Documentation

> ## Web Design for the Industry Autouria
>
> SWD600 Project Report (Live Brief) & Artefact
>
> **Check out the site here:**
>
> **https://wdi-autoura-assignment.firebaseapp.com**

Background

This Autouria website allows users to see different “stops” such as food and drink, and events. Users can also sort it by group context. Alongside this there is a toggle button that allows you to toggle accessible places.

Homepage

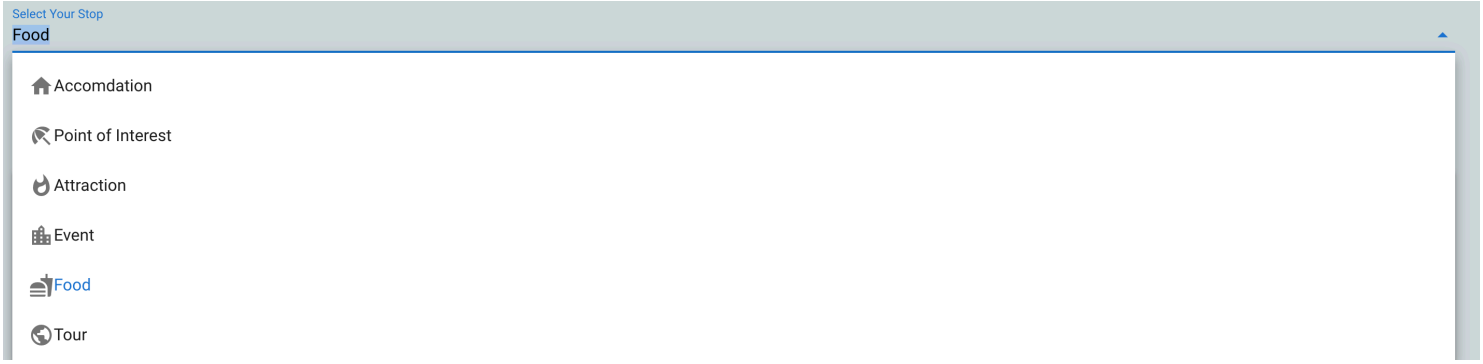
In this section I will discuss the contents of the homepage.

Dropdown

As shown in the image above I created two dropdowns using Vuetify’s combobox which the documentation for this can be found here

The main reason I chose to do the dropdown using the combobox as opposed to using the normal option HTML attribute was because I wanted to add icons next to the label, and this was not possible using that technique.

This was the end outcome



Cards

To display the results, I decided to use the cards layout, by using Vuetify’s card component which the documentation for can be found here.

The first thing that was done was create a basic card layout, which I got from the documentation, and then replaced the image with result.picture.url from the Autoura API (this will show the images of the poi).

```
< v-img
   height="200px"
   :src="result.picture.url"
   v-if="result.picture.url !== null
>
```

However, as you can see in the code above, if there is no image then it will = null and nothing will show. Within the `<v-container>` there is a `<span>` which is

where the following is: `v-text="result.name"`

This displays the name of the poi from Autoura API.

As mentioned earlier, accessible places had a toggle button. The cards display an accessible symbol IF the location is accessible. To do this I used one of Vuetify's card component v-card-actions. This allows a basic condition which is usually used to add and hide text until it's opened. However, I used it to show or hide the accessibility symbol by doing the following: `<v-icon v-if="result.accessibility.wheelchair == true">accessible</v-icon>`

As you can see if the result accessibility is true then display the accessible symbol. In supplement this there is also a button in a form of an arrow next to it which takes you to the next time.

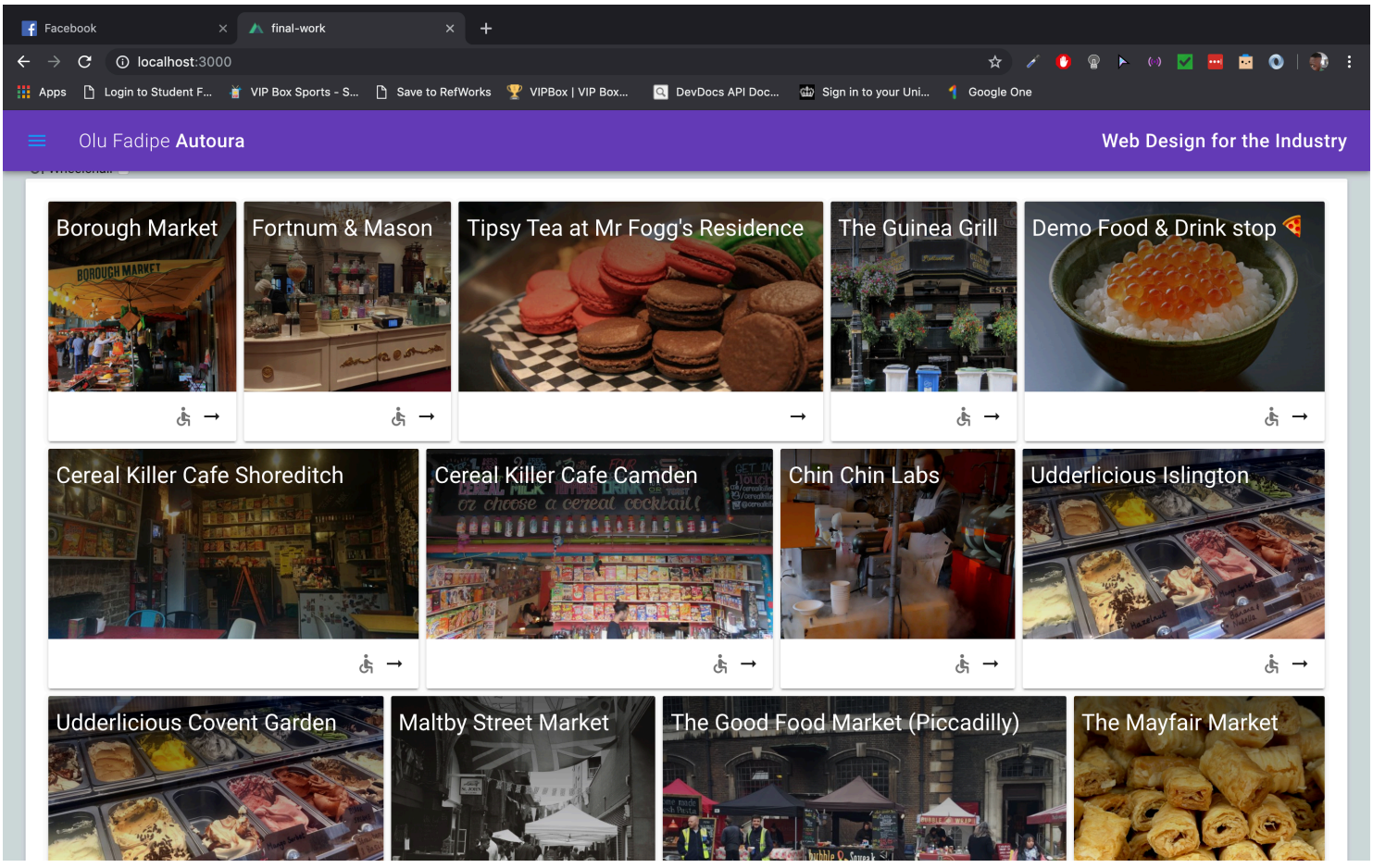
```
<v-btn :to="'/results/'+searchPick.search +'/' + result.stop_id" icon> <v-icon>arrow_right_alt</v-icon> </v-btn>
```

The button is complimented by the following code in the script section

```
let link =
     "https://api.autoura.com/api/stops/search?group_context=" +
     this.groupPick.search +
     "&stop_types=" +
     this.searchPick.search;

   if (this.wheelChair) {
     link = link + "&wheelchair=true";
   }
```

This is what the end result for the cards looked like



POI Page (More Information Page)

When the user clicks on the arrow button on the card it takes you to another page which provides further information such as their website, about the poi, map and weather.

Header

In this section I discussed features I added in the header such as the weather, url and header image.

Header Image

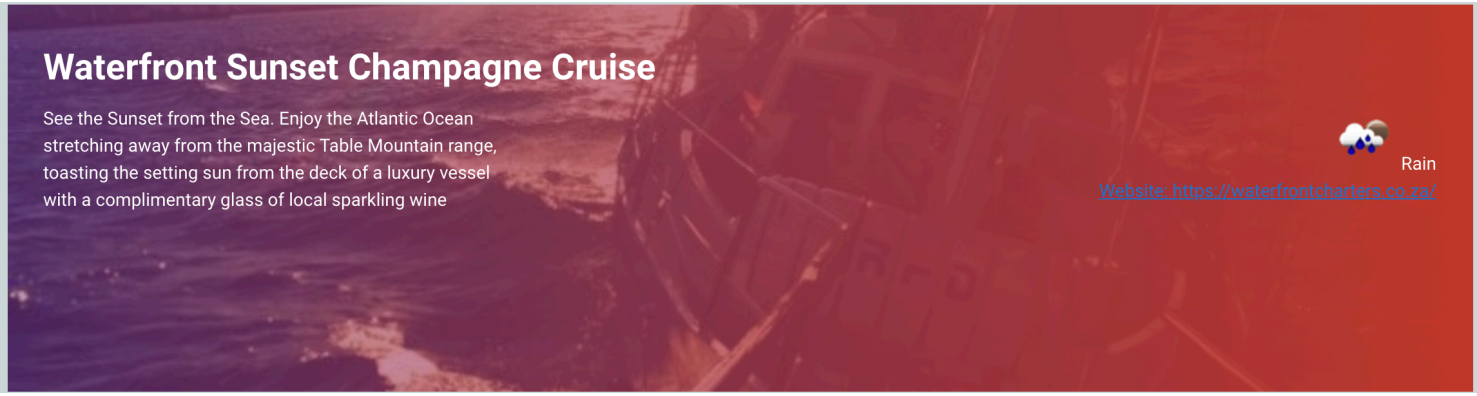
For the header image, I used the images provided in the Autoura API, but to allow the text to be visible I added a gradient by using an Vuetify component called gradient which you can find the documentation for on this page here.

I used the website uiGradients to create a gradient and then converted it into rgb.

```
style="color: white" gradient="to right, rgb(142, 68, 173,0.5), rgb(192, 57, 43,1.0)" height="300" :src="object.picture.url"
```

Other Header Information

I added the name of the poi and the summary of the poi. These are shown on the left side of the header. On the right side however is where the weather is shown which I will go into more detail about how that was done later and the URL of the website which I got from the Autouria API. This was the end result



Main Body

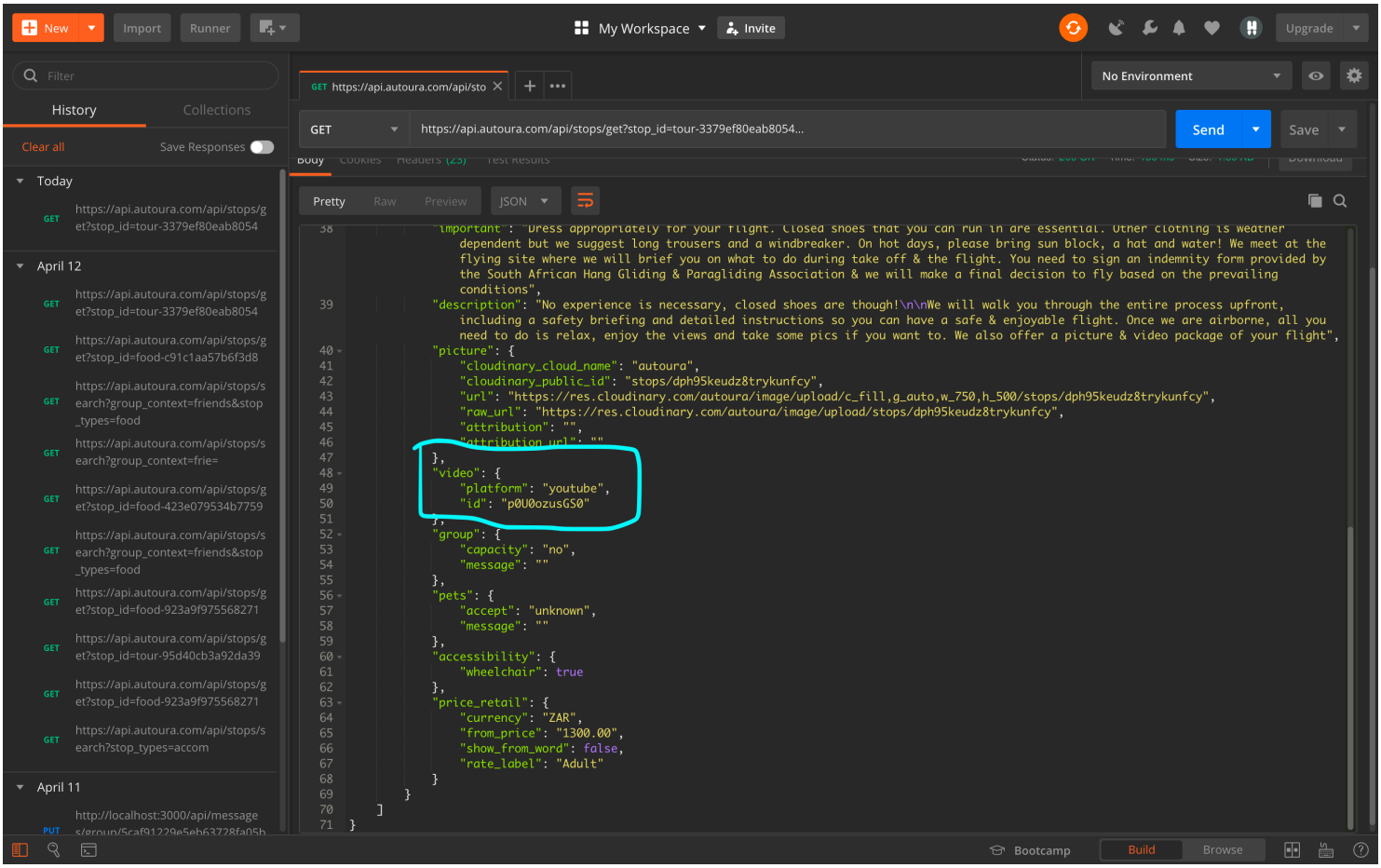
On the left side of the page further information about the poi (Autouria provided a summary and description information for each poi).

On the right side however is where things become more interesting. The right side of this section provides a video from YouTube or Vimeo or if the poi does not provide either it will show an image. When looking through the data provided from the Autouria API I noticed that some of them had videos attached to them. When I investigated further, I noticed that the videos were either from YouTube or Vimeo, so I had to figure out a way to show both. But the first challenge was how I was going to show one. YouTube was embedded like this:

```
<iframe
            v-if="object.video.platform == 'youtube'"
            width="560"
            height="315"
            :src="'https://www.youtube.com/embed/' + object.video.id"
            frameborder="0"
            allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture"
            allowfullscreen>
</iframe>
```

The first thing you will notice is the V-if which is essentially an IF statement in which you can find the documentation for here.

As you can see in this random GET request, I did on Postman in video (the section circled), there under platform it says “youtube”. So, going back to the v-if statement video platform it’s saying if the video platform is youtube then embedder will show. As you can see in the source, it is it adds the ID after the https://www.youtube.com/embed/. A similar thing is how Vimeo video are embedded. The main thing to note with the Vimeo embedded video is that it uses a v-else-if statement which essentially is an else if statement.



As mentioned earlier if there are no videos, then an image will appear, which works the same as how the header works, just without the gradient on top of the images, and uses a v-else statement.

Methods

In this section I will be discussing what methods I used for this project, most notably Nuxt.JS

Nuxt JS

Nuxt JS is a Vue.js framework which essentially is Vue js on steroids. To read more about Vue check out their official document here.

The reasons I chose to create this project using Nuxt are the following:

- Automatically add routes to the route.js file which saves a lot of time.
- During the setup it allows you to easily choose the server, UI framework (such as Bootstrap) and testing framework (such as Jest) and gives you an option to add an axios module.
- Gives you an option to make it pwa ready

Postman

Postman was used to run GET requests. This was one of the first things that was done to find out what information was available in the API data, this allowed me to plan out the rest of the website and what I would had done.

Implementation

Github

GitHub was used throughout this project to make commits and track changes.

Deployment with Firebase

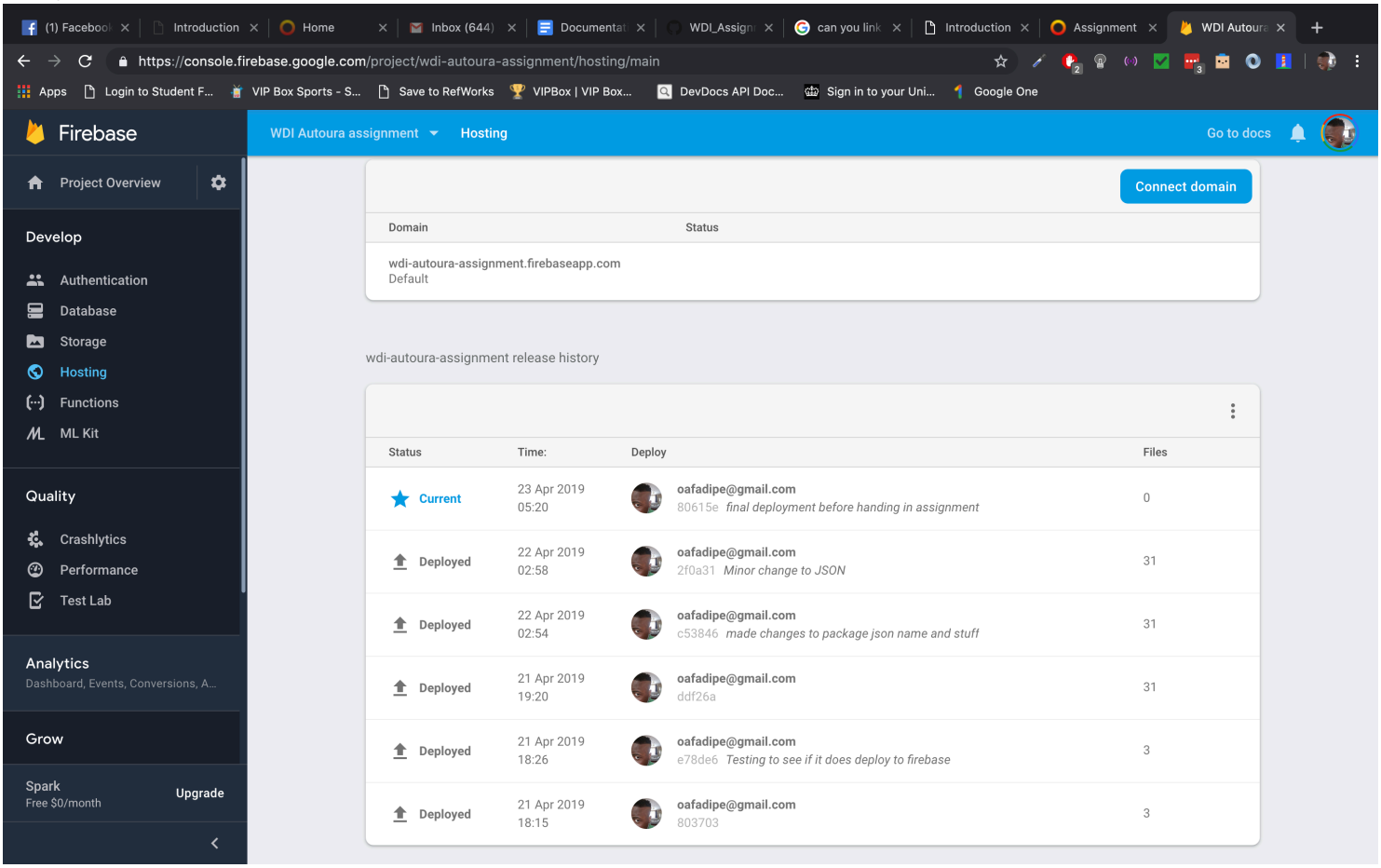
This project is deployed onto Firebase. The first thing I needed to do was create a Firebase project. I mostly followed the deployment page on the official Vue CLI

documentation which find be found here.

However what in the firebase.json had to be modified since I used Nuxt for this project. To solve the problem, I had to do some research and came across an answer on GitHub from a user named Imanullah (which you can find here) whom said putting the following code in the `firebase.json` would solve the problem.

```
{ "hosting": { "public": "./dist", "rewrites": [ { "source": "**", "destination": "/index.html" } ] } }
```

The reason why I used Firebase was for two reasons, firstly I have never used firebase, so I thought this was a good opportunity. It fairly straightforward (well normally) and everything is done using CLI tool. It is also secured and fast, and similar to GitHub you can commits deployments and roll back to previous versions which you can see in the screenshot below



CSS Framework

Vuetify is a component framework for Vue.js was used for this project. It is based on the Material Design. One of the reasons this is being used is because you can easily use Google's material design icons, and because it built for Vue thought it would be a good opportunity to use it. Besides that, Vuetify has pwa support too. To see more about Vuetify click here to read their documentation. Vuetify Documentation

Mapping and Geolocation

For mapping I used '*vue-google-maps*' as opposed to leafletJS because Google Maps is more commonly used on websites. When setting up VueGoogleMaps I needed to go to the Google Maps Platform Documentation

At this point, I needed to get an API key to put *vue-google-maps* js file which I gotten from here

You can see in the following code how it is done. (Because of privacy reasons I have replaced the actual api key with xxxxxxxx).

```
import Vue from 'vue'
import * as VueGoogleMaps from 'vue2-google-maps'

Vue.use(VueGoogleMaps, {
  load: {
    key: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,
  }
})
```

In the actual _id.vue (the page with further information of selected), all I needed to do was this:

```
<GmapMap
            :center="{lat:object.location.geocode.lat, lng:object.location.geocode.lng}"
            :zoom="15"
            map-type-id="terrain"
            style="width: 100%; height: 500px"
          >
  <GmapMarker
              :position="{lat:object.location.geocode.lat, lng:object.location.geocode.lng}"
              :clickable="true"
            />
  </GmapMap>
```

Which was similar to the quickstart on the npm vue2-leaflet-map, however for the position I needed to get the location of the POI from the API.

Weather API

In addition to using the location from the API for the mapping, I thought instead of using firebase database, because I did not see how it fitted into my vision. I used [OpenWeatherMap's Weather API](). I used this to show the weather of the selected POI. The weather API uses axios and then uses the lon and lat from the Autoura API.

```
getWeather() {
      this.$axios.get('https://api.openweathermap.org/data/2.5/weather?lat=' + this.object.location.geocode.lat + '&lon=' + thi
s.object.location.geocode.lng + '&APPID=03bc9df889d6cf24f69ebed9ace4d15b')
      .then(response => {
        console.log("Weather below");
        console.log(response)
        this.weather = response.data
      })
      .catch(error => {
        console.log(error)
      })
```

The weather icons were done doing the following:

```
<div class="weather">
                  <img
                    :src="'http://openweathermap.org/img/w/' + this.weather.weather[0].icon + '.png'"
                    alt
                  >
                  {{this.weather.weather[0].main}}
~</div>
```

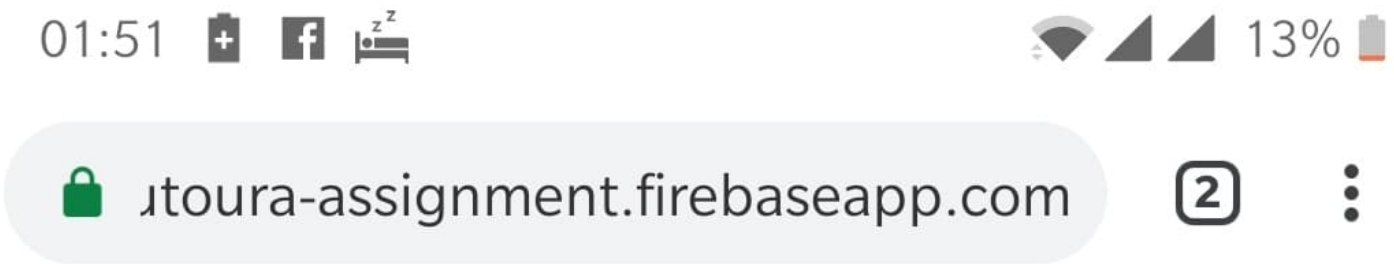
As you can see in the code, I made a div tag tag with a class called weather. Within that I made an img tag. This is where the image for the weather appears using the URL and then it gets the weather data from the this.weather which was made in the script section. It then grabs the weather icon corresponding to the weather for example if it is raining '09d' .png will appear (which is why the png extension is at the end). For more details about the weather icons please check out the documentation [here]()

I wanted to add the icons as a visual representation of the weather.

Progressive Web Application

When setting up this project using Nuxt JS, it allows myself to make it pwa ready out the pack. I added keywords to the meta section of the nuxt.config. Nuxt automatically adds the name, description and other important things to the meta tag, which includes the icons and favicons I added to the static folder (both png and ico). It automatically changes the sizes of the icons dependent of the screen size.

In addition to this I tested it on my mobile and as you can see in the shots below, I can download it to the home screen.



Olu Fadipe ... **Web Design for th...**

This is Autoura

This is the Autoura page, use the drop down to fliter the stops and type of people allowed. Also for users with in wheelchairs, check the accessiblity toggle bottom to show which places are accessible. (If you do not want to use the toggle, the accessible places have a wheelchair symbol o the card). Once you have found somewhere click on th arrow --> on the card to find out more information!

Select Your Stop

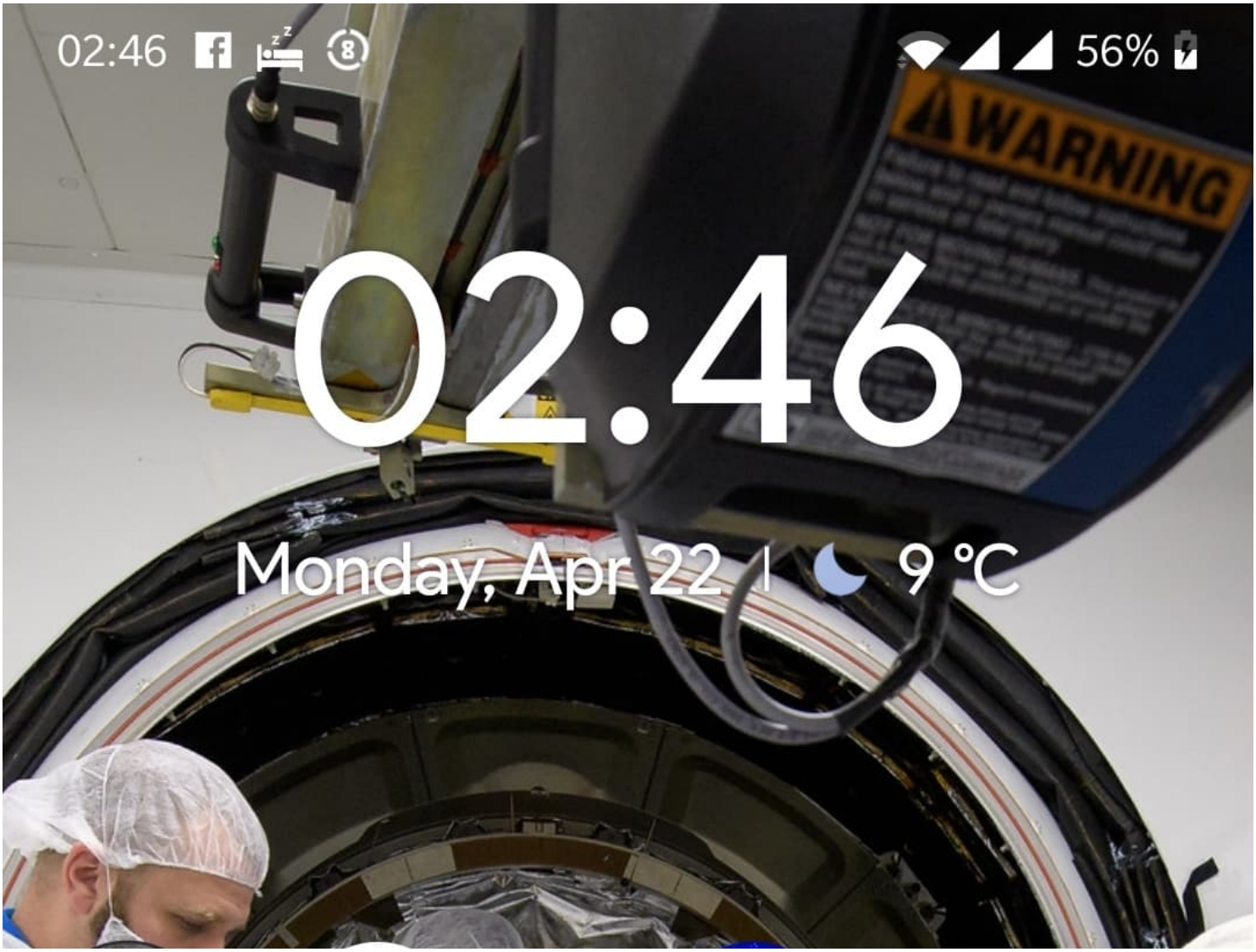
Food

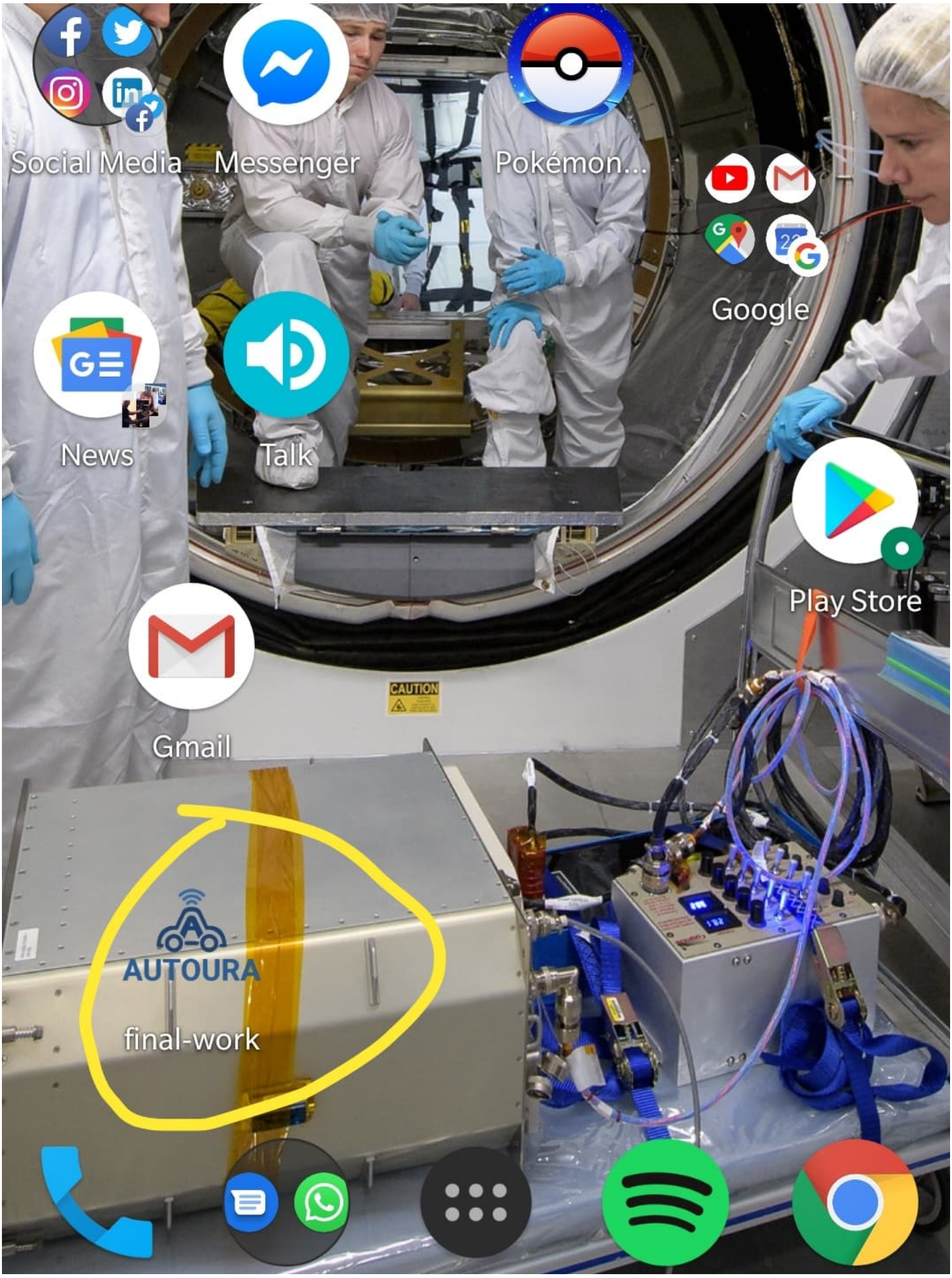
Select Your People

friends

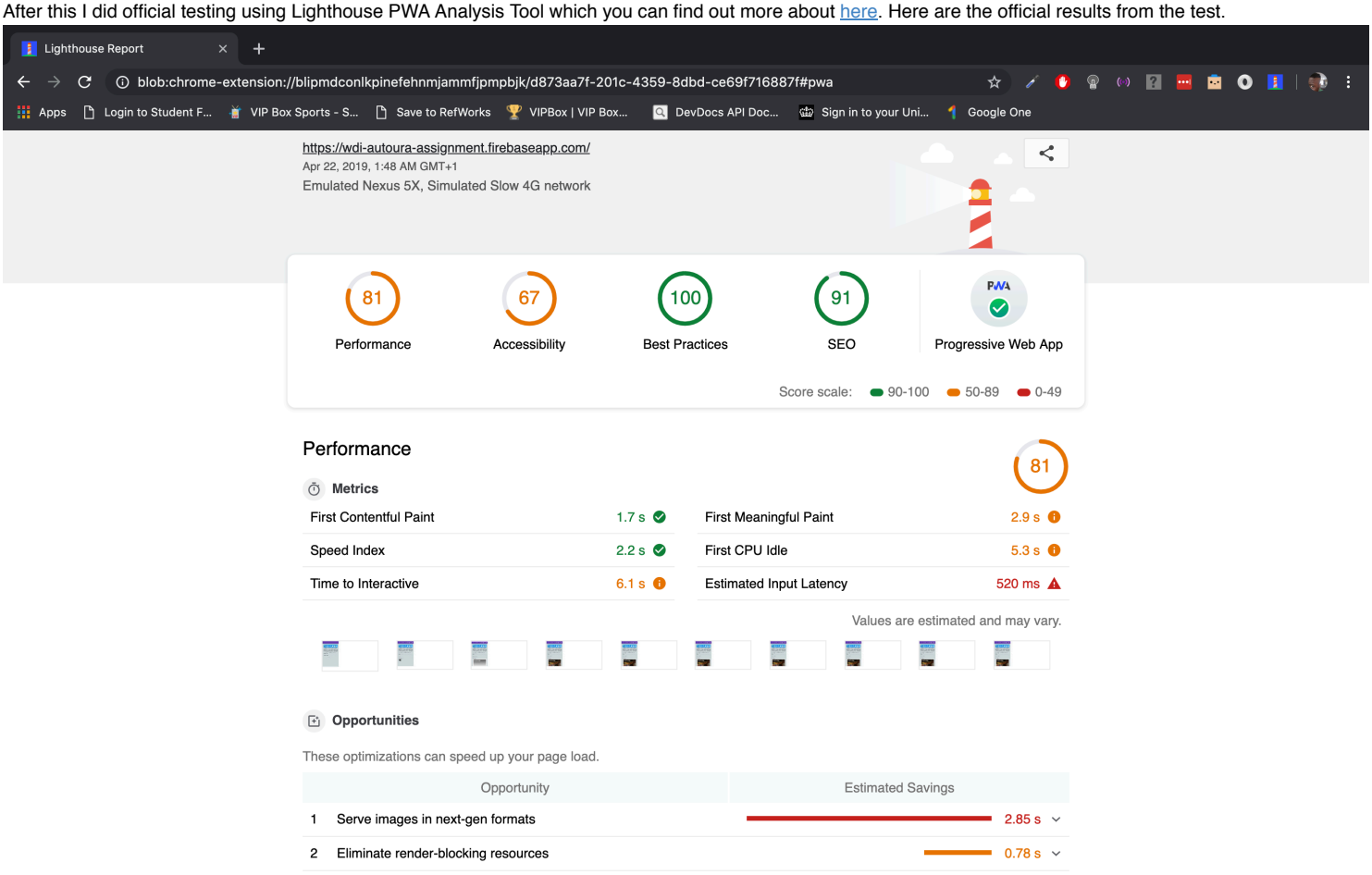
Wheelchair

AUTOURA Add final-work to Home screen




Social Media
Messenger
Pokémon...
Google
News
Talk
Play Store
Gmail
CAUTION
AUTOURA
final-work

After this I did official testing using Lighthouse PWA Analysis Tool which you can find out more about here. Here are the official results from the test.



Officially it is a Progressive Web App, and if you want to see the full results please check the lighthouse folder which has a HTML and JSON files with the full results.

GTmetrix Test Results

I used GTmetrix to provide insight on how well the website loads and provides actionable recommendations on how to optimise it.

After going through the results, I was surprised by the results and the main issue was the total page size which was a bit above average. One of the recommendations was to optimise the images. The full results of this can be found in the GitHub repo here

Loading Animation

I added a loading animation which I got from https://loading.io/css/ which appears on the home page. Which you may notice briefly when you first load the home page.

Reflection

This project was an interesting one, however admittedly the Autouria API was a bit buggy, and was not the best. There were many things I wanted to do with this API that I was not able to do. For example, I wanted to have a separate food and drink section, with checkboxes of different which allowed users to check what food types they want. However, that information was only available for one POI.

Admittedly the mobile web application was not as good as I wanted it to be, especially on the page which had more information about the selected POI, if I had more time, I would had made sure that it was responsive. I would have also done some unit testing (although there is a test automatically created which is found in the test folder), I would have personally wanted to create a few tests myself.

I originally was going to have a splash page but decided to focus on features such as weather API and PWA, but in the future I will add a splash page.

Overall I enjoyed this project and I tried to challenge myself, I learnt a lot throughout such as NuxtJS, VueJS, Vuetify, Firebase among other things.