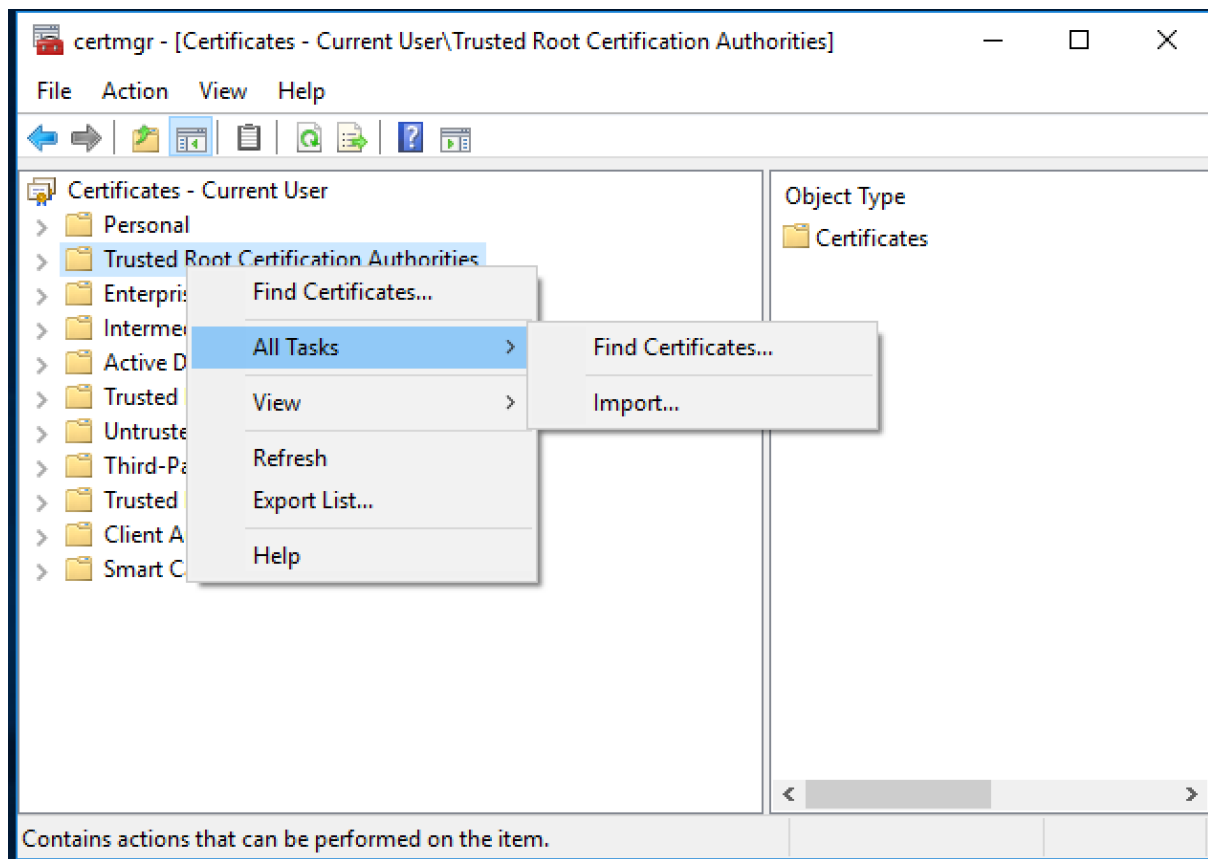
Add self-signed certificate to Windows

Open certmgr.msc



s341856, s341827, s341888

Right click the “Trusted Root Certification Authorities”
Choose All Tasks -> Import...



Click next in the new window, and find the file path to the certificate.crt certificate inside the api folder in the project. Click next and finish in the next windows, a warning will occur at the end. This is due to the certificate not being authorised by a third party. But it will only run on localhost, so it is safe to accept it. Now the certificate is added to the trusted certificates on your computer, you can now enter the website problem free.

If you don't want to run with an unauthorized certificate

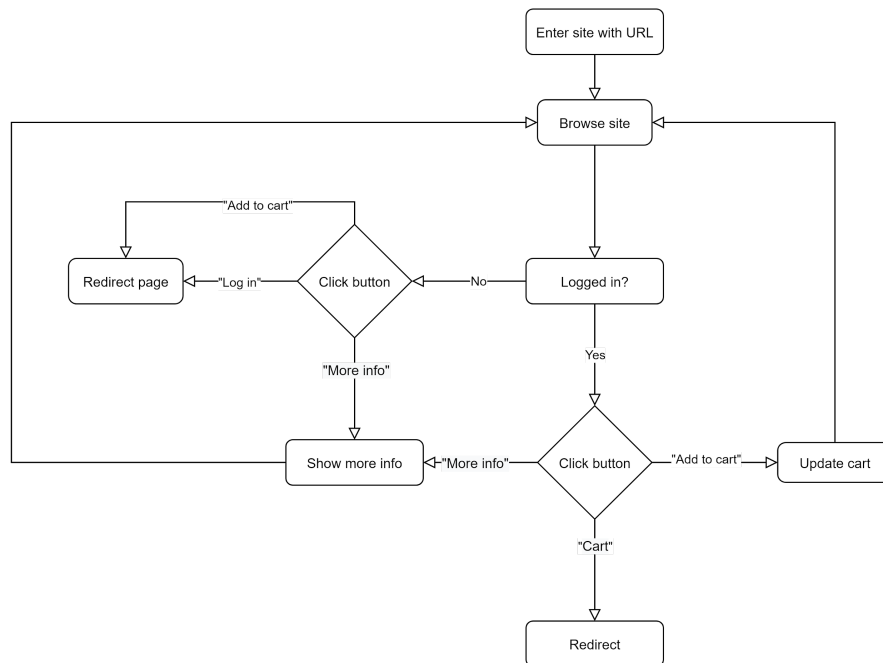
Simply remove the
“, ssl_context=('cert.pem', 'key.pem')”
from the bottom line in the file api/app.py

The bottom line should now say
app.run(host='0.0.0.0', debug=False)

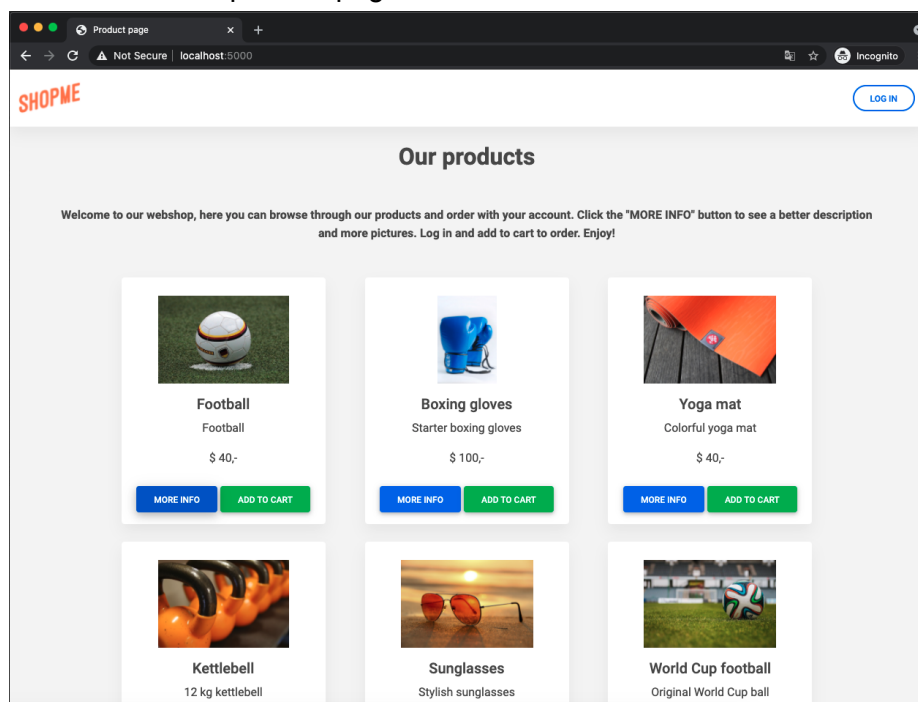
s341856, s341827, s341888

Program flow

Product page



Flow chart of the product page



How the product page looks

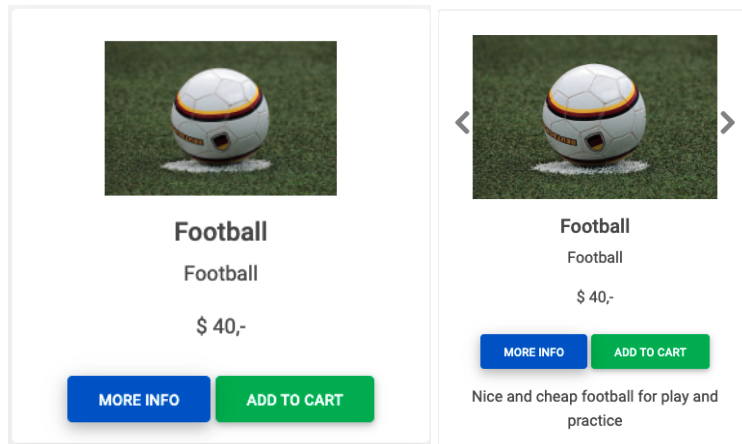
Product pictures are downloaded from pixabay.com (free for use).

The products are collected directly from the database. If a picture can't load or the product doesn't contain a picture, a placeholder will appear instead. When new products are uploaded the database will be updated and the products appear in the product page.

s341856, s341827, s341888

More info

When “MORE INFO” is clicked, arrows for more pictures appear if there are more than one picture. If there is no picture or it fails to upload one, a placeholder will be displayed instead. There will also be shown a longer description of the product.



Create order and add to cart

In order to create an order and add products to this order a user has to be logged in. After a user has been logged in and pushes the “ADD TO CART” button, it will automatically create an order, add the products, and a “success” alert will be shown for 8 seconds.

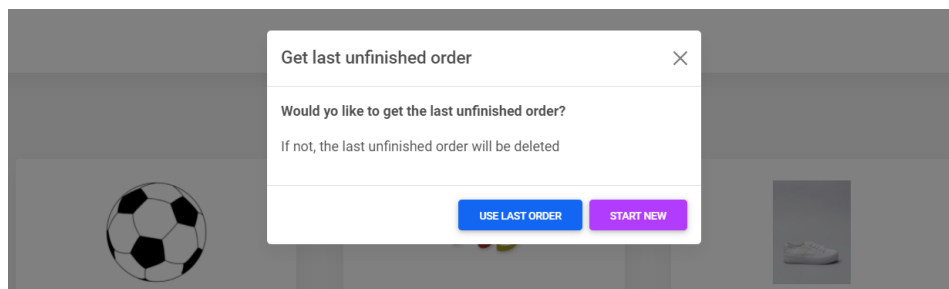
The number of products in the cart will be displayed in the cart icon in the navbar. The cart icon will show how many different products are in the order. It will not count for multiple of the same product.



To check out press the cart displayed in the top right when logged in.

Continue unfinished order

Unfinished orders will be able to open and to be continued if the user is logged out when having an unfinished order in the shopping cart. When logging in again the user can choose to open the unfinished order. If the user doesn't want to finish the order, the old order will be deleted.






s341856, s341827, s341888

Shopping cart page

When the shopping cart symbol is pushed, the page will be redirected to the shopping cart. In the shopping cart the user will see the current order, be able to change the amount of a product, and also delete a product from the order. The total price will automatically change, and it is possible to checkout if the order is correct.

SHOPMEOla

Welcome to your shopping cart

#	Picture	Item	Color	Price	Amount
1		Ball	Black/white	\$100	<input type="text" value="1"/>
2		Strikk	null	\$400	<input type="text" value="1"/>
3		Football	null	\$100	<input type="text" value="1"/>

Ola, view your order




Cart amount:	\$1800
Shipping cost:	Free
Discount:	100%
Total amount	\$0

CHECKOUT

Checkout and receipt

When pressing the checkout button the order will be finished. A receipt will be shown with an auto generated order-id and info about the order. It is possible to print the order as a pdf with the print receipt button.

Your receipt
Order id: 84 **PRINT RECEIPT**

#	Picture	Item	Color	Price	Amount
1		Sko	null	\$700	1
2		Strikk	null	\$400	1
3		Ball	Black/white	\$100	1

Adrian, view your order

Cart amount:	\$1200
Shipping cost:	Free
Discount:	100%
Total amount	\$0

s341856, s341827, s341888

Admin page

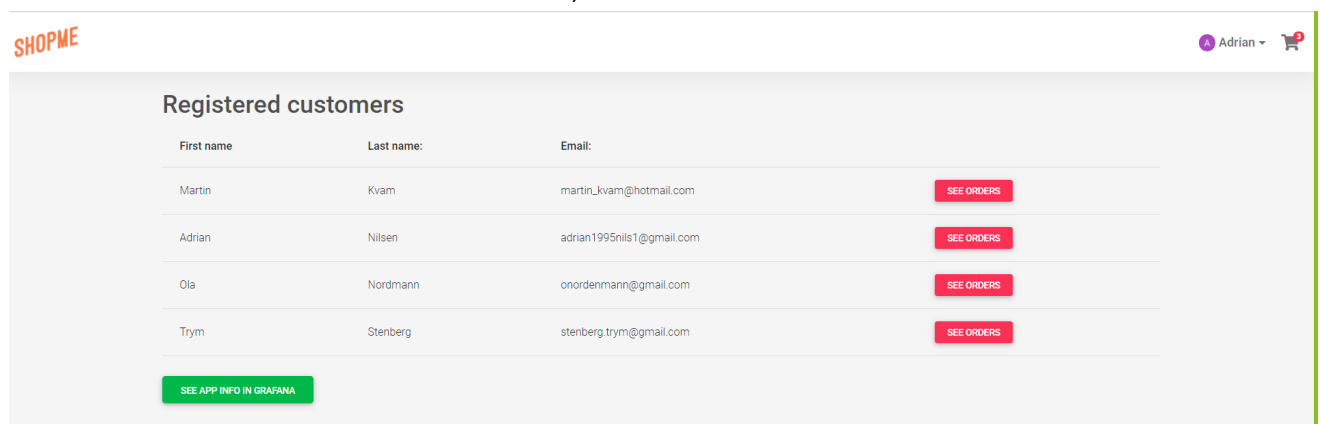
Admin user

Username: onordenmann@gmail.com

Password: admin#123

NB! In order to login to a new google user the browser has to be logged out of the last logged in google user.

If you are logged in as admin, you can view registered customers on the admin page. On the admin page you also find the link to our Grafana dashboard (you can find info on how to view Grafana further down in this document).



SHOPME

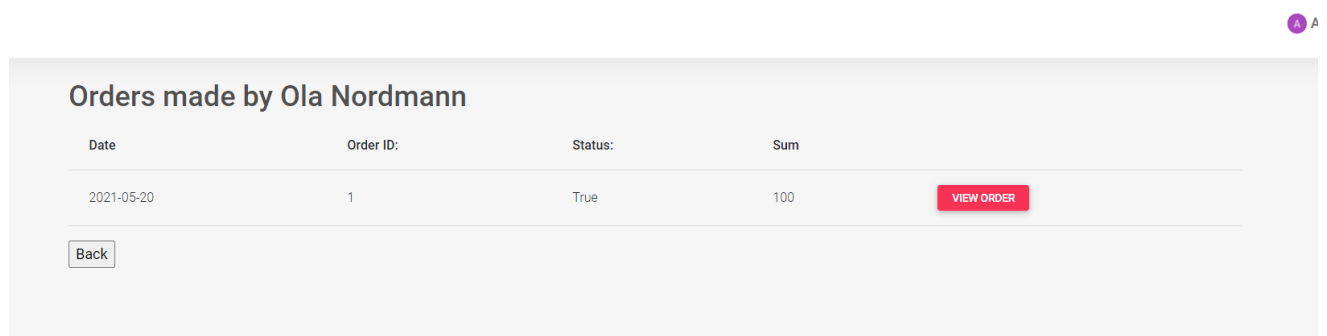
Adrian

Registered customers

First name	Last name:	Email:	
Martin	Kvam	martin_kvam@hotmail.com	SEE ORDERS
Adrian	Nilsen	adrian1995nils1@gmail.com	SEE ORDERS
Ola	Nordmann	onordenmann@gmail.com	SEE ORDERS
Trym	Stenberg	stenberg.trym@gmail.com	SEE ORDERS

SEE APP INFO IN GRAFANA

Next to every customer is a button that takes you to the respective customers previous orders:



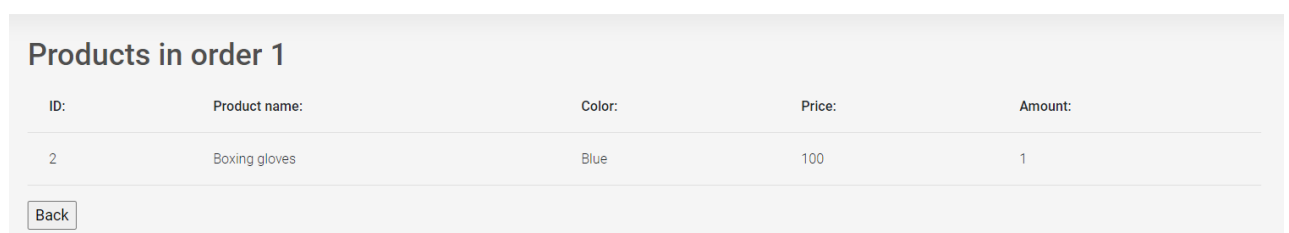
A

Orders made by Ola Nordmann

Date	Order ID:	Status:	Sum	
2021-05-20	1	True	100	VIEW ORDER

Back

Next to every order there is a button that shows all products in that order.



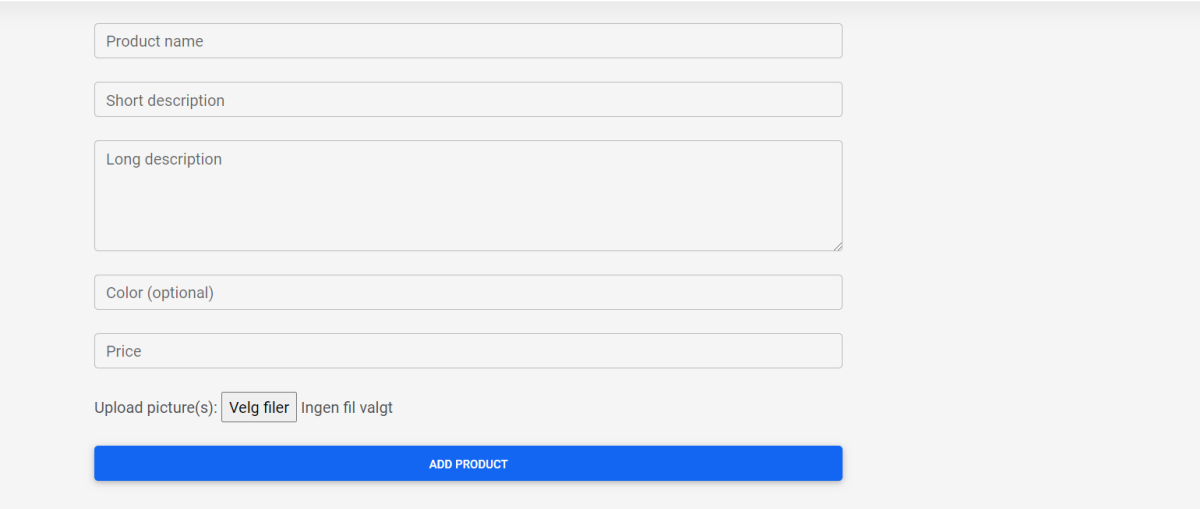
Products in order 1

ID:	Product name:	Color:	Price:	Amount:
2	Boxing gloves	Blue	100	1

Back

Upload product

As an admin, you find the 'upload product' page in the drop down menu:



All the text fields except for color are required. By clicking 'velg filer' / 'choose files' you can upload pictures from your computer. Accepted file types are jpg, png and jpeg. Pictures are not required. If you don't add a picture, or you add a picture of the wrong file type, a placeholder photo is used instead.

When clicking add product, an alert is shown in the top left corner that tells you whether or not the upload was successful. If the upload was successful, you can find the new product on the products page (by clicking the logo).

Technical info

General info

Frontend:

- Custom HTML, CSS and Js
- MDBootstrap and plain Js

Backend: Python3 with flask

Database: SQLAlchemy

SQLAlchemy is used with the flask project to store important data.

s341856, s341827, s341888

Technical requirements

OAuth2

We have used a configuration with “[Authlib](#)” to configure a flask backend google user authentication with OAuth2 as login. This means that every user has to login with Google.

Google setup

The google authentication backend api is set up through “Google cloud platform”. We used an OAuth2 backend platform and started a new project. We created credentials with a main uri and redirect uri's as shown under.

Authorized JavaScript origins

For use with requests from a browser

URIs *

http://localhost:5000

[+ ADD URI](#)

Authorized redirect URIs

For use with requests from a web server

URIs *

http://localhost:5000/login

http://localhost:5000/authorize




http://127.0.0.1:5000/authorize

http://localhost:5000

https://localhost:5000/authorize

https://127.0.0.1:5000/authorize

OAuth 2.0 Client IDs

<input type="checkbox"/>	Name	Creation date ↓	Type	Client ID			
<input type="checkbox"/>	Web client 2	May 11, 2021	Web application	352148568912-qv7v...			

The client id and secret is used for access to the api.

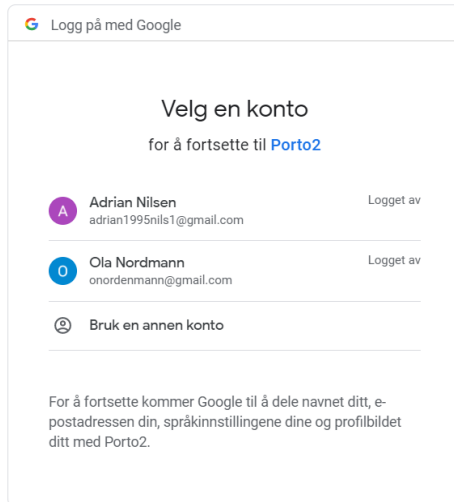
Client ID	352148568912-aqc8n7jb36af5ca1m0pi7om
Client secret	3yIG_r5BdCaiZHmBiuz0w
Creation date	April 26, 2021 at 4:03:41 F

Server setup

The user login with OAuth2 uses the google Authlib configuration. Every time a user is logged in it verifies the user through an access token and the google API. The API gives

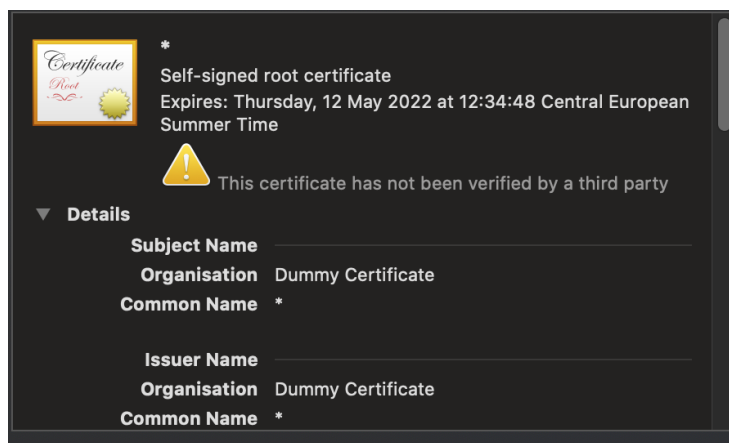
s341856, s341827, s341888

access to the user data. This means that every time a new user is logged in it will automatically be registered as a user in the database with the data that the database needs. Already registered users will access their account when logging in again.



SSL/TLS

Since the page is only local the SSL-connection is not optimal. We have chosen to use a self generated certificate and key for https connection on our page using OpenSSL. The certificate can be added to trusted certificates locally on the computer. Since this certificate is self-signed and not verified by a third party, the web-browser will not trust the connection. It's still possible to visit the secure website by manually "trust it". This can be done directly in some web browsers, or the certificate must be downloaded and added to the chain of trusted certificates. Even when trusted, some web browsers will say that the connection is not secure even when it is, this is due to the certificate not being verified by a third party. The connection is still https and the requests are encrypted.



An auto generated certificate from Flask's Werkzeug's adhoc.

The cert.pem and key.pem is generated by the command
`openssl req -x509 -newkey rsa:4096 -nodes -out cert.pem -keyout key.pem -days 365`

The certificate.crt is generated by
`openssl x509 -outform der -in cert.pem -out certificate.crt`

s341856, s341827, s341888

Prometheus/Grafana

We are using Prometheus and Grafana to monitor the activity on our website. Both Prometheus and Grafana are being run in docker, together with our app through Docker compose.

Our docker-compose.yml:

```
1 version: "3.5"
2 services:
3   flask-api:
4     build:
5       context: ./
6     restart: unless-stopped
7     container_name: flask-api
8     image: flask-api
9     ports:
10      - "5000:5000"
11     networks:
12      example-network:
13        ipv4_address: 172.16.238.10
14
15   prometheus:
16     image: prom/prometheus:latest
17     restart: unless-stopped
18     container_name: prometheus
19     ports:
20      - 9090:9090
21     volumes:
22      - ./prometheus.yml:/etc/prometheus/prometheus.yml
23      - ./prom-data:/prometheus
24     command:
25      - '--config.file=/etc/prometheus/prometheus.yml'
26     networks:
27      example-network:
28        ipv4_address: 172.16.238.11
```

```
172.16.238.11
2
3   grafana:
4     image: grafana/grafana:latest
5     restart: unless-stopped
6     #user: "472"
7     container_name: grafana
8     depends_on:
9      - prometheus
10    ports:
11      - 3000:3000
12    volumes:
13      - ./datasource.yml:/etc/grafana/provisioning/datasource.yml
14      - ./grafana-storage:/var/lib/grafana
15    env_file:
16      - ./config.monitoring
17    networks:
18      example-network:
19        ipv4_address: 172.16.238.12
20
21 networks:
22   example-network:
23     name: example-network
24     driver: bridge
25     ipam:
26       driver: default
27       config:
28         - subnet: 172.16.238.0/24
```

s341856, s341827, s341888

When running “docker compose build”, a container containing the app is built by using our Dockerfile, while the containers for Prometheus and Grafana are built using images that Docker pulls from the latest version online. The command ‘docker compose up’ runs the app on port 5000, Prometheus on port 9090 and Grafana on port 3000.

You can verify that Prometheus is running correctly by viewing the log in the terminal:

```
flask-api | 172.16.238.3 - - [19/May/2021 14:04:03] "GET /metrics HTTP/1.1" 200 -
flask-api | 172.16.238.3 - - [19/May/2021 14:04:13] "GET /metrics HTTP/1.1" 200 -
flask-api | 172.16.238.3 - - [19/May/2021 14:04:23] "GET /metrics HTTP/1.1" 200 -
flask-api | 172.16.238.3 - - [19/May/2021 14:04:33] "GET /metrics HTTP/1.1" 200 -
```

This shows that Prometheus is scraping the ‘/metrics’ endpoint every ten seconds (which is the scrape interval defined in the file ‘prometheus.yml’).

The data for Prometheus and Grafana is saved locally and stored even after the docker container is shut down. This is done by mounting a volume to the data path of the Prometheus and Grafana containers. This makes it possible to view historical data in Grafana.

We also have a setting file for both Prometheus (‘prometheus.yml’) and Grafana (‘datasource.yml’) which are passed to the respective containers in the docker compose file. Lastly, the username and password for Grafana is defined in the file “config.monitoring”.

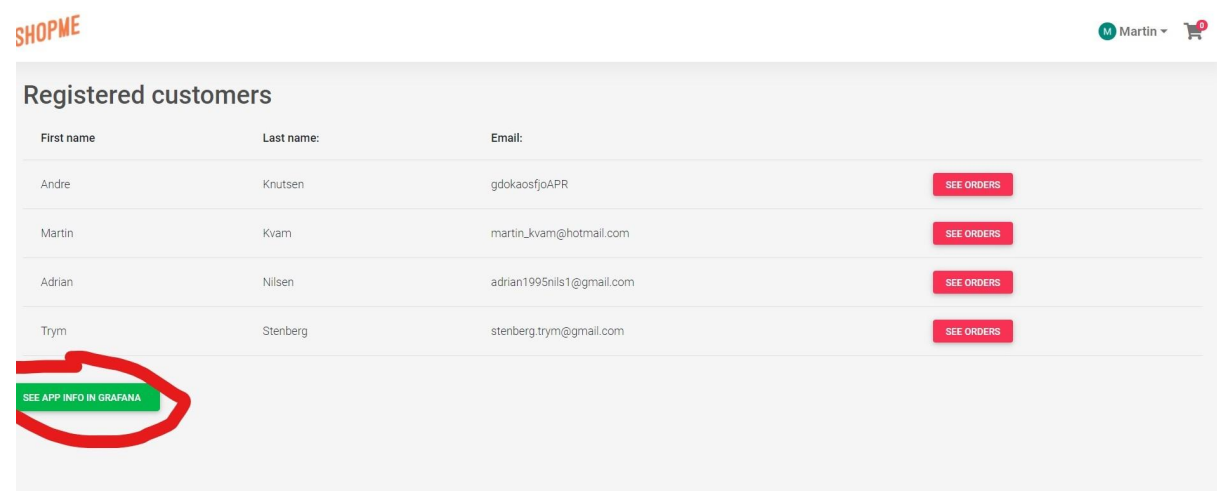
For our Grafana dashboard we took inspiration from a public repository on github (with approval from our teacher):

https://github.com/rycus86/prometheus_flask_exporter/blob/master/examples/sample-signal-s/grafana/dashboards/example.json

We made some minor tweaks to this dashboard to fit our app.

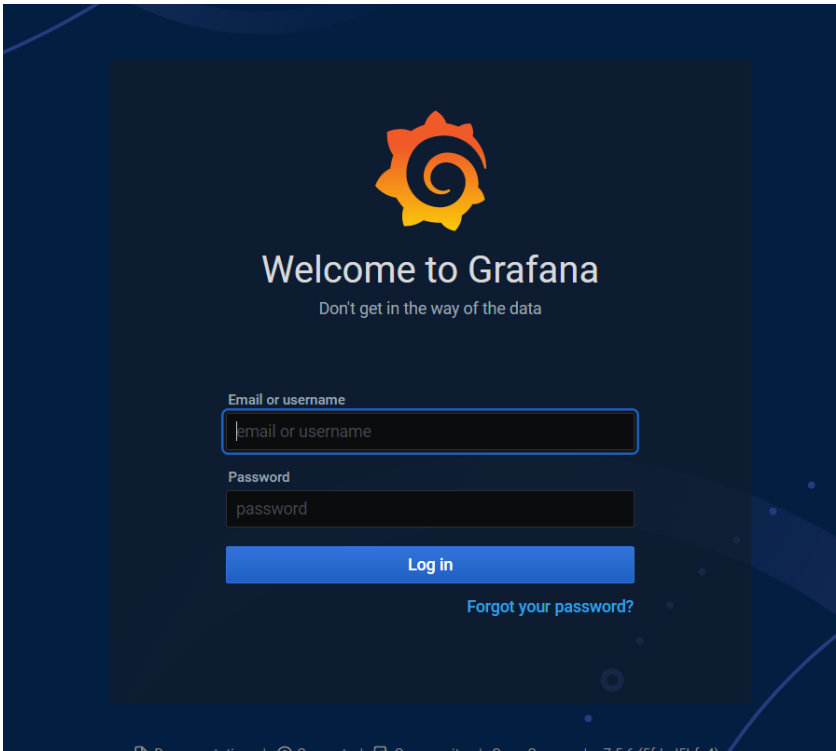
Viewing our dashboard in Grafana:

To view our dashboard you have to use the direct link which is located on the admin page of our website:

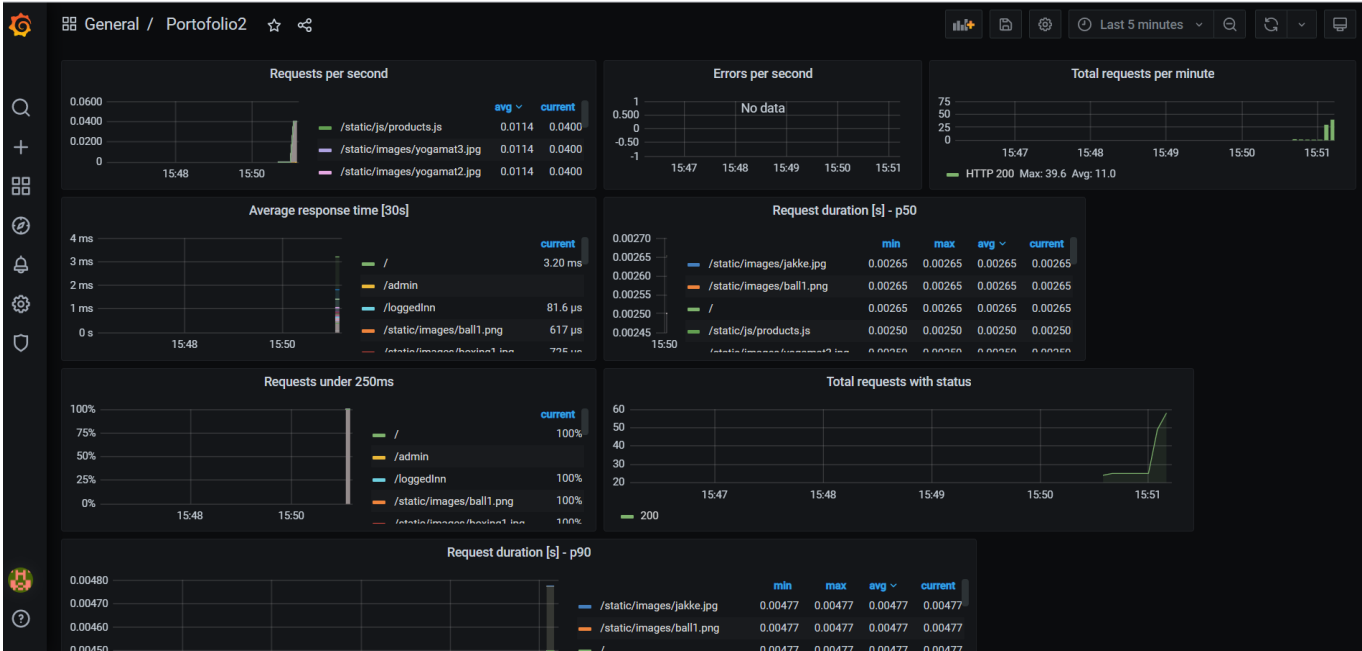


This takes you to a login page. The username is ‘**admin**’, and the password is ‘**admin123**’:

s341856, s341827, s341888



This takes you to our dashboard:



In the top right corner you can change the time interval to view historical data:

s341856, s341827, s341888

The image shows a web application interface with a time picker modal. The top bar has a red circle around the 'Last 5 minutes' button. The modal is open, displaying two sections: 'Absolute time range' and 'Relative time ranges'. The 'Absolute time range' section has 'From' set to 'now-5m' and 'To' set to 'now'. The 'Relative time ranges' section lists various time intervals, with 'Last 5 minutes' selected. A message box explains that recently used intervals will appear here. The browser time is shown as Norway, CEST, UTC+02:00.

Absolute time range

From: now-5m

To: now

Apply time range

It looks like you haven't used this time picker before. As soon as you enter some time intervals, recently used intervals will appear here.

[Read the documentation](#) to find out more about how to enter custom time ranges.

Relative time ranges

- Last 5 minutes ✓
- Last 15 minutes
- Last 30 minutes
- Last 1 hour
- Last 3 hours
- Last 6 hours
- Last 12 hours
- Last 24 hours
- Last 2 days
- Last 7 days

Browser Time Norway, CEST UTC+02:00 **Change time zone**