Quantified Student

Dashboard documentation

Thijmen Brand



# Table of Contents

[Table of Contents 2](#_Toc117165259)

[Version History 3](#_Toc117165260)

[Introduction 4](#_Toc117165261)

[Dashboard description 5](#_Toc117165262)

[Requirements 6](#_Toc117165263)

[Must have 6](#_Toc117165264)

[Should have 6](#_Toc117165265)

[Could have 6](#_Toc117165266)

[Wont have 6](#_Toc117165267)

[Non Functional 6](#_Toc117165268)

[Security concerns 7](#_Toc117165269)

[Acceptance testing 8](#_Toc117165270)

[System tests 8](#_Toc117165271)

[Acceptance test 8](#_Toc117165272)

[User acceptance test 8](#_Toc117165273)

[Software architecture 9](#_Toc117165274)

[Technical documentation 10](#_Toc117165275)

[Conclusion 11](#_Toc117165276)

[Literature 12](#_Toc117165277)

# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Comment |
| 11-10-2022 | 0.1 | Thijmen Brand |  |
| 20-10-2022 | 1.0 | Thijmen Brand |  |

# Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
|  |  |

# Introduction

This document will describe the processes, analysis and technical aspect concerning the front-end quantified student dashboard.

The quantified student dashboard is the place where students can visually see their performance with the use of graphs. This dashboard communicates with the quantified student back-end to get the information shown in the graphs.

# Dashboard description

As said in the introduction. The quantified student dashboard is the place where students can get visual information about their study performance. This performance can be measured by a lot of factors and other factors then can have influence on those initial factors. In order to visualise these factors, different data points will be plotted in graphs to compare against each other.

But not every student finds the same information useful for study performance. To accommodate for this difference in data interest can you pick what data sources you want to configure in the dashboard. You can for example configure your canvas account, weather information and attendance to contrast these. But you can also only add canvas to your account.

Besides that, can every student choose their over graphs they want to see on their dashboard. After you configure a data source, graphs for that data source will become available to add to your dashboard. This way every student can choose their own valuable information.

After students have chosen what information they want to see on their dashboard, they get the ability to move the graphs around and make them bigger or smaller.

On top of all individual graphs there is one overall performance graph that is always visible. And as described earlier, performance if different for every student. That is why you can select which data points contribute to the overall performance.

# Requirements

Every well structured software system must start with an idea. This idea is then turned into requirement the final software must meet.

|  |  |  |
| --- | --- | --- |
| Must have | |  |
|  | As a student I want to be able to see plotted graphs with information | |
|  | As a student I want to be able to decide which data source is relevant for me to configure in the dashboard | |
|  | As a student I want to be able to dis-connect a data source from my dashboard | |
|  | As a student I want to be able to choose which graphs I want to see on my dashboard | |
|  | As a student I want to be able to enlarge or shrink a graph to my preference | |
|  | As a student I want to be able to shuffle the graphs to personalise my view | |
|  | As a student I want to see an overall performance trend to see how I am doing | |
|  | As a student I want to define what performance is for me by selecting data points in the performance graph | |

|  |  |  |
| --- | --- | --- |
| Should have | |  |
|  | As a student I want to be able to see upcoming canvas deadlines | |
|  | As a student I want to be able to share my data with peers compare it with them. | |
|  | As a student I want to be able to choose what data points I share with my peers | |
|  | As a student I want to be able to revoke access for peers to my data | |

|  |  |  |
| --- | --- | --- |
| Could have | |  |
|  | As a student I want to see my feed pulse data | |

|  |  |  |
| --- | --- | --- |
| Wont have | |  |
|  | As a teacher I want to see the performance data of my students | |

|  |  |  |
| --- | --- | --- |
| Non Functional | |  |
|  |  | |

# Security concerns

The dashboard will contain a lot of sensitive information about the student. It will process many personal health data or study data. This is something we have to handle with care. This sensitive information will be served over an api and thus this api has to be very secure. The detailed API protection can be found in another document and in this document I will keep it in scope of the front-end dashboard.

One common mistake when it come to consuming an api is over fetching. This is when the api sends more information then necessary and that can be a security risk. In order to prevent this we use an GraphQL api where the api consumer can select what data it wants. This way the dashboard can only select the data that is necessary to plot the graph the user wants to plot.

Besides the api security risk there are also front-end only security risks like XSS and CSRF. It is very hard to keep out every possible attack, but there are a few things I can so to limit the chance of such an attack happening.

The first is taking the time to implement an feature properly. When you want to do it as fast as possible with minimal effort you are guaranteed to make mistakes which can be used by attackers to take advantage of (Paduraru, 2020).

The second quite obvious thing is to sanitise all input fields. Input fields are one of the most vulnerable places of a webapp because that way people can directly interact with your code (Paduraru, 2020).

And at third I will make use of the CSP (Content Security Policy) header. This header allows us to control the content that is allowed to load in. It can be used to specify approved origins for various content types (Paduraru, 2020).

# Technical documentation

<https://github.com/quantifiedstudent/Dashboard-v2/tree/develop>

## Graph documentation

The quantified student dashboard has a lot of graphs. To create and configure every graph by hand is not very scalable. So in order to make the process of adding and generating graphs easier I have made this a totally generic process which I will walk you through in this documentation.

Afbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst, monitor

Automatisch gegenereerde beschrijvingThere are a few components and classes to this graph generation process. The main thing is an array of ChartContainer objects which hold the charts. These chart containers are then being displayed in the modular grid. But there is one step above displaying the graph and that is the graph definition. In the dashboard you are able to select the graph you want to add to the dashboard. These different graphs are pre-defined by the class QsGraph

The static read-only properties are the predefined graphs with the necessary data. The DEFAULT property however serves as a loading state for when the data is not loaded yet.

The first is the label the graph should get.

The second is the graph type. These are defined in an Enum which is shown next to the QsGraph class.

And the third property is the api endpoint the data should come from.

So when you select a new graph to add to the dashboard you are selecting one or more of these underlaying classes. Which then will be provided to the chartContainer component.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

The chartContainer component is responsible to fetch the data from the desired graph(s). and pass it down to the actual chartComponent.

There will be an useState hook to keep track of the data the api has provided. But when there is not data yet, this will be the default graph I discussed earlier. Then the component will fetch the data for every graph provided and add for every graph an object to the returnValue array. This is to keep track which labels and data belong to which graph. After all the fetching is finished the state will be set to the fetched data in order to be used by the chartComponent.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

The ChartContainer component makes use of one other class called ApiRequest to make the actual request and to extract the gotten data into usable data. This ApiRequest class does two things at the moment. It is initiated with the endpoint the request is supposed to go to, and then there are two methods. The GetData method makes the actual api call and after it gets an response from the api it will provide that data to the extract class which is responsible for extracting the data into a usable form factor.

--code on next page--

Afbeelding met tekst

Automatisch gegenereerde beschrijving

After the ChartContainer has all the data it will be passed on to the ChartComponent which is responsible for displaying the actual chart. It gathers the data provided to it and passes it on to the QsChart class which on his turn is responsible for generating the chart.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

The QsChart class has a few properties.

Charts which quires an array of objects with chart, labels and data (the object that is generated by ChartComponent)

Labels, this is the X axis of the chart

And an canvas html element to draw the chart on.

Then this class generates for every chart a dataset with the label, data and type, It gives every graph another colour and generates the configuration for the chart. And after all of that it generates the chart with Chart.js and return it to be displayed on the screen

Afbeelding met tekst

Automatisch gegenereerde beschrijving

# Conclusion

# Literature

Paduraru, A. (2020, December 17). *How to master front-end security*. QUALITANCE. Retrieved October 20, 2022, from https://qualitance.com/blog/how-to-master-front-end-security/