Afbeelding met tekst

Automatisch gegenereerde beschrijving

Web 2.0 Application

eventguard

Maxim Brabants | Kwinten De Rop

| Web Technologies | December 23 2022

Table of contents

[Introduction 2](#_Toc122722892)

[Frameworks & Technologies 3](#_Toc122722893)

[Database Management System 3](#_Toc122722894)

[Used packages 3](#_Toc122722895)

[Functional Requirements 3](#_Toc122722896)

[Responsiveness and Mobility 3](#_Toc122722897)

[Home page/ landing page 4](#_Toc122722898)

[HTML5 FEATURES 4](#_Toc122722899)

[Registration process 4](#_Toc122722900)

[Frontend form validation 6](#_Toc122722901)

[Backend form validation 8](#_Toc122722902)

[Organiser 10](#_Toc122722903)

[Map-based component 10](#_Toc122722904)

[Nominatim Geocoding API (Web Service) 10](#_Toc122722905)

[Search Bar 11](#_Toc122722906)

[Profile Page 12](#_Toc122722907)

[Review system 12](#_Toc122722908)

[Product wish list 13](#_Toc122722909)

[Provider 15](#_Toc122722910)

[Scrollable inventory with search bar 15](#_Toc122722911)

[Add articles 16](#_Toc122722912)

[Article details 16](#_Toc122722913)

[Resources 17](#_Toc122722914)

# Introduction

(Kwinten & Maxim)

This project report describes the full development of a Web 2.0 application around a topic of free choice. The ultimate goal of the website is to allow users to view, search, create, manage and share information.

The idea behind this particular project is an event organizing application where there exist two roles in the system. Either you have the users who are named as ‘organizers and this type of user is the one who is willing to plan an event or wants to organize an event in the future, but of course these users are in need of different kinds of materials or equipment in order to set up the particular big or small event.

This is where the companies or providers come into use because these are the type of users that offer or lend this kind of equipment to various organizers. The type of equipment can vary from sound installations, light equipment and even stage building constructions. Actually, every sort of provider instance that can offer some kind of physical entity can be involved in the system. Briefly described: organizers can look or search up equipment or even companies on the website and providers can add new content to their page in the form of new products along with their information. Other functionalities are presented in the remaining part of this document.

Our main focus is to provide a platform for youth movements, as there is currently no easy way to find the right equipment for a reasonable price. That being said, the website is open to every kind of organization, as they might need/offer the same kind of equipment. Our goal is to make sure an organization can organize an event anywhere without having the long search for the right equipment.

# Frameworks & Technologies

(Kwinten)

For the frameworks we used Vue.js for the front-end and Node.js for the back-end. Vue.js is pretty easy to learn out of all the front-end frameworks for web development. We choose Node.js because we used this in the exercise sessions of the course ‘Web Technologies’.

# Database Management System

(Kwinten)

For the database, we used MongoDB. As one of the world’s most popular NoSQL databases, we mainly chose it because of the easy to learn commands. In hindsight, there might be other options that suited this project more, as MongoDB does not support image storage.

# Used packages

(Kwinten)

For our project, we used the following packages:

* Bcrypt: A library to help you hash passwords.
* Mongoose: This package help sus to get a connection with our database. It is a [MongoDB](https://www.mongodb.org/) object modeling tool designed to work in an asynchronous environment.
* Axios: Axios helps us to make HTTP requests from Node.js.
* Vuetify: This is a UI Framework with handcrafted Vue components.
* JWT: JSON Web Token is a library that gives us the ability to securely transfer information over the web.

# Functional Requirements

(Maxim (All))

This section of the document enlightens all the present functionalities that are included in the system. First, we will cover the registration and login process. How can users register for an account in the system? How can they log themselves in? What kinds of information do they need to provide in order to create an account? All these kinds of questions will be answered in this section.

## Responsiveness and Mobility

This website is also responsive on a certain level which means that some of the pages can get resized to a certain minimum without affecting the content. (Profile Page, Home Page, …)

## Home page/ landing page

Every website has some sort of page where non-logged in users have access to and from there, they can choose to register themselves or login with an existing account. The landing page of our site just consists of some images and text next to those images with the goal of informing the user what this site has to offer.

### HTML5 FEATURES

The full structure of the home page is as follows:

* **The header:** For the header, we used the official **HTML5 header tag.** We use this header to show our navigation bar which is present on every page of the website. At the start, navigation to login and register page are possible with the navigation bar.
* **The main content:** Underneath the header comes the main content which was just mentioned (images and text).
* **The footer:** For the footer, we used the official **HTML5 footer tag.** We use this footer to show some additional information about EventGuard and also a top navigation button to navigate back to the top of the website.

The structure that is described above is actually the main structure used in every other page of the site. We use the header and footer as static structures to remain the same except for the header navigation bar content. In between the header and the footer, we insert our current Vue component.

Furthermore, we included the official **HTML5 form tag**, both in the registration form as well as in the login form.

## Registration process

This is almost every time the main entry point of nearly every website, out there on the internet. Users need to provide some personal information, they create their accounts with it, then they login in with this data and only from that moment on, the website can be used to its full potential.

Afbeelding met tekst

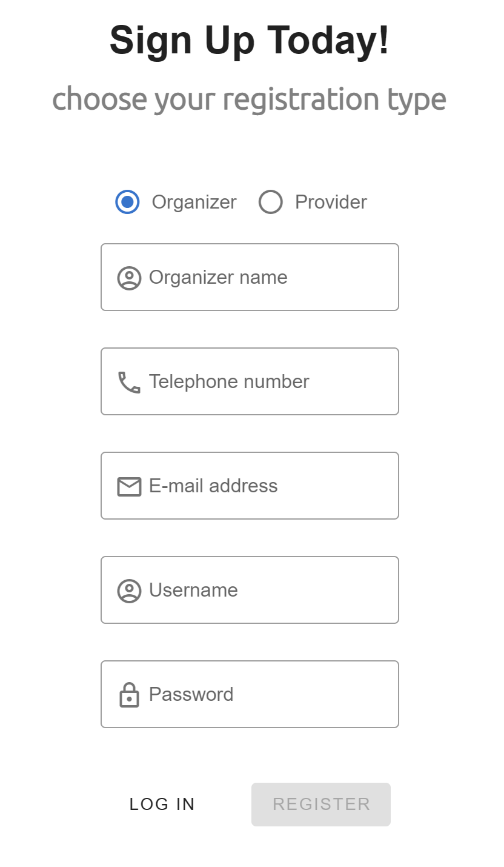
Automatisch gegenereerde beschrijving

Figure 1: organizer register form

Figure 2: provider register form

The above two snippets are taken from the official registration page of the website. On the left side, the registration form for the organizers is visible and on the right side, the exact same version but only for the providers/companies. Actually these two pages look exactly the same, except for the name fields where the label differs a bit. We chose to provide two separate forms, since we have a MongoDB collection for each of the user types. So we have a collection for all the organizers and one for the providers. This way it’s easier to store their information in the correct collection afterwards.

## Frontend form validation

Of course, all the provided user data in the form fields have to be checked upon potential dangers or false information. For example, some malicious people could abuse certain form fields in order to perform known SQL-injections in case of no validation or users could maybe input an email address that doesn’t correspond to an existing one. For this purpose we included some input rules for each field to prevent these situations from happening.

Afbeelding met tekst

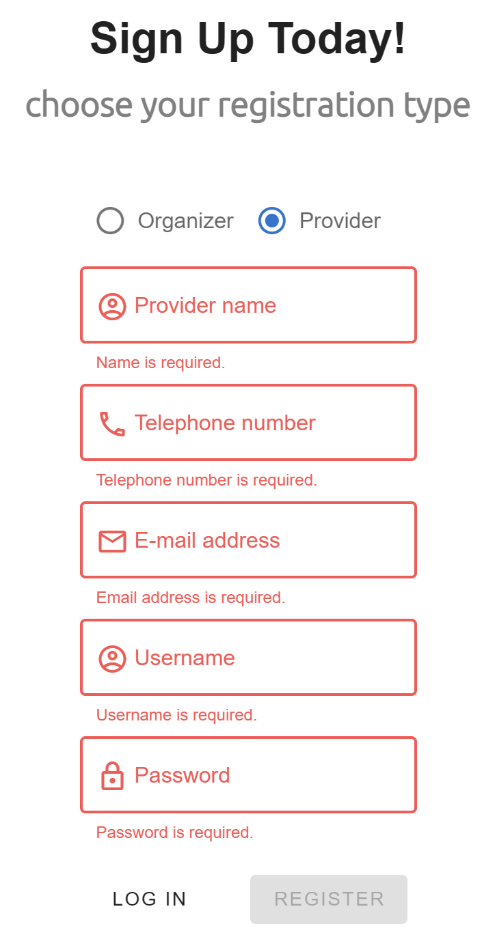
Automatisch gegenereerde beschrijving

Figure 4: disabled register button

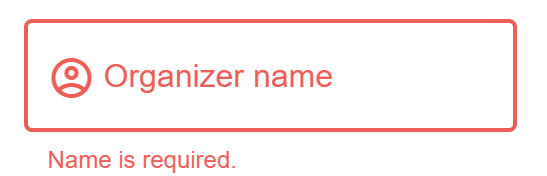
Figure 3: required validation

To begin with, all the form fields are required to be filled in, so no fields may be empty when submitting the form. In case there would be an empty field, it gets clearly highlighted by the system. On top of that, the system automatically disables the submit button (REGISTER button) when one or more of the fields are not filled in according to the rules.

Not only this condition is checked when the user inputs its data. Each field has an extra set of distinct validation rules defined as follows:

**Name – input field**

For the name, we just **require it to be filled in**. No extra conditions are added here because it’s simply a name which doesn’t have to be checked against some format.



**Telephone number – input field**

When it comes to the mobile phone number, we expect the user to provide just a **valid phone number**, meeting the following regular expression:

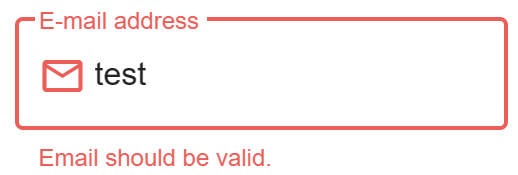
/[\+]?[(]?[0-9]{3}[)]?[-\s\.]?[0-9]{3}[-\s\.]?[0-9]{4,6}/

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Email address – input field**

The user also has to submit an email address that is **valid** with respect to the following format: “**x@x.x**”.



**Username – input field**

The username is a value that is required to be **unique** to every user. Although this is not checked real-time, furthermore the username has to be **longer than 6 characters**.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Password – input field**

A password is needed as well to log in along with the username. The validation rule states that the password has to contain at minimum **a lowercase letter, a number and an uppercase letter**.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

We use the following regular expression for this purpose:

/(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{6,}/

If all the conditions are met and the user has provided the right input in every field, the user gets redirected to the login page. Also, a visual confirmation message is shown at the top so the user knows everything went fine.

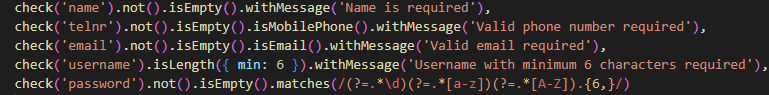
Afbeelding met tekst

Automatisch gegenereerde beschrijving

## Backend form validation

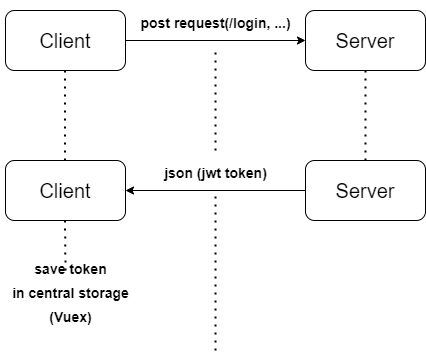
All the data, submitted by the user or any person that fills in the form, is eventually sent to the server and there it gets processed, but assume we wouldn’t have any form where the user could enter their credentials. Then the data would be entered through some web interface or request engine. This way, any data could be sent to the server without getting validated.

As a consequence of this, the server is also equipped with backend validation. Our server performs this with the help of Express middleware functions where we put one intermediate function for each value



When the server-side validation is performed, the data can be processed and put in the database. After all this is done, the user gets redirected to the login page where their email address and password can be entered. As mentioned earlier, form validation is performed on the login side as well. As soon as their credentials are sent to the server, these ones get checked upon their existence in database. So, the username, which is a unique field in the database, gets checked. When it exists, the password gets checked with the help of the **bcrypt ‘compare’ function** because if we just compare the normal password with the one that is encrypted in database, it won’t work.

When the user is finally authenticated, the server creates a **JSON Web Token** which gets sent back to the user. The logged in user can use this token to later authorize themselves with it if they want to access resources specific to that user account. The full process is actually as follows:



Of course, this JWT token has to be stored somewhere on the client side. For this we used **Vuex.** It is a state management pattern + library for Vue.js applications. It serves as a centralized store for all the components in our application, with rules ensuring that the state can only be mutated in a predictable fashion.

As soon as we want to change something about this user state, we can just call methods in this state management to either get some user data, or set data about the user.

# Organiser

(Maxim (All))

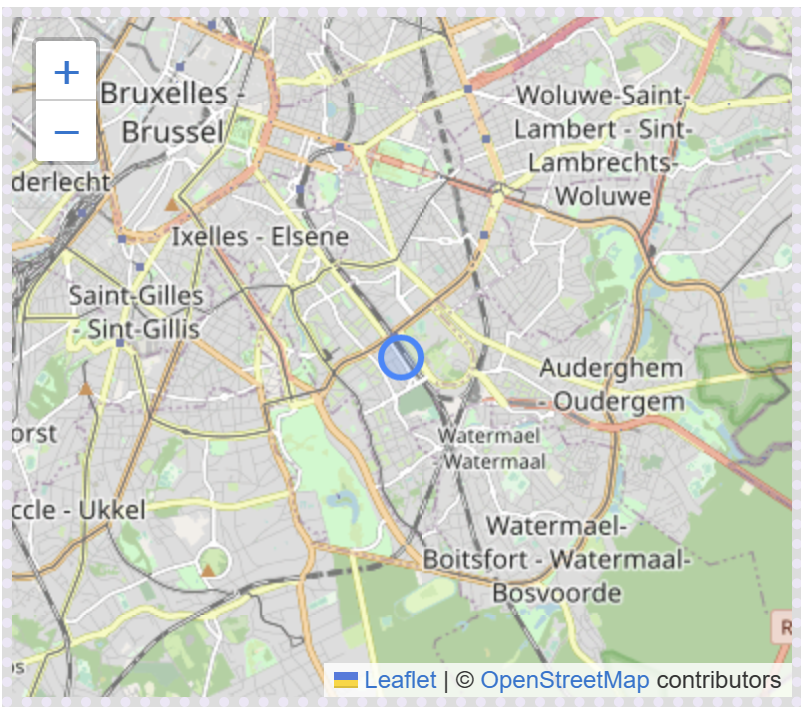
Let’s begin with discussing the functionalities, an organiser is able to do. So first, it has to be possible for organisers to search for companies/providers to rent equipment from. For this one, the organisers have to options: either they can look for companies/providers on a map-based component or they can simply search for them based on the name of the company.

Let’s discuss these two options:

## Map-based component

For the map aspect we were planning on using the normal Leaflet library but after further research, we found out that there also exists an alternative specifically for the VueJS frontend framework namely ‘**Vue2 Leaflet’**. This library provides Vue components for Leaflet mapping components to allow simple construction of declarative maps.

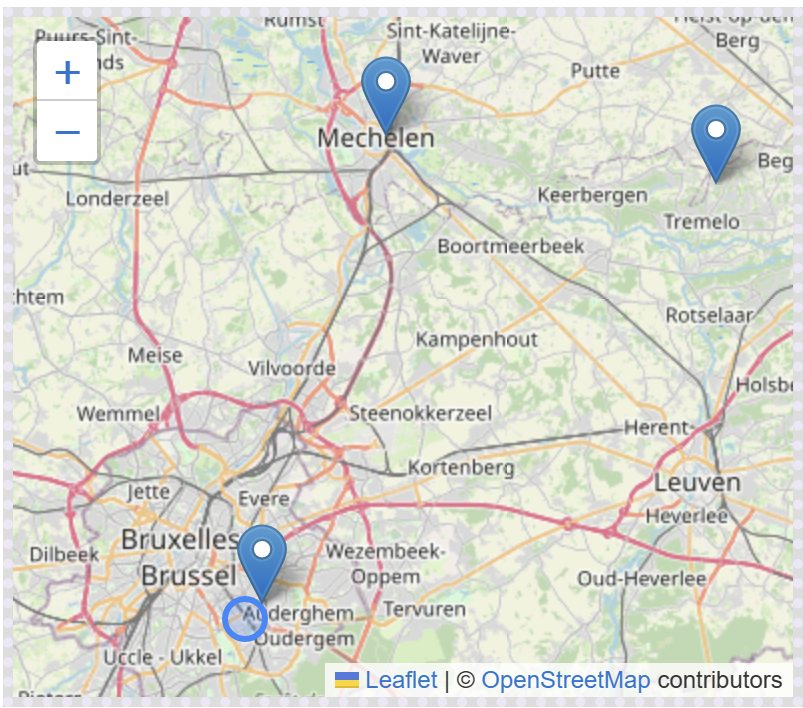
Initially when the organiser is logged in, they see the map immediately, additionally with their current location at the centre of the map. From that point on, the user can start looking for different providers on the map.



## Nominatim Geocoding API (Web Service)

The way that providers/companies get displayed on the map is by means of markers with a latitude and longitude value. As soon as users hover over the marker, they can see the name and address of that particular company displayed in a small tooltip.

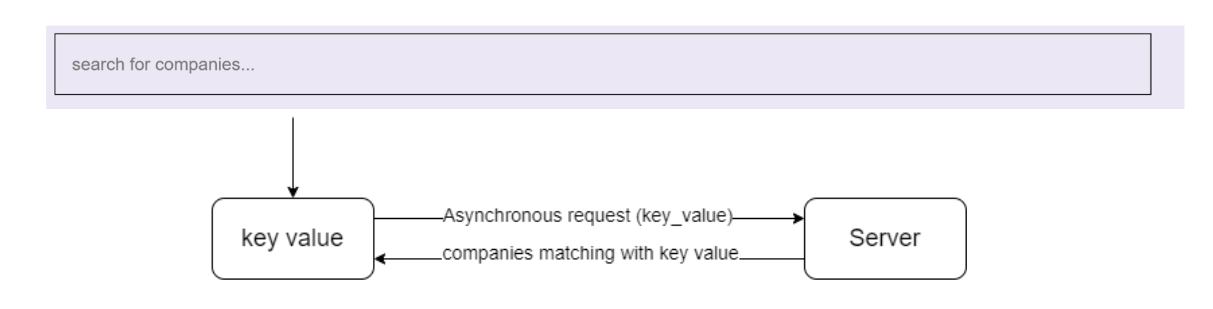
When companies/providers register for the platform, they have to provide a real address, which is not the case with organisers; they don’t have to provide this information. When the map loads, the addresses of all the providers are retrieved from the database and by means of the geocoding API (Nominatim), they each get converted to a pair consisting of a latitude value and a longitude value. These two values are then used to indicate where a specific company is located, which corresponds to the real address on the map.



The organiser has then the ability to click on one of these markers as well and this will redirect them to the official page of that provider which will be discussed in the provider section.

## Search Bar

Next, the user also has the option to search for certain providers or companies based on their name. This is very useful in case the user doesn’t exactly know where the company is located, but the name is known. One concrete feature about this functionality is that it doesn’t have to be the exact name that has to be filled in. As soon as the user types in one letter, an asynchronous request gets sent to the server and the server is going to look for all the companies or providers where the company name starts with that letter. If the user then even types more characters, the search is going to be more detailed. The more characters are filled in, the more precise, the search results will be. Underneath is a visual overview of how it works.



The companies along with a picture and their name get displayed in a card-like component. Only the first then results get sent back by the server.

## Profile Page

Every user in the system has the ability to consult their user data that they provided in the registration form. This page is divided in three columns.

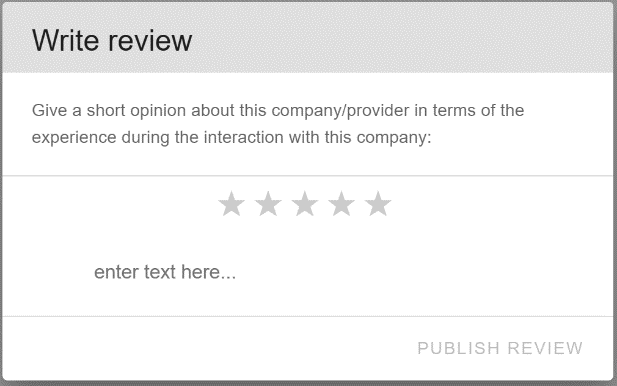
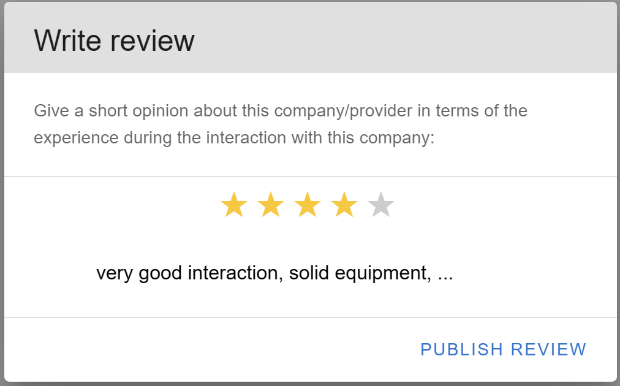
* The left section contains the profile avatar of the user along with the email address of the user and the name. It is possible to toggle between different pre-set avatar pictures. There are four pictures in total.
* In the middle section, the user data is presented, which can be modified by the user itself.
* The right most section is reserved for a wish list. This functionality is described in the wish list section.

One important thing to mention here is that the user data is not retrieved from the database yet when the user logs in to their account. This is only done when the user navigates to this page. The first time, this data is queried from the database but after that, this data gets stored in central storage (Vuex).

## Review system

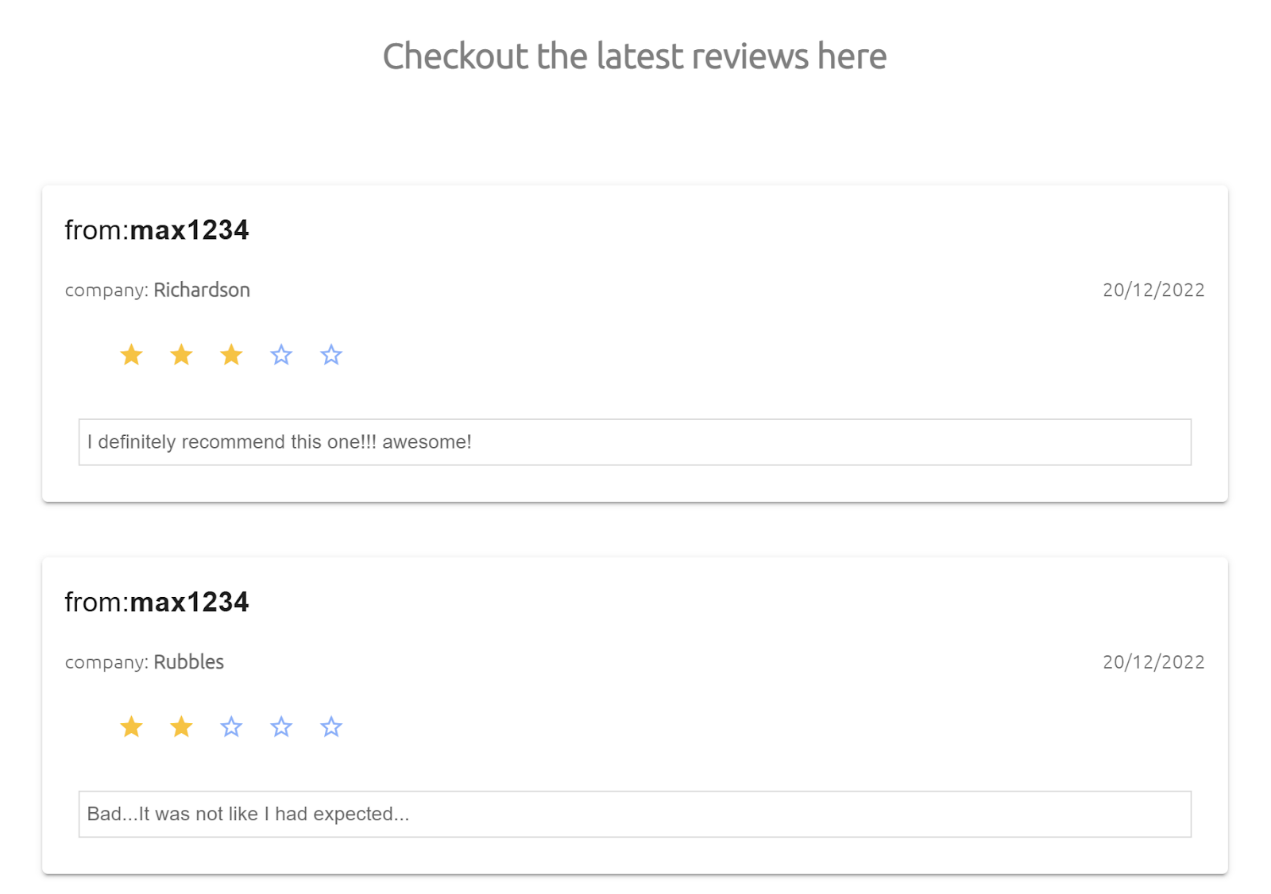
If organisers are interested in the services of some particular company, they can contact that company by means of a telephone number or an email address. The further interaction between organiser and provider is not handled over this site, but could be a nice extra functionality. As soon as the contact between organiser and provider is over and the interaction is fully taken care of, it is possible for the organiser to leave a small review about the articles offered by the company and/or how the interaction between the two went.

The way, the user can place a review is by going to the official page of a provider and by clicking on the ‘place review’ button. When clicked on that button, the following dialog pops up.

It is then possible to write the personalized review by choosing an amount of stars followed by a short message (max 50 characters). It is not possible to place a review when one of the values are empty.

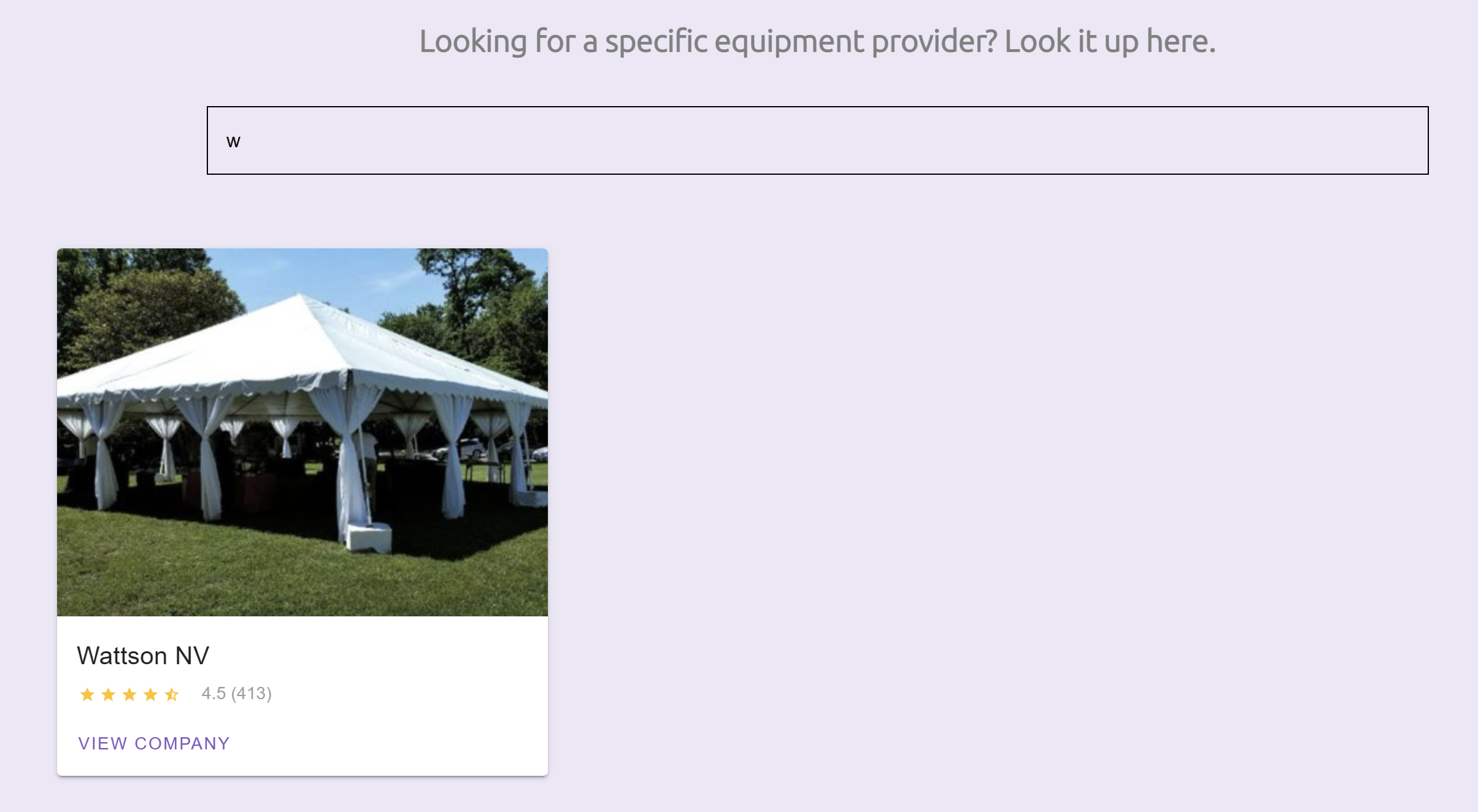
When a certain organiser has written a review about a particular company, this review becomes visible for every other organiser and company/provider. The public reviews of every user can be found underneath the search bar and they look like follows.



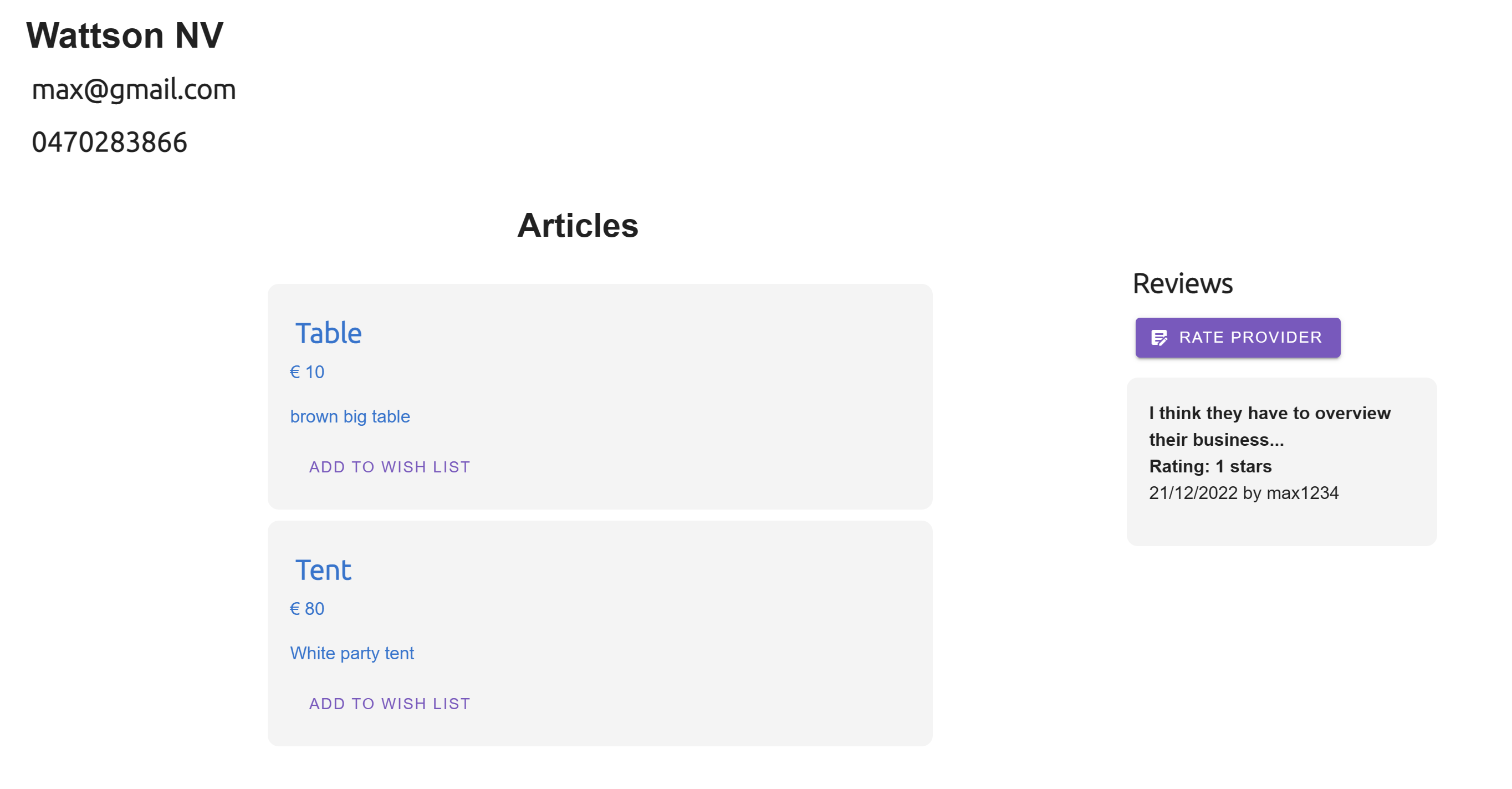
## Product wish list

Finally, one last functionality that got quickly implemented is the idea of a wish list for the organisers which already got mentioned in the profile section. The general idea is that organisers have to plan events and they need certain equipment but the list of equipment is probably very long, so it would be useful for them if they could keep a list of noted articles they’ve already chosen from a company. This is where this product wish list comes into play.

This is done by navigating to the company page and clicking on the ‘Add to wish list’ button on an article list item whereafter this article gets added to the wish list on profile page. Obviously it would also be useful if they could modify and delete these products but since the time limited us, it could not be implemented anymore.

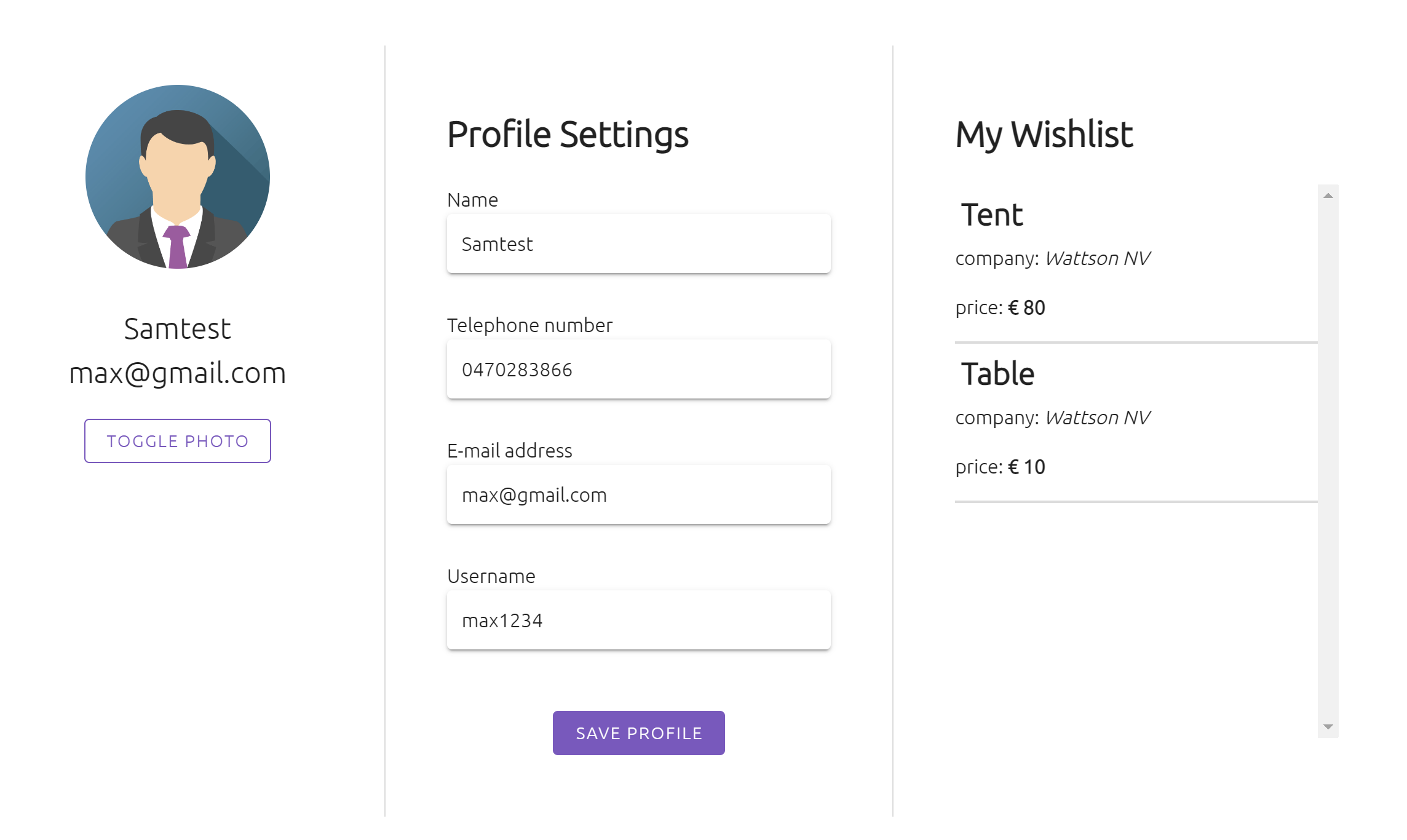


In this example, the user looks for the ‘Wattson NV’ company. After this the ‘View Company’ button can be clicked whereafter you land on the following page:



On this page, the user gets to see all the details about that particular company including their articles and their reviews from other users. Next the user can click the ‘Add to wish list’ button to add that particular product to their wish list.

When the user navigates back to their profile page, they get to see their added products along with their details.



# Provider

(Kwinten (All))

### Scrollable inventory with search bar

It is necessary for the providers to be able to have a clear overview of their products. They can see the name, price and short description of the product. To navigate easily to a certain article, the provider can use the search bar or the filters (based on price).

Afbeelding met tekst

Automatisch gegenereerde beschrijving

### Add articles

The provider can add a new article at any given time, by filling in an easy form. They need to define the name, description, price and details of the product. The description is used for a small explanation what the products does, and the details are for listing the specifications of the product (weight, size, energy label, …). The provider can also choose an image of the product, to make it more visual to the organizers. Due to the lack of support of images in MongoDB, this is unfortunately hardcoded.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

### Article details

When clicked on an article, the screen will show a detailed overview of the article details. Next to the image, you can see the details of the product who got defined in the form. On the bottom, you can find contact information about the provider instead of a order button. That is for the reason that our targeted users our youth movements, who consist of volunteers. It is not their primary occupation to rent out products, like some big companies. The organizer will need to contact the provider to see if the product is available for the wanted period, and when they can pick up and give back the equipment.



# Resources

* https://www.youtube.com/watch?v=YrxBCBibVo0&list=PL4cUxeGkcC9hYYGbV60Vq3IXYNfDk8At1 (all ninja’s VueJS tutorials)
* https://github.com/vue-leaflet/Vue2Leaflet/issues/476
* <https://codepen.io/hesguru/pen/BaybqXv>
* https://vuetifyjs.com/en/components/text-fields/