IMAGECHASE

Documentation

Internet Technologies - COMP10020

Contributors

Jens Schlegel	B00485114
Jonas Leitner	B00398786

Website

http://imagechase.herokuapp.com/

1 TABLE OF CONTENTS

2 Table of figures 3
3 Introduction 4
4 Overview 4
4.1 Background 4
4.2 Core Functions 6
Overview of Competitions 6
Detailed competition information with join functionality 6
Overview of Submissions from Competition 7
Registration and Profile 7
Administration and judge functionality 7
4.3 Advanced Functions 8
Global Leaderboard and Prices 8
Corporate accounts 8
Improve UX with subcategories 9
4.4 Data Protection 9
5 Implementation 10
5.1 User Interface 10
5.2 Technology Stack 16
5.2.1 Virtualisation – Docker 16
5.2.2 Frontend – Nextjs 16
Backend – Nest.js 18
5.3 Data organisation 20
6 Hosting 27
6.1 Services 28
6.2 Scalability 29
6.3 Tracking and statistics 30
7 Conclusions 32
8 References 33

2 TABLE OF FIGURES

Figure 1 Photocrowd Competition Website 4
Figure 2 Life Framer Competition Website 5
Figure 3 Competition Overview 10

3 INTRODUCTION

This is a specification document that will lead you through the creation of a fictional competition website demanded by the business named IMAGECHASE. The software house for whom we work for demands a generic implementation so that the program may simply be customized to suit the needs of other potential clients. The purpose of this website is to establish a place where individuals could participate with diverse image competitions that are available on the site. The website is designed in a way that it facilitates users to enter for any image competition. To make its usage effortless even for admins and judges, we built an administration interface that allows to establish new competitions and enables the rating of submissions directly on the website. In the following, we will discuss the functions and features needed, how we implemented them and justify why we have chosen these design choices.

4 OVERVIEW

We have decided to create a competition website for images. As the competition scheme is very easy to transfer to another area, we are building our app in a very modular way. This makes it possible to rewrite a few components, if desired, to meet the new requirements. However, it is not necessary to reinvent the entire logic. This enables us to provide faster development for new customers who want to have a similar application.

4.1 BACKGROUND

When you launch your own product, it is essential to familiarise yourself with your competitors in order to learn from them and to discover gaps so that you can stand out from them. Through the analysis, we were able to work out important core functions that our competitors have already successfully implemented and adopt them for ourselves. It was also possible for us to work out functions that we think are missing and that would allow our product to stand out successfully on the market.

Photocrowd

A strong competitor that does a very good job and from whom we could learn a lot is the website from Photocrowd (https://www.photocrowd.com/).

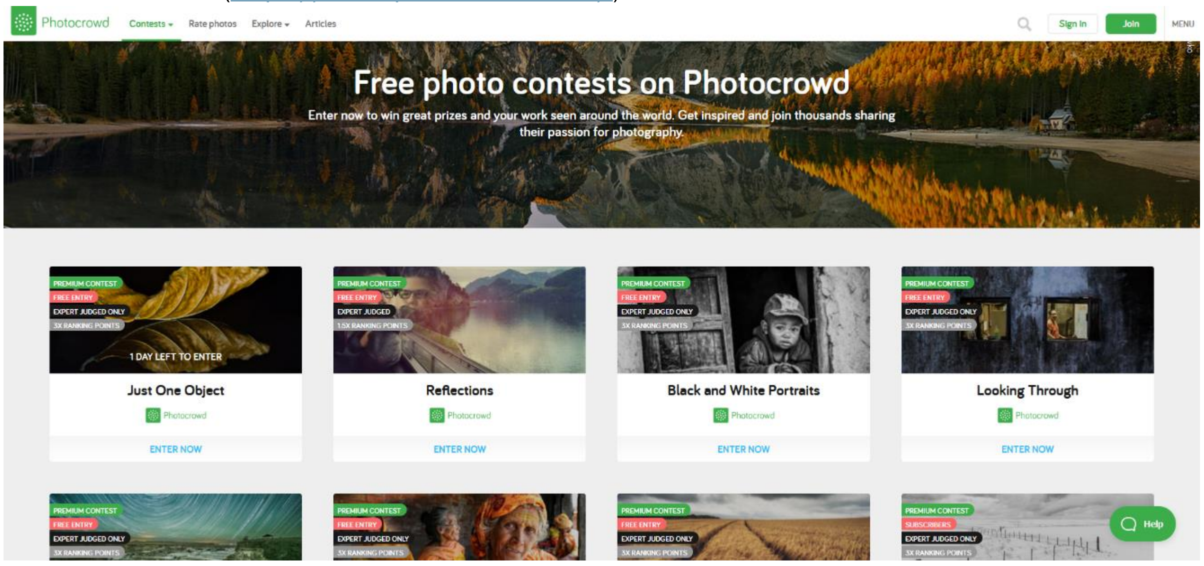
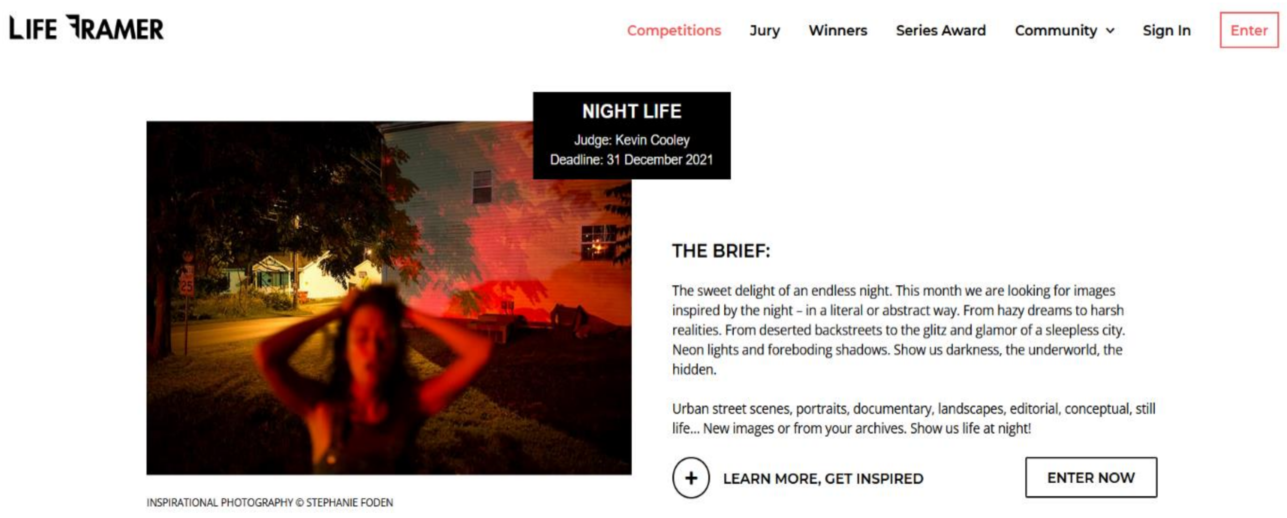


Figure 1 *Photocrowd Competition Website*

The website offers a good overview of all current competitions. It allows a very quick overview for choosing the competition that suits you and join it easily. There is also the possibility to filter by photo category. We find this functionality very helpful, but we would like to implement it in a more user-friendly way. Additionally in comparison to Photocrowd we would like to add a short description of the competition on the competition overview page. This helps users find the right competition even faster without having to click on it, pull out information on a second page, and then jump back to the overview. Photocrowd has set up the overview page of the competition in a well-structured way. You can find all the information you need at a glance and view the entries that have already been added. We also found this implementation of Photocrowd to be successful and have therefore oriented ourselves on it. The ranking system implemented by Photocrowd in which you collect points is a progressive idea. This motivates users to be more active. We would like to implement this feature as an additional function after the initial release (Photocrowd, 2021).

Life-Framer

Another interesting and educational competition site for us was Life Framer (https://www.life-framer.com/competitions/). It is a bit smaller than Photocrowd but has implemented some interesting features. Firstly, there is an info text directly on the overview page of the competitions. Via a toggle button it is possible to display further information. This is also an exciting approach from which one can derive a few small things. Personally, however, we think that you don't get an overview of all the competitions quickly enough on Life Framer. Nevertheless, they have implemented a very simple and modern design which is not too cluttered with information like the one from Photocrowd. Life Framer uses a business model where you must pay directly to submit your images. We are not convinced of this and would prefer to use a different business model (Life Framer, 2021).



The sweet delight of an endless night. This month we are looking for images inspired by the night – in a literal or abstract way. From hazy dreams to harsh realities. From deserted backstreets to the glitz and glamor of a sleepless city. Neon lights and foreboding shadows. Show us darkness, the underworld, the hidden.

Urban street scenes, portraits, documentary, landscapes, editorial, conceptual, still life… New images or from your archives. Show us life at night!

Figure 2 Life Framer Competition Website

Summary

By analysing our competition, we were able to identify some helpful core functions. We have taken the best features from both websites and put them together into ours. For example, we would like to take over the simple overview of the competitions from Photocrowd, but still add a small description like on Life Framer to get a little more information. So we'll take the middle way between the two approaches. In addition, we have been able to identify errors and are glad that we do not also make these. On both sides there is always only one judge for a particular competition. We don't think that's fair because the judge can be very subjective. We want to offer our users the most objective evaluation possible and have therefore implemented our data structures in such a way that there are several judges for a competition. Furthermore, we could observe that there are different photo categories but not really categories for, drawn pictures, 3D modelling or similar. Here we could discover a gap in the market and we see great potential if we further elaborate these missing categories and give them a platform.
In the next chapter, we will look at the core functions and later also at the advanced functions that we were able to work out.

4.2 Core Functions

It is important to clearly define which functions are necessary and must be implemented before the initial application can be released. These functions are part of the core of our application and are indispensable.

Overview of Competitions

Probably one of the most important basic functions of a competition website is the overview of existing competitions. If this is unstructured and not user-friendly, you are at a big disadvantage. Therefore, we have placed great emphasis on making our competitions overview as user-friendly as possible and to provide our users the quickest possible oversight compared to other websites. Our website also offers a platform for different categories, not only for photography. Thus, as an example, drawing competitions can be held. In order to get an optimal overview of the desired competitions, there will be a filter function according to categories. In addition, the competitions are sorted by start date. There is also a small description field of about 200 characters. This is the optimal length to show the most important key data. However, it is not too long and can be skimmed over quickly. All relevant information for the preselection of a competition is displayed in a clear 'card'. This includes title, type, description, start date and a cover image representative of the competition. To further improve the user experience and to make joining the competition as easy as possible, there is a button on this competition card that takes you directly to the submission form.

Detailed competition information with join functionality

After the preselection of the competition, the user is now naturally interested in the details of the competition. Firstly, the instructions are necessary to know, as are the rules of the competition. Furthermore, it is interesting to learn more about the judges. For this reason, a short biography of the judges is displayed. Information such as the start and end dates of the competition and the current participant count can also be found here. If the user has now decided to take part in the competition, we give him the opportunity to upload a picture in a submission form directly under the information. We have decided to give users the opportunity to add a small description to their submission. Here they can add their personal thoughts and if desired describe their submission a little bit. This is especially relevant since we don't just have simple picture competitions, but many different categories such as drawing your favourite anime character or 3D modelling your own virtual Metaverse character.

Overview of Submissions from Competition

Another feature that is relevant for us and that we have attached importance to is the overview of all submissions of a certain competition. If you want to join a competition or have already joined one, it is of course particularly interesting what other competitors have submitted. But also, for any other user, even if you are only interested and do not want to participate directly in the competition, it is very interesting to browse through the submissions. It is possible to see the picture with the corresponding user and his description. If there is already a rating, the rating between one to five is displayed. It will be possible to sort the submissions. For example, you can choose between the best rating or the most recently added. Another conceivable feature is to additionally enable an upvote option for normal users. This provides further interactivity between users, which can enhance the user experience.
Since the submission is visible to everyone, the user must have agreed to the GDPR-compliant conditions before submitting his submission.

Registration and Profile

Just about every internet service requires some form of registration these days. This process can sometimes be annoying, as you have gone through this process dozens of times before and each time you are asked to think of a new secure password. In addition, you always must give your private data and you never know exactly how trustworthy they really are. To make this registration process as user-friendly as possible and to increase the user's confidence that their data will be treated with integrity and confidentiality, we have decided to use Auth0. This is an authentication and authorisation platform. They take care of the authentication of users professionally and therefore definitely have more experience than we do. They also allow registration via Google or other recognised services. So, all you have to do is allow Google to pass on your data to Auth0 and you're registered. It's that easy and user-friendly! To be GDPR compliant, there will be an option on the initial release website to read and agree to the privacy policy before registering.

After successful registration, we give our users the possibility to view and edit their own user profile. All data stored by us can be viewed via the profile. The user can easily change this data, but also delete it if the user wishes to do so. This gives the user a good user experience and at the same time considers the GDPR principles. The user can display a small biography on his own profile. In addition, all competitions in which one has taken part are displayed with the corresponding submission. Furthermore, it will be possible for the user to decide whether he wants his profile to be publicly visible or whether he prefers it to be private. In this way, other users may have the opportunity to browse through other profiles and gather inspiration.

Administration and judge functionality

Another relevant point that should not be underestimated is the administration interface for judges and administrators. This target group works many hours with our application and it is absolutely necessary that it is fun for them and as user-friendly as the rest of the website. Otherwise, we might lose judges to competitors. Therefore, enough time is also invested in the restrictive pages to make them easy to use. For admin users, there is a form on which a new competition can easily be created. A title, a competition type, a description, the instruction as well as the rules can be specified. Validation of the data is also carried out directly to avoid errors. Here it will also be possible to select judges who will then later judge the competition. If you are registered as a judge and click on a competition, you will be offered the possibility to give a rating between one and five stars. Furthermore, there is a small text field with which it is possible to give the user personal feedback.

Most of the data such as competitions, the associated submissions, or which user has admin rights and which is a judge for a particular competition gets transmitted via a REST API interface with our own backend and is stored in a relational database. In order to store the user's secret data with an even higher level of security, we use the Auth0 authentication service. Since saving images can quickly take up quite a bit of storage space and scaling problems can quickly arise if you take care of this yourself, we want to manage the images using the Cloudinary service.
Our proof-of-concept page already shows most of the core functions described, which makes it easy to imagine the final initial application.

4.3 Advanced Functions

This chapter deals with functions that can be implemented after the release of the initial app. For example, to further improve the user experience, to increase the number of users by creating further incentives or to implement further business models to increase sales. The release application will of course be available in English. This will clearly attract the largest number of users. However, there are numerous other important languages with which we can greatly increase our user numbers after a successful implementation. Internationalisation of the application is an enormously important point for global success, and we would advance the internationalisation of the application in parallel with the development of further features.
To give us a further advantage over our competitors, we should make the app even more appealing by offering more incentives to our users. Empirical user tests should be carried out and the data obtained should be used to enhance the user experience by making changes to the user interface if necessary.

Global Leaderboard and Prices

If you hold a competition and there is something to be won for first to third place, this naturally offers a small incentive. But if you consider how many users could enter a competition, most users will go away completely empty-handed. With an increased number of users, this could also discourage many users from joining in the first place. To counteract this, we would like to create a points system with a leaderboard, in which incentives are created for everyone to participate. For example, it could be possible to get points multiplied by the final ranking of the submission or to get a certain number of points if you participate as regularly as possible. There are many possibilities here. Of course, you can share this score with friends and motivate each other. But so that the points are not just a fictitious number, it would be possible to award small prizes such as vouchers or similar with advertising partners when you reached a certain number of points. With this feature, we could significantly increase user activity and likely even increase the total number of users.

Corporate accounts

At the moment it is only possible for an admin to create a competition. This competition will then be held directly by us as IMAGECHASE. This means that at the moment, companies can of course contact us and we will enter the competition into the system for them. However, this is of course not a sustainable solution. It would be much more effective if companies had their own account. Here they can write something about themselves, add a logo and create their own competitions. A label of who is holding the competitions can be added to the information. Companies can have many reasons for holding competitions for images. For example, they may be looking for a new image for their website or a new design for a specific product. By holding a competition and rewarding the winner accordingly, they get many different suggestions and can choose one directly. By giving companies the opportunity to run competitions on our website, we can of course be remunerated. There are various conceivable models, for example, a certain fee per competition created or a classic subscription model. By implementing corporate accounts into our application, we can create a completely new revenue stream. In addition, companies could indirectly increase our visibility by promoting their competition. These two points make us more marketable.

Improve UX with subcategories

Over time, more and more competitions will be added to our platform. Obviously, we are very happy about this, but at the same time we must think about a solution how we can still offer our users a good and quick overview of the various competitions. We already have a sorting function for categories like drawing, photography etc. But these can also be subdivided again. For example, for photography we can create subcategories like, black and white, portrait, nature or similar. For drawing, categories such as sketch, acrylics or watercolours are possible. It is very important to make sure that the UX does not suffer when an application grows. Often more and more functions are added, but users do not use them properly because they lose track of them. It is important to us not to make this mistake, which is why we pay close attention to the UX and also plan user tests at certain intervals.

4.4 Data Protection

One of the important things that a company should consider is the user's data and their privacy. If this is not respected, companies often have to pay high amounts of compensation. And in addition, the company loses the trust of the users, which leads to customers deleting their accounts. To ensure that this does not happen, we have thought about the security of the user data from the very beginning. We have decided that it is safest to have an external provider take care of the security of the user data. They know a lot more about security than we do, have a better secure infrastructure and have been doing it for a long time.

The external provider we entrust with the data of our users is Auth0. They store the data of many well-known companies, such as AXA, SIEMENS, AMD, and Pfizer and never had a data protection case. Because of the interface of Auth0, it is very easy to integrate secure authentication into the application. In addition, further functions are available for protection against an attack. One of these, for example, is the Suspicious IP Throttling function. This means that measures are taken in the event of too many incorrect login and registration attempts originating from a single IP address. Another function is brute force protection. This prevents it from being possible to log in from many different IP addresses at the same time. To provide users with even better security, you can also simply add multi-factor authentication through Auth0. We haven't added this feature yet, but it would be a feature that could easily be added in the future. (Auth0, n.d.)

It is also important for users to know what happens to their data. Where they are stored? Who all has access to it? How long they are stored there? Are they automatically deleted? In order to answer all these questions, we have linked to the **Terms of Service** and **Privacy Policy** in the footer on our Sign-up page.

What features in the app design and implementation will allow it to comply with GDPR?

- GDPR features
- Nature of personal data
- Design for consent
- Users can see their own personal data
- Processing and Deletion of Data
- Data security
- Authentication

5 IMPLEMENTATION

5.1 USER INTERFACE

We used the vector graphic program Figma to design our website and bring our ideas on a mock-up down. Because designing the user interface in a vector graphic program is much quicker than programming it directly. And you are also often not satisfied with the first attempt anyway, so you have to revise it a few times until it fits. Since Figma was also developed for working in teams, we were able to work live in the same document and exchange ideas about the designs. We started with the home page and visualise our different ideas for it. We made relatively quick progress, as we both wanted to keep the website relatively simple.

The first part was to design the navbar. Here we kept to the usual design of a navbar. At the top left is our logo, to the right our pages, competition, leaderboard and prices and on the right a button to log in or out. Below we wanted to show our users an overview of the current competitions that are taking place. Since we wanted to awake the interest of the users already on the home page. Therefor we designed cards for the competitions to display them. We didn't want that a card takes too much space, as the user should see a collection of competitions. We then placed the image at the top of the card and centred the necessary text and a join button underneath.

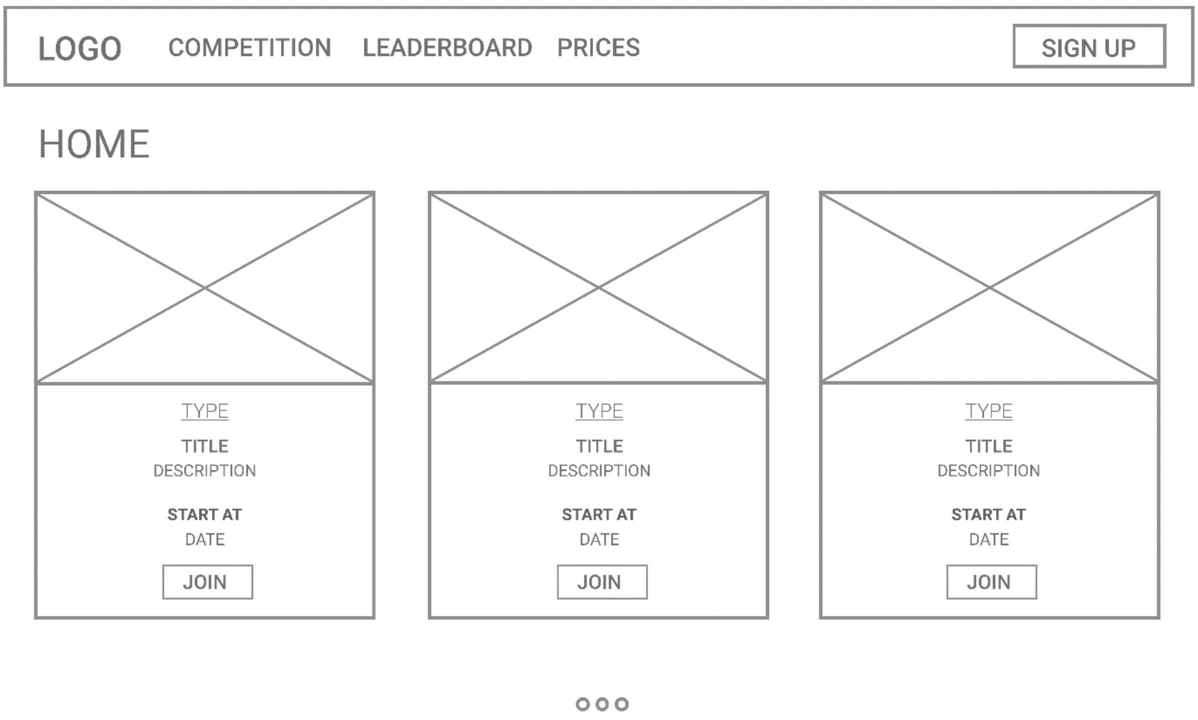


Figure 3 Home page

This is our result for the home page. This way our users have a good overview of the current competitions that are taking place, and the users can join them directly or have a look at the already submitted submissions by clicking on the card. Then you will be redirected to the page of the selected competition.

Here on the page are the most important information about the selected competition in the box at the top. The circles with the letters represent the judges who will review the submissions for this competition. If you point the mouse cursor at a judge, a pop-up window will open with the judge's full name and bio. Clicking on a judge should later take you to their profile. This feature is not yet implemented. Below you can see all the submissions that have been submitted. Here it was important to us that the image is seen as large as possible. That's why we designed the card horizontally, so that a submission card takes up the entire width. On the left half of the card, we have placed the image and on the right side we placed the text of the submission.

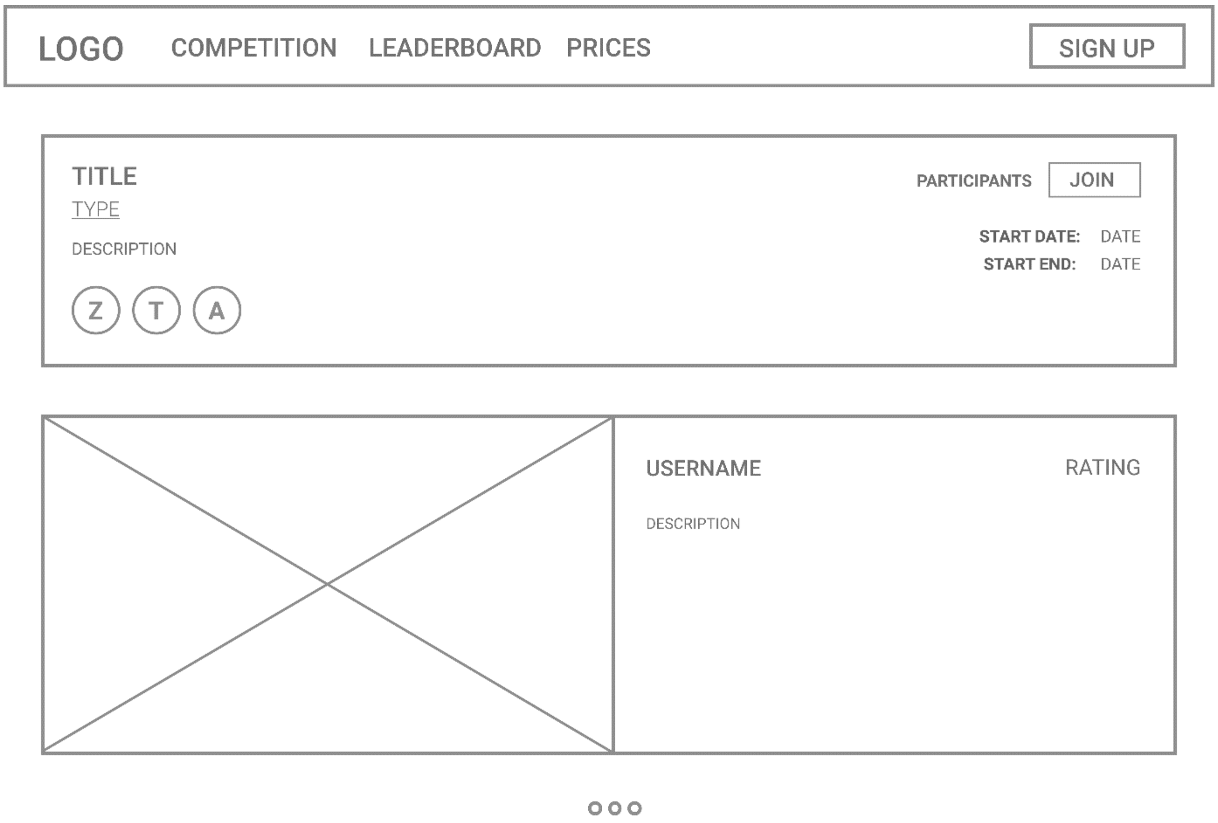


Figure 4 Selected competition

Here is the result as we imagined it. Here in the competition box, we have again the same join button that we had on the home page in the competition cards. When you click on the button, you are redirected to the join competition page.

Here we have another competition box, but this time it is more detailed. So instead of a description we have an instruction and a rules description, which gives the user all the important data for his submission. Under the competition box we have everything the user needs to create his submission. Here we have a field where the user can upload his image, and an input field where the user can add a description for his submission. And a button so that the user can submit his data.

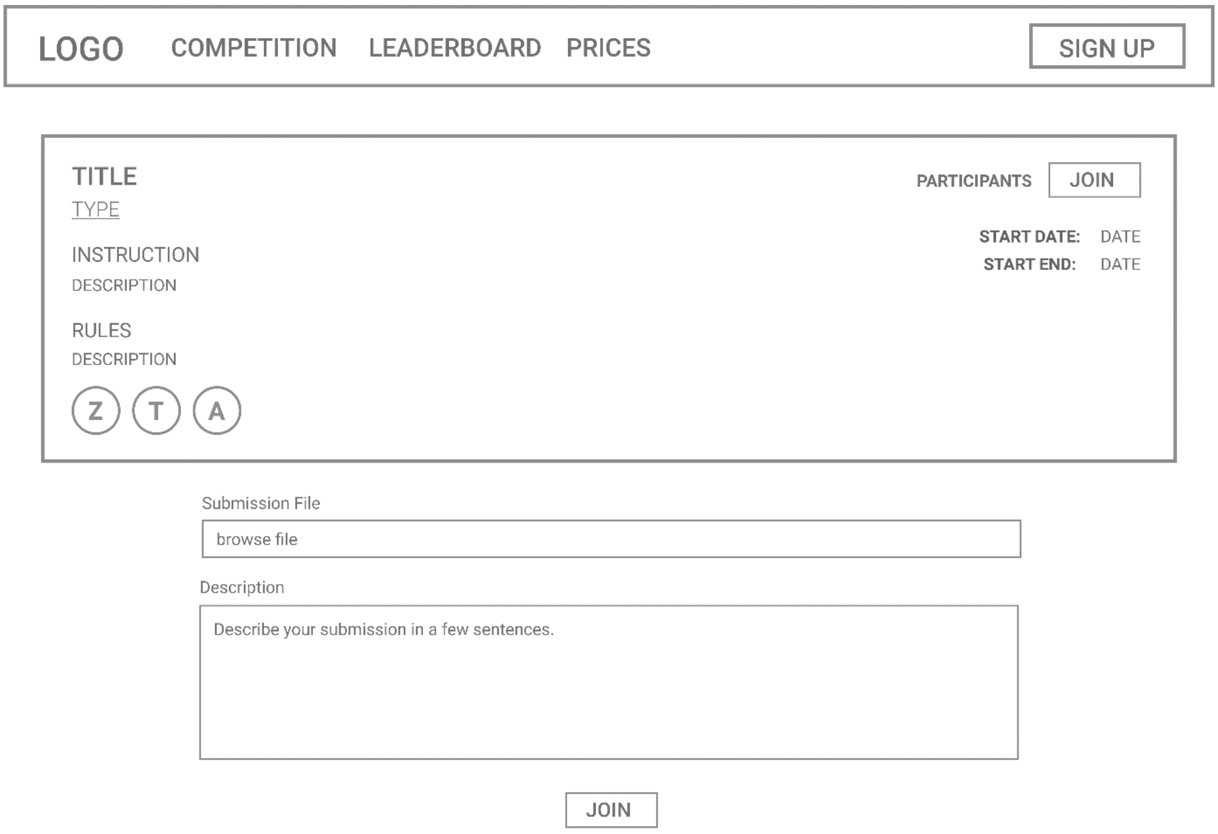


Figure 5 Join competition

For a normal user it was already all the important pages. But if the user is an admin or a judge, it looks slightly different.

The administrator has a create competition button on the home page that takes them to a page to create a new competition.

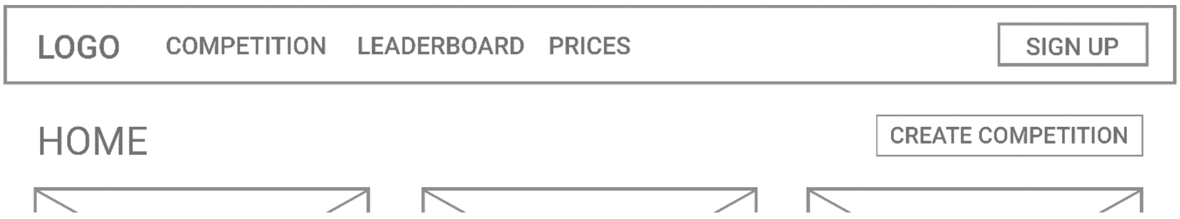


Figure 6 Admin home page

On this page the admin will find many input fields to add all the data to the competition. Also, a date picker to set the day for the start and end date and a button to submit the competition.

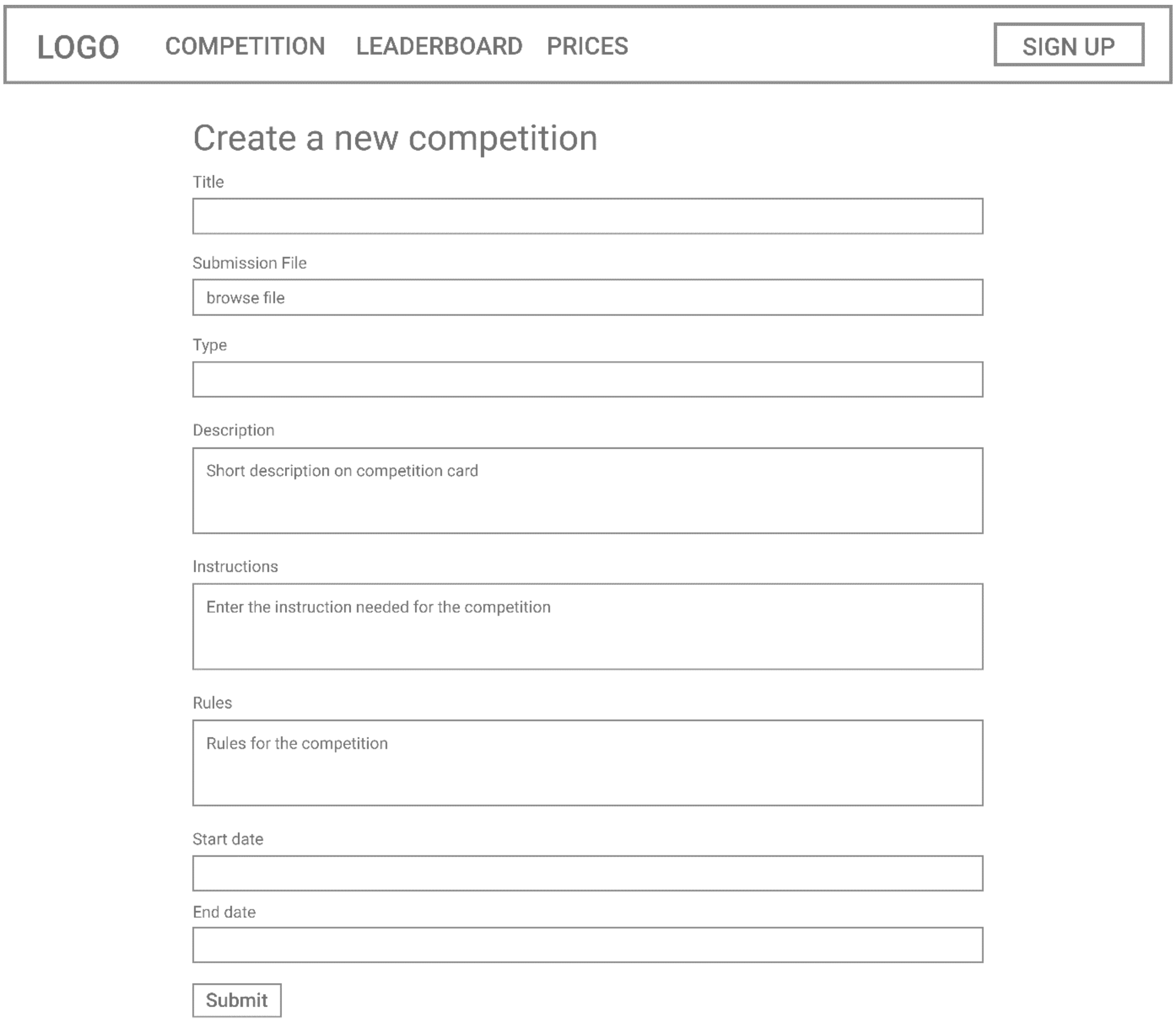


Figure 7 Create competition

If the user is a judge of a competition, then a field appears on the submission cards where the judge can rate the submission. In order to rate, the judge must choose how many stars he wants to give the submission and can also leave feedback.

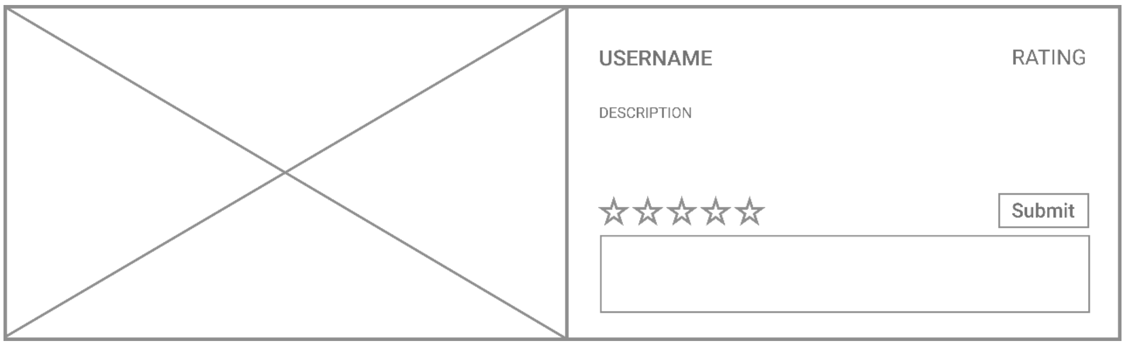


Figure 8 Rating

Figma was a great help in realising our design for the website. By designing and discussing the pages at the same time, we were able to avoid misunderstandings and improve our design for the website. Also it helped us to be more efficient, as we did not have to consult and it was clear how the pages had to be implemented.

Before implementing the design, we asked friends what they thought of it and did small usability tests. Because it is important to us that the users find their way around and that it is clear where they can find what. This way you can find weak points in the user experience before implementation.

Sitemap

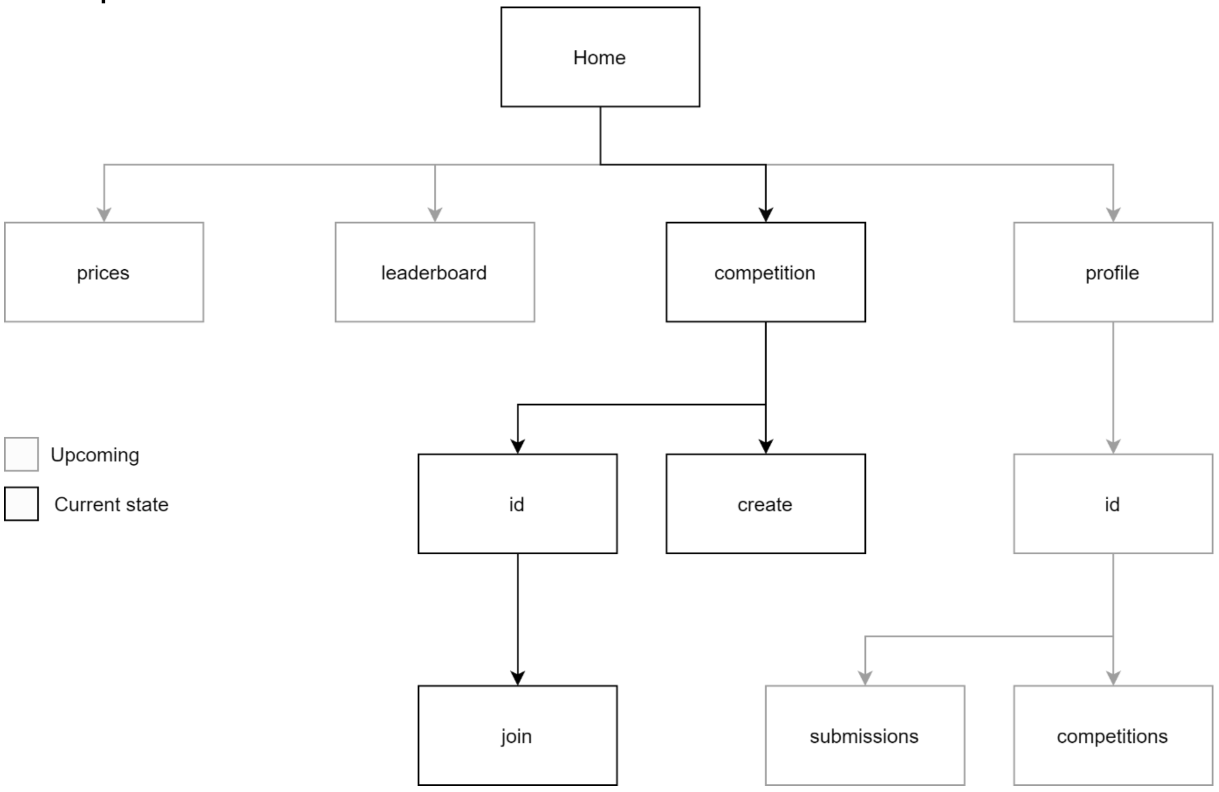


Figure 9 Sitemap

The user is on the home screen. He selects a competition and then lands at the url at:

```
https://imagechase.herokuapp.com/competitions/:id
```

Under the selected competition the user can add a submission. To create this submission, the user goes to:

```
https://imagechase.herokuapp.com/competitions/:id/join
```

If the user is an admin, he can create a competition at this address:

```
https://imagechase.herokuapp.com/competitions/create
```

The other addresses are currently not implemented but are already planned to be implemented. You should be able to see the competitions and submissions created by other users. Furthermore, it should be possible for the user to edit and delete his competitions and submissions on their profile.

5.2 TECHNOLOGY STACK

○

Type	Framework
Frontend	Nextjs
CSS	Bootstrap
Backend	Nestjs
Database	PostgreSQL
ORM	Prisma
Authentication	Auth0
Data fetching	SWR

5.2.1 Virtualisation – Docker

- Docker virtualisierung....

5.2.2 Frontend – Next.js

React is a powerful frontend framework. It is the largest in the world and is maintained by Meta. That's why we decided to use React for this project. But React has some downsides, like the pages are not generated statically. As a result, the performance of the website is not so good, which has a bad impact on the user experience. This also decreases the SEO score of the site. Since the pages were not generated statically at build time, the data must first be fetched on the client side to display them. That's why we decided to use the react framework next.js from Vercel, which includes features like SSG (static site generation) and SSR (server-side rendering).

It is clear that with our site users are always creating new content and we cannot rebuild the site every time so that the static pages can be created for the new content. Therefore, we use a connection between getStaticProps and getStaticPaths to get the data for the competition information box that you see when you select a competition.

GetStaticPaths ensures that we can preload each selected competition. This is done by getStaticPaths to fetch all competitions and declaring each id of a competition as a path. When a new competition is created, a new path is also created. GetStaticPaths then automatically generates the path for the new competition. We then pass these paths to getStaticProps. This way we have the ids for each competition in getStaticProps and can then fetch for each competition. It then fetches the information for each competition. GetStaticProps is called when the page is created or with revalidate when a request comes in. This will statically create the competition information box. (Next.js - Data Fetching, n.d.)

But Nextjs also has other advantages. One of them is the pages folder. To create a new page, you just have to create a page inside the pages folder. This page is then available with the name of the page over the URL. With the Nextjs link component you can also easily link the page. With React you have to install the react-router package. (Evans, 2020) Especially for developers it is good that Next.js has implemented a Rust compiler. Through this refresh and build times are much faster than with React.js. This saves time during development. (Next.js - Version 12, n.d.)

Also, the Nextjs image component is very useful as it contains built-in performance enhancements. This has an impact on the SEO score. Because images only load when they appear in the user's viewport. And blur-up placeholder can also be applied to the images through the image component. This allows to display the image directly with blur, and then load it to make it sharp. (Nextjs - Image, n.d.)

CSS - Bootstrap / react-bootstrap

Bootstrap is one of the biggest CSS frameworks. It is mobile-first, which means you develop the components for smartphone sizes first, and then for the other sizes. This is important for the responsiveness of the website, as you should always get the best experience regardless of the screen size. In className you can use the bootstrap variables to style the components, or you can also use the style attribute to write CSS in it to style the components. To efficiently develop a beautiful and user-friendly website, we used components from the UI library react-bootstrap.

A frequently used component was a button. Here the variations primary and outline are available.

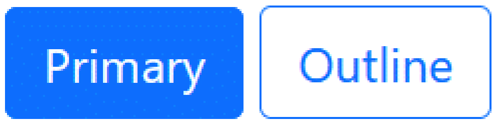


Figure 10 Primary outline button

These components have implemented a state. This means that when I hover over a button or click on it, the appearance of the button changes. This is important for the feedback of the user, so that he knows that his action has been registered.

Hovered state:

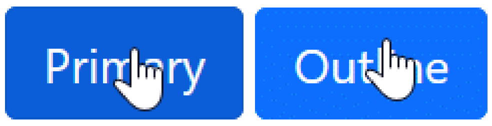


Figure 11 Button hover state

Here you can see that the colour of the primary button has darkened. And the outline button has filled with colour and the text has turned white.

Pressed state:

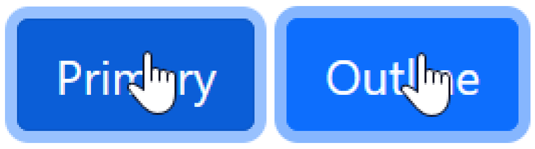


Figure 12 Button pressed state

If the button is pressed now, it gets a border.

Of course, the button was not the only component we used. We also used our navbar, cards, badge, and forms for the input from react-bootstrap. (react-bootstrap.github.io, n.d.) We then adapted these imported components to create a unique look for our website.

Backend – Nest.js

Express is a powerful Node.js backend framework. Express gives the developer a lot of freedom to decide which tools, middleware, and hooks to use. It is also open to the developer how they want to structure their code. Since Express does not come with a specific template engine. This often leads to the project structure not being beautiful.

Nest.js uses under the hood the node.js framework express. Unlike Express, you are much more restricted in Nest.js. It uses design paradigm and certain tools. Also, it guides developers to code in a certain way. It takes care of error handling and to reduce boilerplate code by wrapping the controller with a try-catch block, parse the request body, adds error hander, middleware, logger, etc.

Also, an advantage is, that it comes out of the box with typescript. Typescript give JavaScript strict typing, which is helpful in order not to develop bugs. This saves time that doesn't have to waste on debugging.

Nest.js has many decorators. You have decorators for the controllers, HTTP verbs, param, etc..., which implements the corresponding functionality under the hood and helps to make to code clearer what it does.

These all features come out of the box in nest.js help us to finish the project faster, with code that is not error prone. (Rahman, 2019)

Database – PostgreSQL

For our database we wanted to take a Relational Database Management System (RDBMS). One fundamental advantage was the referential integrity, which ensures accuracy and consistency of the data. This is because RDBMS use primary and foreign keys, for the relations to tables. This ensures preventing the deletion of a related record without first removing the primary record. (Pattinson, 2020)

For us, a clear structure in the database and consistency of the data is very important.

Therefore, as relational database only MySQL and PostgreSQL came into question. Since we use Heroku for hosting and they prefer PostgreSQL over MySQL and offer operational advantages for its use, we decided to use PostgreSQL for our RDBMS. (Hristozov, 2019)

ORM - Prisma.io

It is always a good idea to use an ORM or an ODM. With an ORM, you don't have to write SQL queries, which makes it much easier. This speed up development, which means shorter development times and lower development costs. (JHarley1, 2011)

ORM comes also with many useful features, such as automatically connecting to the database, transactions, migrations, seeds, streams, or other useful features. These features also contribute to a good developer experience and efficient work.

With an ORM we use the language of choice to interact with our Database, this way we can stick to one language and don't have to switch to SQL. (Hoyos, 2019)

Since our backend is written in Typescript, the language we use to interact with our database is Typescript. The ORM Prisma is well suited for this, as it is an ORM specifically for TypeScript and Node.js. This gives us type security when writing database queries in Prisma.

We can create easily new migrations with a cli command, or even try out our new schema with a cli command.

And with Prisma Studio, we always have a visual insight into what our database looks like right now. (Prisma, 2019)

Prisma makes it very easy for us to integrate with our PostgreSQL database and helps preventing errors.

SWR

We used a hook called useSWR to fetch our data. This hook has wrapped our fetch request. This provides the components with a continuous and automated stream of data updates. SWR comes with a few more features. Among them, built in cache, real time experience, SSR / ISR / SSG support, polling on interval and revalidate on focus. (swr.vercel.com, n.d.)

We use it to fetch a lot of things like the competitions and submissions. We have noticed that with useSWR we can give our users a better experience. When data is changed, it is instantly reflected in the user interface.

5.3 Data organisation

Table name	Attributes	Type	Decision
User	id name admin bio sub createdAt	serial varchar(100) boolean text text timestamp(3)	If the user wants to contribute, then he/she must be logged in so that the activities can be assigned to the user. In addition, important things such as the name or the sub must be saved for each user.

Competition	id
	title
	content
	type
	description
	rules
	instructions
	userId
	startDate
	endDate
	createdAt
		serial
		varchar(50)
		text
		varchar(30)
		varchar(200)
		text
		text
		integer
		timestamp(3)
		timestamp(3)
		timestamp(3)
			For a competition platform, the competitions play an important role. Therefore, important things such as title, description, instruction, etc… must be stored for each created competition.

Submission	id content description userId competitionId createdAt rating	serial text text integer integer timestamp double precision	Since we have competitions, we also need submissions. These make the competition exciting and let the users take part in it. Through the competitionId we can assign the submissions to the correct competition.
Rating	id feedback userId submissionId rating createdAt	serial text integer integer double precision timestamp(3)	To motivate users with prizes, the rating is essential to decide which user deserves the prizes. In order to be able to assign all the ratings to a submission and to know from which user the rating comes, we need this helper table.
Judge	id userId competitionId	serial integer integer	To know which users can rate which competition, we also need a helper table for Judge.

ER diagram

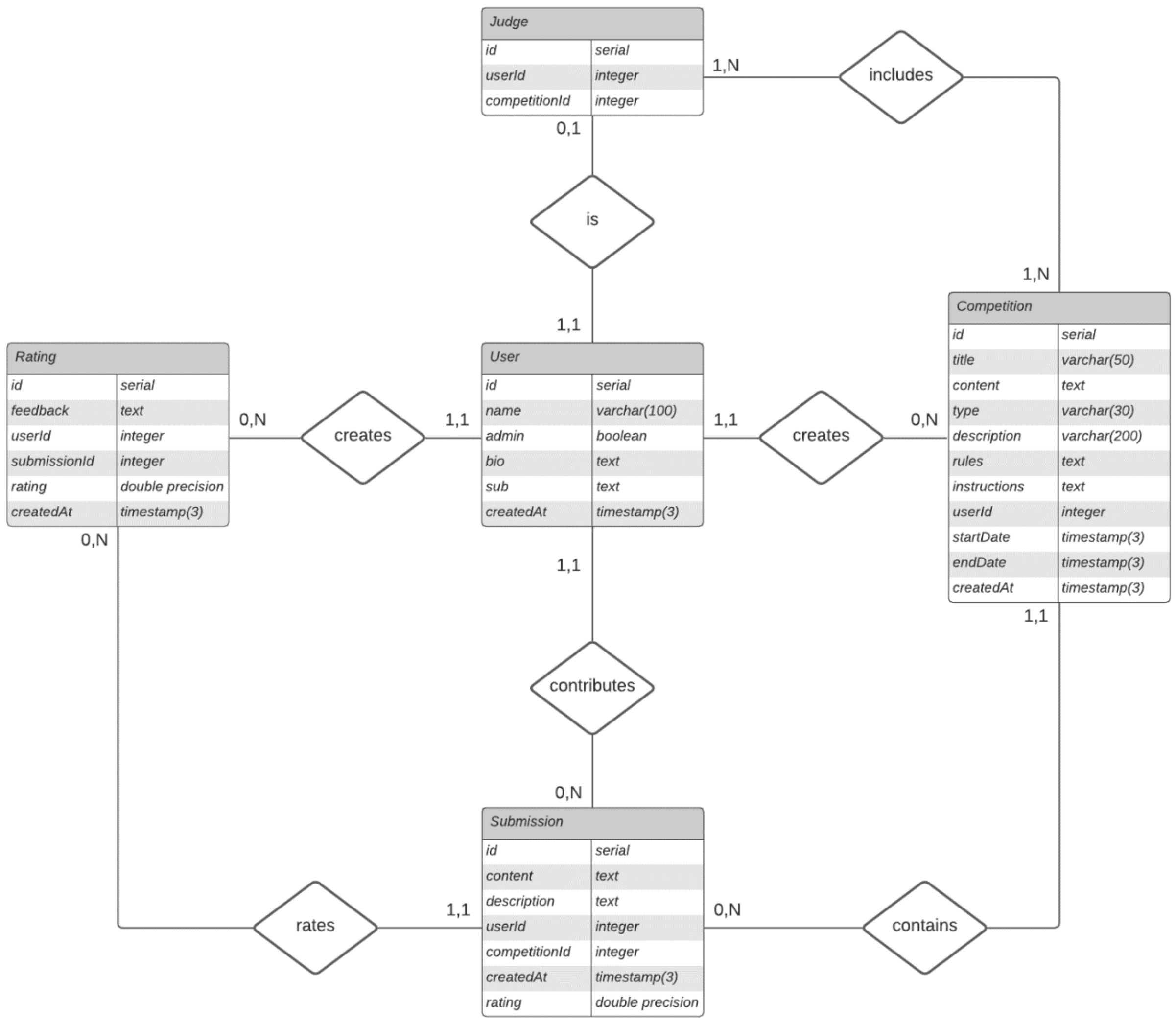


Figure 13 ER diagram

First, we give each file entry an id for a cleaner data structure. In our data structure, the user plays the central role. The user is connected to all tables. So, Competition, Submission, Rating and Judge always have a userId, so that we can connect the corresponding user to it. Our user has a Boolean attribute admin, if this is set to true then it allows the user to create a new competition.

For the judge, we created the helper table judge, so that a user can be assigned to the corresponding competitions where he/she is a judge. A judge entry is then created for each competition and user. This allows to assign users as judges for a competition, who can than rate the submissions.

If a user is not an admin or a judge of a competition, then it allows them to create a submission. When creating the submission, the user automatically joins the competition and is no longer allowed to make another submission for this competition. So that we can ensure that each user only submits one submission per competition. But it is possible for a user to revise their submission if they are not happy with it.

Selected query language

We have chosen the well-known REST API as the interface for our frontend and backend. This allows us to query, send, update, or delete data at our defined end points in the backend. We send JSON as the data type for communication, which also makes it easy to modify the response to add more fields. The information that arrives at the endpoints is then processed by Prisma, which then updates our database. A useful advantage of REST is that it has caching built in. This means that when a user goes to the website, all competitions are loaded with a GET request. If the user then selects a competition and then goes back, the information is then there much faster because it is still in the cache. (REST API Tutorial, 2018) Another advantage is that REST takes advantages of HTTP, that's why we don't have to install additional package. (MuleSoft, n.d.) This keeps the size of the installed packages smaller.

REST API Endpoints

Controller	API Endpoints	HTTP Verb	Purpose
users/	:sub	GET	To get the data of the logged in user.
	:id/submissions	GET	To see a history of submissions from the user on their profile.
	:id/competitions	GET	So that the admin user can view his created competitions.
	:id	PUT	Allows the user to revise their bio. Or change their username.
		POST	To participate in submissions, the user must first create an account.
	:id	DELETE	A user's privacy should always be respected, so the user should have the option to delete their account.
competitions/		GET	To get all the competitions to display them in a list on our home page.
	:id	GET	To get the data for a selected competition.
	:id/submissions	GET	To get all submissions to a specific competition. When selecting a competition.
	:id/judge	GET	To get the judges for a competition to display them in the competition information box.
	:id	PUT	To let the admin, update the title, type, description, rules, instruction or start and end date of their competition
		POST	To create a new competition.
	:id/judge	POST	To add a Judge to a competition.
	:id	DELETE	To delete a competition
submissions/	:id	GET	To get the data of a single submission.
	:id	PUT	If the user wants to revise their submission, for example to add something to their description or change the image.
		POST	Allows users to participate in the competition by submitting an entry.
	:id/rating	POST	For a Judge to create a new rating for a submission.

	:id	DELETE	If the user is not happy with his submissions, then he is still able to delete it.

Which endpoints need which data

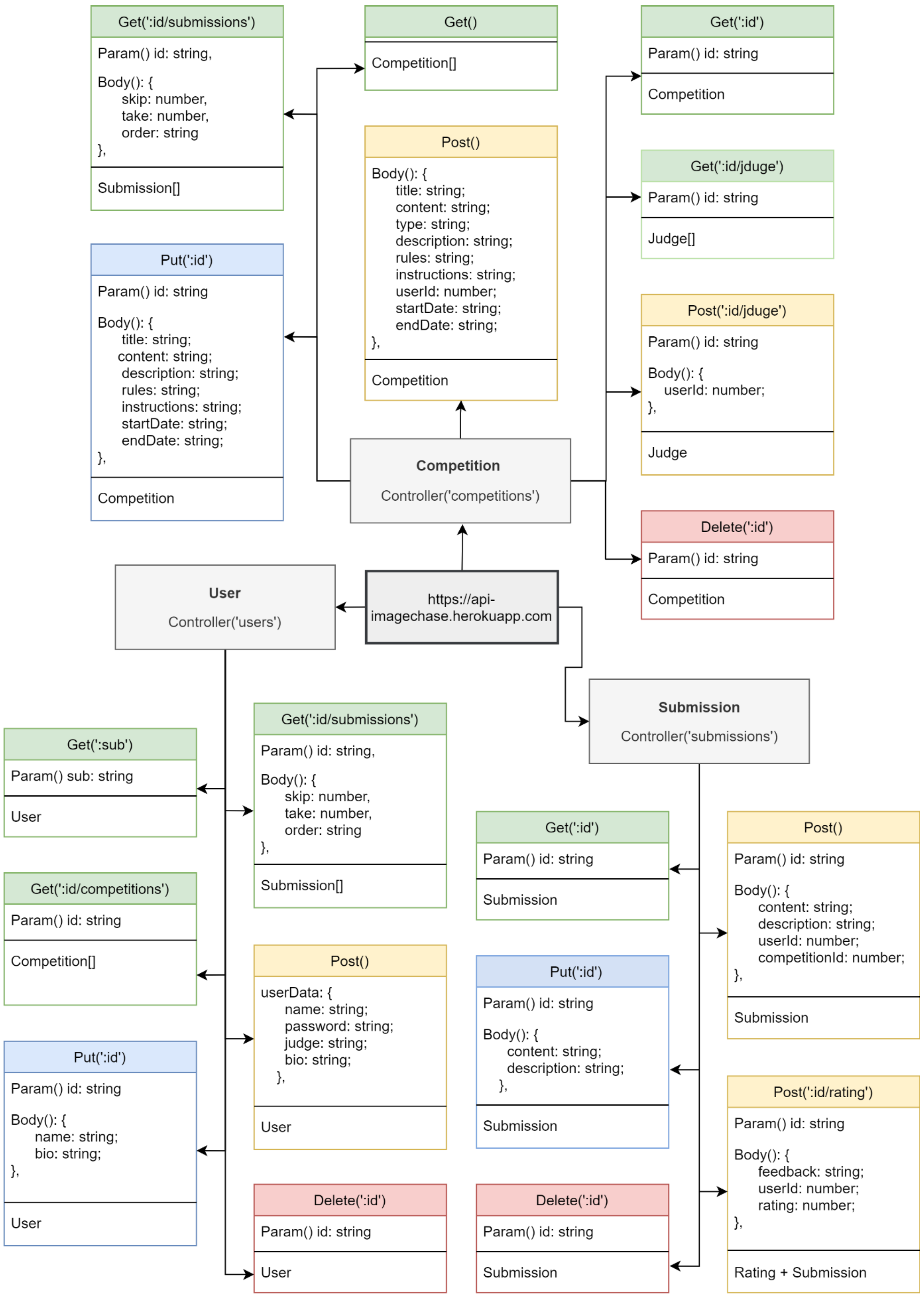


Figure 14 API endpoints

When a get request is sent to our controller competition, it responds with all competitions. However, if you only want a specific competition, then you can add the id of the competition into the query to receive only this as a response. The same can be done with the other controllers. Or a post request can be sent to the controller to create an entry. However, it is important to note that the required fields are included in the body of a post request. Otherwise, the post request is not possible. If something was posted by mistake, it can be deleted with a request to a controller and with the id in the query of the post. The same request URL can also be used for a put request to update entries. Here is it again important that a body with the updated fields is included in the request.

How will an API be arranged to access the data

Now, to get the submissions from a particular competition, we need to know the id of the competition. The first competition created has the id 1 and increases by 1 with each created competition. If we now take the 1 as the id, then the get request looks like this:

```
https://api-imagechase.herokuapp.com/competitions/1/submissions
```

And the response to the get request looks like this:

```
[
	{
	"id":1,
	"content":"Here is the url of the uploaded image",
	"description":"This is the description text of a submission",
	"userId":1,
	"competitionId":1,
	"createdAt":"2021-12-07T00:37:40.509Z",
	"rating":0,
	"user":{"name":"jens.b.schlegel"}
	}
	…
]
```

The submissions and competitions are sorted by createdAt by default, so that the newest ones are always at the top. But this is not always helpful, so we have implied that you can include a body in a get request that allows sorting the submissions by rating. Then the best submissions are displayed at the top. In order for this to work, a field called order with the value votes must be entered in the body:

```
Body {

    "order": "votes"

}
```

Order is not the only option that can be used in the body. Take is another useful function. With take we can determine how many submissions are returned to us. With take and the value 3 we get the 3 best submissions back and thus we know the winners of the competition.

Another option is skip. The name describes pretty well what this option does. It allows us to skip submissions, which is useful when we display 20 submissions on each page. When we are on the 4th page, we can skip the first 60 submissions with skip and then use take to fetch the next 20 submissions to display them.

API testing

To test the API endpoints, API platforms are quite helpful. These allow manually entered data to be sent to the endpoints. This is important to check what is responded to these requests. This is a good way to see if everything goes as expected, or an error occur. We used the platform Postman to test our APIs. For each controller, we've created a collection with the matching endpoints. This made it easy for us to create post requests to create a user or a competition, as well as to retrieve or update data. This was useful because we knew when we integrated the endpoints into the frontend that they had to work. As we tested each endpoint extensively beforehand. Postman has definitely helped us to save some time, as we have seen the response to our request directly. And we were able to test every single endpoint directly after implementing it in the backend.

It was also very useful that we could already display competitions and submissions in our user interface without being able to create them on the site. Because the first pages we developed were the start page and the submission page and at that time the pages for creating competitions and submissions were still missing. Through postman we could create already all the data to display the content on our site. Also functions like deleting competitions, submissions and users are not yet implemented on our site. However, as this is already implemented in our backend, we can use postman to delete entries and keep our data structure under better control.

6 Hosting

Nowadays, it is easier than ever to deploy your own application. You don't have to spend a lot of money to buy your own hardware and hire an administrator to configure it. Thanks to the numerous cloud services available today, the deployment process has become much easier, especially for small businesses and start-ups. However, you first have to get an overview of the numerous cloud services on offer, understand them and finally decide on a particular one.

6.1 Services

If you want to deploy your own application via a cloud service, you will quickly come across the terms Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). Both options are widely available and there are many providers to choose from.

IaaS allows companies to buy computing resources on demand without having to buy the hardware themselves. You still have the ability to access your server and storage directly, but you no longer have to worry about the physical management of the hardware. You usually get access to the resources via a dashboard or an API. As an IaaS client, you have to take care of runtime, operating system and middleware in addition to your application. Depending on your needs, this can of course be advantageous, but it can also be tedious additional work.

PaaS is another alternative. A kind of framework is provided on which developers can build and thus easily develop their own applications. The management of servers, operating systems, software updates, storage and the infrastructure is managed by the provider. This allows developers to focus completely on the development of their application. PaaS is based on virtualisation technology, which makes it possible to easily scale up and down resources as needed. (Watts and Raza, 2019)

Heroku

We have chosen Heroku as our cloud service. Heroku is a Platform as a Service provider based on a managed container system. It additionally offers integrated data services with a strong ecosystem. Here it is possible to extend your application directly with 3rd party cloud services via add-ons. These bring a range of functionality with them, such as data stores, logging, monitoring and more. Heroku runs applications in so-called dynos. These are smart containers that run on a reliable and fully managed runtime environment.

Heroku also advertises their Heroku Developer Experience. They offer an app-centric approach so that the developer only has to focus on the development and continuous delivery of the app and is not distracted by servers and infrastructure. For deployment, they offer an excellent and simple solution, directly through development tools such as Git, GitHub or Continuous Integration Systems. Another big advantage of Heroku is its operational experience. They help developers with troubleshooting common issues and identifying negative trends of your application health. They also offer tools such as notifications if something goes wrong and automatic scaling of your application.

A frequently heard counter-argument to cloud services is that you pass your data on to third parties and that it may not be secure. Heroku is aware that many developers entrust them with sensitive data. Therefore, they regularly carry out numerous audits to protect their clients' data. (Heroku.com, 2020)

We already decided to work containerised with Docker during the development of our application. This allowed us to develop our frontend and backend independently in their own environments. Since our app is already containerised, our decision to use Heroku was even clearer. Heroku offers a container registry for deploying with Docker. Here you can deploy a pre-built Docker images to Heroku directly via the CLI. This is also possible with the integrated CICD on GitHub. We only need to push our project to GitHub and it will be deployed directly to Heroku. This process is effortless and thus saves a lot of time.

We have now created two apps on Heroku one named 'imagechase' for our frontend and one called 'api-imagechase' which contains our backend project. By using docker, we were able to define all the necessary steps directly in our Dockerfile so that our applications are built directly and are designed for a production environment before they are deployed. This way, our image only contains the files it needs for production and does not have all the excess development data with it, which would unnecessarily inflate the app. Using Heroku's add-on function, we were able to add a PostgreSQL database to our backend with one click. We didn't have to set up a database and integrate it into our app on our own. All we had to do was access the variables Heroku put in the environment and everything worked seamlessly. Another plus point is that Heroku directly brings TLS with it. You don't have to worry about certificates or other configurations. You can access your application directly via https in the browser without any problems.

Another deciding factor for us was Heroku's pricing system. Here it is possible to choose the 'Free and Hobby' tier. This is completely free for non-commercial apps like our proof-of-concept page or personal projects. This means we can test the service and our application without any costs. As soon as we are ready for the release of our application, we can choose the ‘Production’ tier. Here we pay 25 dollars per month for a standard dyno, which is enough for us to start with. We have two applications, the frontend and the backend, so we need two. For the Heroku Postgres database, we also have to budget another 50 dollars. In total we would be at a cost of 100 dollars per month when we release our app. If the traffic increases over time, it is very easy to scale up the application by changing the tier. (Heroku.com, 2021b)
What we really like about Heroku is that the prices and services are clearly defined. With AWS, for example, you enter your credit card but don't know exactly how much the final fee will be.

6.2 SCALABILITY

Scalability is a factor that should be considered right from the start of development. But especially when choosing a hosting service, scalability is definitely a priority factor. Especially when your app grows and the number of users and traffic increases, you are of course happy. But if you then have to deal with scaling problems and downtimes, this can quickly spoil the joy. Especially as a company, high costs can arise quickly when the own services are down. Not only that, but your own users could soon be annoyed and, in the worst case, switch to the competition. Such problems are of course to be avoided at all costs.

Heroku

With Heroku, it is easy to scale your dynos according to your needs. For this purpose, they offer easy-to-use tools such as the Heroku Dashboard or directly via the Heroku CLI.
There are two scaling options. On the one hand, it is possible to scale horizontally. This means adding more dynos that run in parallel. Especially when the traffic volume increases, the performance can be increased by splitting the HTTP requests to different instances. However, this scaling option is not always sufficient under all circumstances. The other option is vertical scaling. This means upgrading to larger dynos. In this case, you buy a better pricing tier and thus get more memory and CPU resources. Heroku also offers the possibility of autoscaling for certain tiers. The application is automatically scaled for traffic spikes and you do not have to manage this yourself. (Heroku.com, 2021a)

Cloudinary

As a provider of an image competition website, you must of course be aware that you have to store several images and that large amounts of data can quickly accumulate over time. We have decided to store our images via the provider Cloudinary. At the beginning we use the Free Tier. Here we have 25 GB of memory at our disposal. As soon as this becomes too small, we can upgrade to the Plus Tier, where we receive 225 GB of storage for 89 dollars per month. (Cloudinary, 2021)

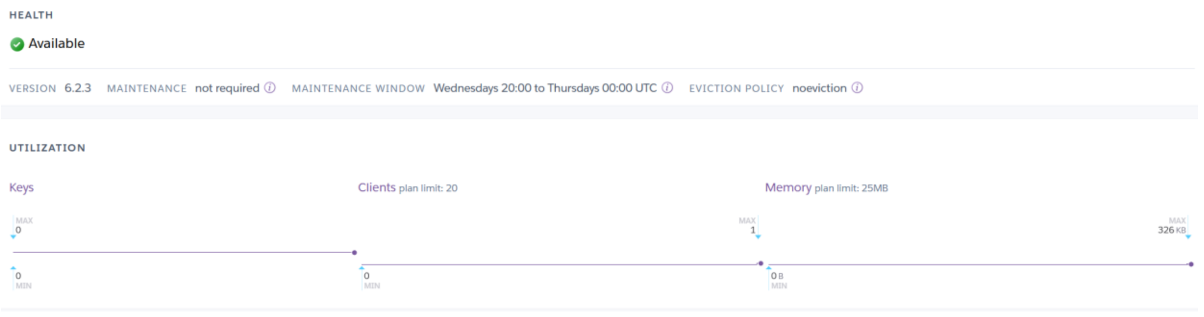
Through the services we have selected, we are able to react very quickly to increasing user numbers, traffic and storage. With Heroku as well as Cloudinary it is easy to increase the resources against payment.

6.3 Tracking and statistics

To get an overview of your running application it is important to monitor it with appropriate tracking tools. This allows you to react in time to certain events and then scale up if necessary. Various statistics can be observed that give you an insight into your own app. For example, the memory used by the application, or the current request throughput can be viewed. Or even how high the request time is. These metrics can be important factors to evaluate the health of the app and to intervene if necessary.

Heroku

With Heroku, there are numerous ways to monitor your application in different areas via the add-on function. For example, various add-ons are offered for application performance monitoring (APM). For example, the New Relic APM add-on can be used here. You can configure this according to your preferences and, for example, identify parts that slow down the app.
There are also various tools for monitoring the platform. One is Heroku Redis. This can be used, for example, to detect when a dyno is running out of memory or when there is a rapid increase in database load.


HEALTH
Available

VERSION 6.2.3 MAINTENANCE not required MAINTENANCE WINDOW Wednesdays 20:00 to Thursdays 00:00 UTC EVICTION POLICY noeviction

UTILIZATION

Keys: MAX 0, MIN 0
Clients plan limit: 20: MAX 1, MIN 0
Memory plan limit: 25MB: MAX 326 KB, MIN 0 B

Figure 15 *Heroku Redis Dashboard*

A somewhat clearer and more comprehensive tool is the add-on Librato. You can additionally expand it with further metrics according to your own wishes. (devcenter.heroku.com, 2021)
Most add-ons can also be tested free of charge at first. However, you still must deposit a credit card for this. If the metrics are not sufficient, you can of course upgrade the tools for a fee.

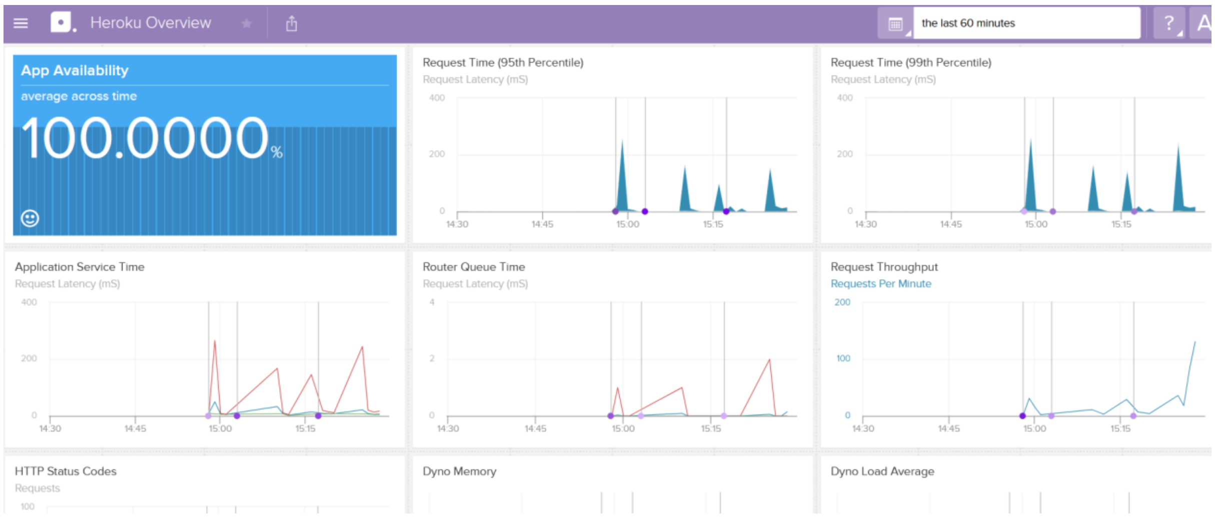


Figure 16 *Librato Dashboard for platform monitoring*

Auth0

By using the authentication provider Auth0, we can see useful information about the activity of the users in a dashboard. This way, we can see in a diagram how many users sign up on which days. This is very useful because it allows you to observe the growth of the application. This way you can see on which days there were many new sign ups. This can then be used to understand why there were many new sign ups on that day and applied to the marketing strategy accordingly. The same also applies if there were only a few sign ups, so that we can try to prevent this in the future.

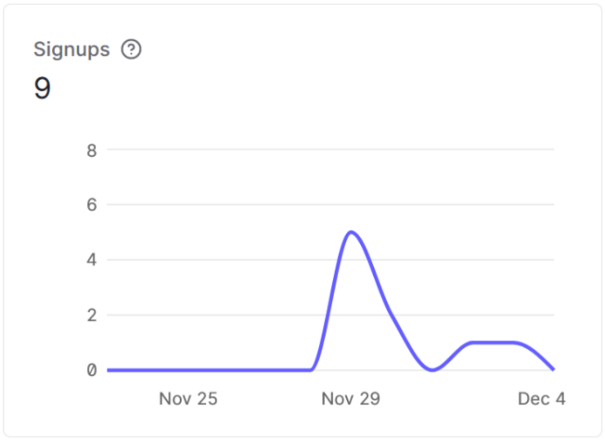


Figure 17 Signups diagram

Not only new sign ups are important, but also that the new users enjoy the application and continue to use it. For this we have the active users dashboard, where we can easily track how many users are using the application every day.

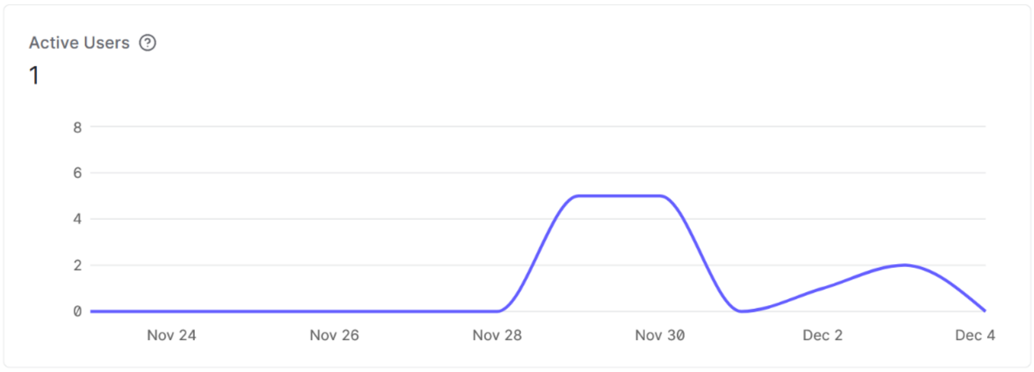


Figure 18 active users diagram

This metric is critical for determining whether we can keep our users.

7 CONCLUSIONS

All reports must draw conclusions. Yours should say why your approach to the brief is the most appropriate.

8 REFERENCES

All references must use Harvard format and be cited in the text following the UWS referencing guide.