**Jayanth Kalisi**

**Open-StreetMap survey game**

**Notebook/Future-guide**

**(Based on my Internship from July 5, 2018 – August 19, 2018)**

Note: The structure of this notebook is based on the steps I had taken, not only to learn react/react-native but also start on the game survey user interface. I did not structure it based on the dates I had done certain steps or otherwise. However, it is structured in a chronological order of how I tackled the problems.

**Goal:** my goal in this is to make a questing mechanism. This means that I need to display the map image and write the quests, make the quests interactable, and then build an APK of it.

Also, I used Windows laptop and so, testing worked best on an android. The tutorials I refer to can be refer to tutorials that a Mac or linux user can use as well.

**First, a an article: (**<https://medium.com/the-react-native-log/tales-of-a-react-native-beginner-after-a-year-of-learning-7c72b2c51808>). This article was what I had initially used in order to truly understand react native’s purpose. I got stuck countless number of times learning react native, so I thought of using this as a method of learning. It helped me analyze what to get started on first.

**Contents:**

* **Tutorials (page 2-5)**
  + **AVD setup**
    - **Brief overview of what is different now**
  + **React-native overview**
    - **Brief overview of what is different now**
    - **Navigation**
  + **React**
* **Packaging into APK (page 6-7)**
  + **Init**
    - **Breakdown**
    - **Problems I had faced**
  + **CRNA**
    - **Breakdown**
    - **Differences between the two methods**
    - **Problems I had faced**
* **Json files and how they work on react (page 8)**
  + **Brief overview of what is different now**
* **Ejecting from react native (page 9-10)**
  + **Brief overview of what is different now**
* **Actually creating and calling from the JSON files (page 11-12)**
  + **Structure**
  + **Framework**
* **Displaying the information in List Format. (Failed due to deprecation from .45 native) (page 13-14)**
* **Opening and using OSM mapping. (page 15)**
* **Access into the OpenStreetMap/Access-Map game. (page 16-18)**
* **APIs and Overpass Turbo (19-23)**
  + **Breakdown**
  + **API Trials**
* **Webview in React-Native (24-25)**
* **Merging 2 Displays made (26-27)**
* **Getting the geolocation of a person (28-29)**

**Preliminary note to those reading in the future**

These took me more than a few hours to complete, since the installation process was so tedious. The Android virtual Studio set up, in itself, was a time killer. Also, figuring out that the exact steps necessary was also quite tedious and time consuming. The documentation online has quite a few holes.

The thing is that I did not just follow the tutorials. Every time a concept was introduced, I went back and tried to wrap it up in my brain for tactile understanding. So, a 2 hr video for me was around 4 days of information breakdown, analysis, and filling the holes left behind on the tutorials. I hope this comes in useful to you.

For any extraneous file extensions that I needed to manipulate, I used the notepad (for example, .keystore files). For other .js, .html, .css, java, and .json files I used the Visual studio code IDE software by Microsoft. React native’s developer tools section on the more-resources page (<https://facebook.github.io/react-native/docs/more-resources>) has a list of recommended software to use for certain occasions.

If you are using this as guidelines for the future, I have explanations and changes that need to be implemented based on the tutorials I have found. There are also better methods than the ones I have listed first. I have made sure to take note of that. Please take your time to read through all the content in the block before moving onward.

**Helpful tutorials:**

<https://www.youtube.com/watch?v=KRLLjlpy0r4> : This shows how to setup the SDK (Software Development Kit) and AVD (Android Virtual Device) on android studios. This should only be followed if you have a windows laptop. It is slightly out-of-date, but it was good information nonetheless. Here are a few differences that I found:

The SDK manager options recommended here were still there- Marshmallow build tools and the system image. What needed to be done was to check a box at the bottom right of the window that pops up at the SDK manager window. Once that is done, more detailed items and actions can be found.

The AVD manager option did not appear and wasn't available in the program or the terminal. To access it, I had to start a new project. Also, The AVD manager icon (located in the upper right corner next to the SDK manager icon) was GREYED OUT.

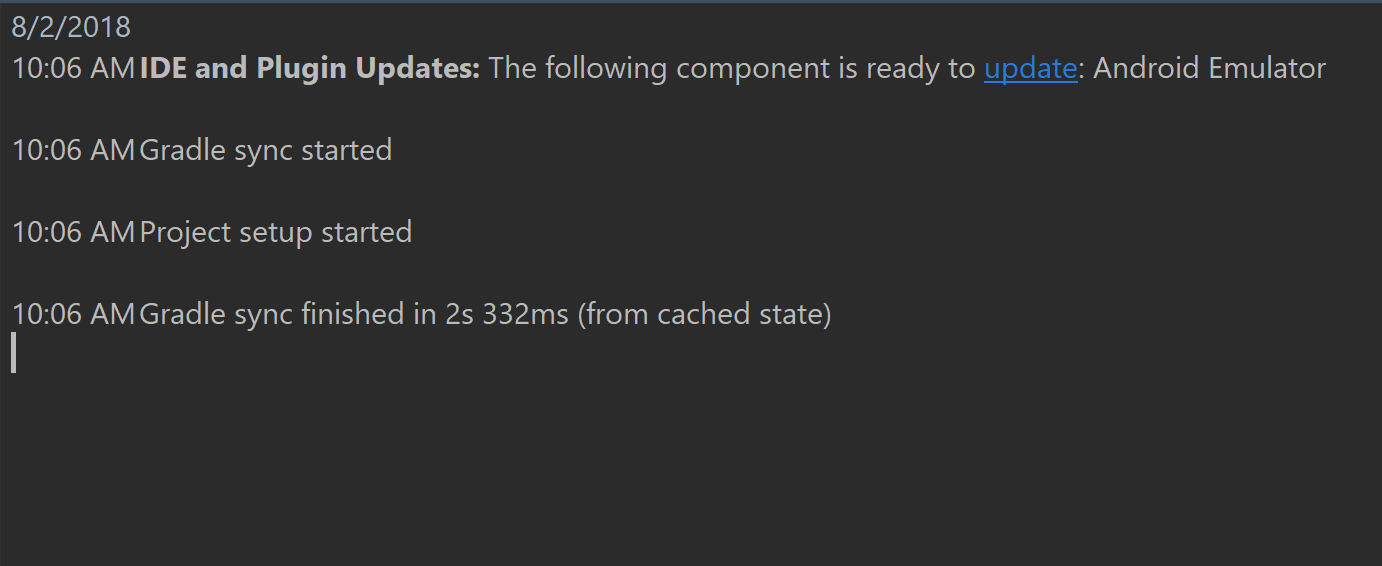
At the same time, I also got an error from Gradle saying that the Gradle Sync process had failed. This appeared at the bottom of the window and below it was a link.

I clicked that link and downloaded the component identified as missing and tried again. It found another missing component and I did the same thing (merely clicking the link below the error) this time Gradle was successful with the Sync process and the AVD manager icon became active.

After clicking on the AVD button, I got a message saying that the Gradle was not loading properly. After I clicked the start icon next to my android virtual device, it started and then subsequently immediately crashed, saying the hardware needed acceleration. I had to go into the system BIOS and had to enable virtual emulator for my hard drive. (took me 4 hours to figure this one out).

I have a windows computer after all, so I also had to make sure that my processor could handle such virtualization. Luckily, it was able to do so.

Setting up a new AVD was no trouble at all from there. The only quirk was that there was a significant delay when I clicked to crate the AVD and when I launched it as well. There was no sign of activity for a little while, but it worked just fine, so don't be too alarmed if you do either of those things and nothing happens right away. (est. 4-5 mins)? That may just be on my end.



My AVD Gradle now looks something like this. Repeating the refresh process made the reloading faster.

<https://www.youtube.com/watch?v=mkualZPRZCs> : This shows a brief overview of what react native is and how to use it. This also is a bit out of date. Here are some things I have found:

When the video-author uses *react-native init ProjectName* you don’t need to do the next steps. You just must use *create-react-native-app ProjectName* instead.

**Note:** the CRNA process (Create react native app) does not help for overall development. It helps you start quicker. If you really need to get the full force of react native behind your app, it would be helpful to look through the **Ejecting from react native** content block that comes later in this document.

I had initially decided to continue with the *init* step. This was what happened and my theory of the reasoning:

I spent more than 1.5 days trying to fix this issue. It continually said development error: 500. I had no idea what to do, until I realized he is using a slightly out-dated method.

* + Although react native users can still use the init process, my computer seems to have a faulty haste module map. The error provided was something like this:



As such, I recommend doing the CRNA process. The init step may or may not work for you since there is a huge bug around with the connection of the init process to the AVD. Doing the CRNA process in command prompt should help. Once that happens, it seems react needs a *npm start*.

The CRNA process also helps in letting you use the app that you built on your own phone. If you want to run the app on your android or iphone physical phone instead of the virtual one, it is best to follow the steps on the command prompt. It does a good job of listing it out.

**Note:** As I did the *npm start* feature, I got a prompt asking if I wanted to load this on my phone. Don’t worry, that should happen. If you decide to use a physical android phone like I did, install an app called EXPO from the android play store to get code to run on it. With iPhone, I believe you need to SMS a remote server. I may be mistaken in the iPhone department.

Doing the display on the virtual device really helped for debugging and checking for any ‘unintended features’. However, it is just as good when testing on your phone. Android uses a QR code Scanning mechanism via the app mentioned above.

My only quirk is that it takes about 2-5 minutes for this to load. The main reason why this is the case is because when booting the process for the first time, the AVD does not have the app installed. After this, it is much quicker, with only 1-2 minutes of time spent per reloading of the bundler.

*Also, the index.android.js and index.ios.js he refers to is not really an issue.* Apparently, everything that I do will be now in one file called **app.js** which will be called in one cumulative **index.js**. They changed things because it was a lot of duplicate code. Now that I am looking at the tutorial again, he copies and pastes a lot of code, and now it is unnecessary to do so – Because there is only one file.

Its still possible to receive data from device specific sensors, all that is necessary is to manipulate the code in such a way. I will try to have a phone sensors tutorial session.

Once I went to the navigation portion of the tutorial, it becomes impossible to continue, since the video-author’s react-native is at native version .40, and right now, I am using the .56 version. During this version break, the navigation had gone through 3 major changes. This changed from navigation, to react-navigation, to screen-stack and tab-stack, and now it is at stack-navigator.

I continued my learning in this tutorial instead: <https://hackernoon.com/getting-started-with-react-navigation-the-navigation-solution-for-react-native-ea3f4bd786a4>

This uses the init protocol and runs via the react-native run-android, so there may be troubles in that regard. I was able to side-step this by downloading the code from Git-hub and then running the code. If, however, problems arise, it would be best to find another tutorial about tab-stacking and the like

The react-native .56 documentation (the version I am using) would be better suited in this regard.

<https://www.youtube.com/watch?v=pgAvVxowaYU> : this is a tutorial to React.js. I only used this in the beginning, when I was unsure of how to use react. I also thought that react worked better on mobile than react native. This is not the case. It is just a snippet of how react works; but it is 2 hrs in all, so it will be time consuming.

I only went through this because I had thought I needed to learn React as well. Learning react helped me understand react native better. However, the tutorial above provide a good grounding for react native in a way, so I think there will be no problems for future react native work.

**Packaging into an APK:**

I’m now trying to figure out how to make the packaging work. If I get used to it now, than I will not need to debug this with a finished game. The first hurdle seems to be that we need verification from android that the app we make is acceptable. There seem to be 2 ways to do this.

**Init and ejection method:**

After finishing the steps on the react-native page

(<https://facebook.github.io/react-native/docs/signed-apk-android>) it is easy to make the APK. I have yet to put this on my own phone, but if the emulator is any sign, it works well.

Some of the steps there are a bit confusing, so here is a breakdown of what I discovered:

* Firstly, the android location/file is only visible to me on the git file I downloaded from the hackernoon website that I used for the navigation tutorial. Otherwise, I can’t seem to find it. It seems it is automatically formed when we type the ‘react-native run-android’ during the init step. I do not know of an updated manner for the ‘create-react-native-app’ method. I will put it down below if I ever come across the method. **(note from the future: yes, I have managed to figure out how this would work it is mentioned at the bottom)**
* Second, the process is tedious. On the site, it shows the alias and the keystore and key passwords, but it doesn’t show you everything else. There is name, location, organization and a few more questions that this process asks. It would certainly have helped me to know more. An example is shown below:
* C:\Program Files\Java\jdk1.6.0\_25\bin>keytool -genkey -v -alias company -keyalg R
* SA -keysize 2048 -validity 10000 -keystore company.keystore
* Enter keystore password:
* Re-enter new password:
* What is your first and last name?
* [Unknown]: John Smith
* What is the name of your organizational unit?
* [Unknown]: Android
* What is the name of your organization?
* [Unknown]: Company
* What is the name of your City or Locality?
* [Unknown]: Albany
* What is the name of your State or Province?
* [Unknown]: NY
* What is the two-letter country code for this unit?
* [Unknown]: US
* Is CN=John Smith, OU=Android, O=Company, L=Albany, ST=NY, C=US correct?
* [no]: yes
* Third, all previously installed builds need to be uninstalled. This means, that all expo run builds must be uninstalled from your phone (previous versions of the same project at least) must be erased before you can upload the newer version.
* Finally, I had trouble finding the */.gradle/gradle.properties*  file. To fix this, I had to go through cmd instead of the file explorer. Just be sure to do that if you are running through trouble.
* Also, when it says you need to type *cd android*, and then type *./gradlew assembleRelease,* on windows, it helped me to use the *gradlew assembleRelease* instead. That helped quite a bit, since there are errors that pop up if you do use the ./ method instead.

**CRNA (Create react native app) version:**

* <https://www.quora.com/How-do-I-build-an-APK-file-from-a-create-react-native-app-CLI-project> this could be the method, I have yet to test it, but all of my sources point to this direction. Hopefully, it is an acceptable source.
  + It works for the first few components. I ran into an error saying it is necessary to say the java packager in exp.android.package at app.json. I just made one up saying “com.taskar.helloWorld”. Note that this is in reverse DNS notation (like the ones used in JAVA).
  + <https://github.com/react-community/create-react-native-app/issues/114> I used this as reference for more details
    - Like the process says it takes around 8-10 minutes in a regular computer. For me it took 15 mins because my computer is quite slow.
    - Success! APK using this method is made. It is much better than the init method that follows. Next time onwards, I will use this for the Open-sidewalk game. This is because this process only needs one step to complete from now on: *exp build:android*
    - To check if the APK is good, all you must do is to drag the APK from the file explorer on to the android emulator. It should automatically install. Once it is done, you must go to the app list and physically select the app. Afterwards, the app should play as it usually does. Otherwise, your code has an error and cannot compile.
    - Once this is done, I can go on to work with my JSON files.

**Json files and how they work on react**

Importing json files from a website was quite simple. The regular *import ‘./filepath/file.json’* or the *import ‘superficialWebsite/example.json’* both work in this regard.

Writing to and reading from the files is the main problem. There is some differences between the multiple documents online. I need to go through them and test them out on multiple dummy projects. Lets do this.

I started following this video: <https://www.youtube.com/watch?v=FPsJhKhDuyw&t=4s> It may be of some use. I am trying to look for a better one with an example project step by step guide. That would be ideal for understanding.

<https://www.youtube.com/watch?v=5vFgqCfggC0>: this is a better resource. Instead of a step-by-step guide, it tells you everything you need for each file. While the explanations are lacking, they can be found while looking at the react native documentation for the particular module.

I have tested how to fetch the data. Writing is what I must do next. It seems that stack overflow may have some answers. <https://stackoverflow.com/questions/37672144/write-edit-overwrite-a-json-file-in-react-native>, this seems to be the website to look into.

**Sidenote:** throughout all of this my project was not building to the android emulator. It was very disappointing. I recreated a project to see if it works. I just have to wait and see now.

It turns out the problem is memory space. I ran so many projects, that I forgot to uninstall quite a few of them. As such, I cannot run more in the emulator. So, I went beck into the emulator and uninstalled all the things I did not need. It seems to be running smoothly now.

It took me a whole day to fix this issue.

Anyway, I found this video (<https://www.youtube.com/watch?v=eBhgwJH5SrQ> that shows how to use AsnycStorage to save data in key value pairs. I need to look through this to see how the make the code function.

Considering the react native documentation asks us to use this to write to .json files, I considered that to be true.

I have looked into this, but I still do not understand what each statement does in the AsyncStorage method done. I will have to research a bit more.

(<http://facebook.github.io/react-native/docs/asyncstorage.html>). This documentation shows all that is necessary to understand AsyncStorage. It makes sense to me now.

**Ejecting from React Native.**

I recently met a guy called Nick Bulton, a PhD student who knows a lot about the ins and outs of react native. He told me that I need to find a way to improve the development process of my app, since the process required to receive data from OSM (open-street-map) needs a better development environment.

As I was trying to figure out how to improve the development of the phone app from what it is, I came across this term – ejecting – while looking for the method. What I found fascinating was that it became completely possible to use this as a backbone of the app.

This link: (<https://github.com/react-community/create-react-native-app/blob/master/EJECTING.md>) helped me figure stuff out. This link also helped me wrap my head around the concept of react-native ejecting: (<https://stackoverflow.com/questions/45251746/reactnative-eject-explained>). I need to re-analyze some stuff because I may need to eject it for the future.

I think I am going to try that in a dummy file. It may come in handy for the future. So, I made a CRNA (create-react-native-app) folder called Ejecting. I followed the steps on the github account. As such, I managed to eject the app with minor issues. One thing to note is that react asks a few questions throughout this process. I answered them in kind, and it was a smooth process.

The one issue I had received was that there were no batch files. I decided to ignore that. From then on, I managed to use the yarn start, and react-native run-android mechanism.

This led me into an error. It states, ‘Application EjectingTrial has not been registered.’ I will be tackling this issue from now on.

The way you solve this issue is by going through the app registry module from React native. For example, mine was something like this:

AppRegistry.registerComponent('EjectingTrial', () => App);

That let me solve this issue.

The tricky part about this is that all the data you would usually get on your cmd type systems gets ignored in the filter. Node and even React does not know what the problem is. It complies smoothly. The only reference you have is on the phone.

With this, Ejecting is a success!

One thing to note is that there are now split files for android and IOS. This means that, when I am making the questing mechanism, I would be fine. However, when I am building the game as an APK or other type of system, I would need to be able to manipulate the folders in such a way as to build them properly.

After ejecting, I realize that the command prompt window which ran npm/yarn start, is now actually telling me the basics of the errors that I had made. I purposefully maintained a lot of errors to see what pops up for each type of error. It seems to be picking the errors up pretty well.

**Actually creating and calling from the JSON files**

**Structure:**

The JSON files I am going to create is quite structured, since each component is going to be used by others. So, I have come across 5 distinct types of questions and actions that are worth doing.

First, is the standard true/false type question. For this, I have labeled it **Boolean**.

* The main thing is that there can be only 2 answers. It should be pretty self-explanatory.
* I did not write true or false in the json file, since it should be easily achievable in the code in react native. I can just write a conditional, like *if the type is Boolean than make the only options available to be yes or no.*

Second, is the **integer**. This is the standard number estimation question.

* The main premise is that the player should be able to decide on the number of people in a general area.
* I have added a **minLimit** feature to this integer. This tag would allow for the answers to be legitimate. For example, some of my questions have the value of minLimit as 0, and others have it as 1.
* This is a number, so in the code I will need to write a statement to weed out any string characters if possible.

Third, is the **limit** type question. It is the ‘on a scale from 1-10’ question.

* The main purpose of this is to cater to personal surroundings. If people mind noisy locations we can use this to decide better movement patterns.
* We could also use this for the pollution levels.
* I am not specifying the limits on the react code, but on the json file. So, tags like **minLimit** and **maxLimit** exist. And need to be called.

Fourth, is the standard multiple choice question. I have labeled it as **MCQ**.

* The main purpose for this is to adhere to sidewalk material. We can also use this for the curbs and such for acknowledging how they look when numbers are not viable.
* I structured the code with a number of options (**numOptions**) tag. This means that the code in react will be calling this tag specifically.

Fifth, is the **picture** based question.

* Honestly, this idea is the most troublesome one. Currently, I have no idea how to interact with the camera of an android or IOS phone to gain access to the camera. I need to figure this one out.
* Basically, in order to know where their home is, and the general surroundings, this idea was formed. It is a ‘for-fun’ idea, and worst case, does not need to be implemented. However, if it were implemented, the main thing I would do is to write a method/function in react which is called every time the camera question pops up.

**Framework:**

Next, I will explain the frame of questions, or the topic of questions and how I have structured them. The main problem here is that if this does not make sense to anyone other than me, then we are in a huge debacle. What I am doing here is to make confusion as limited as possible.

* When you first open the json, you start off with the quest tag. This tag is of utmost importance since this is the topic of the JSON file.
  + After, when you zoom to a lower level from the quest tag, you will see two specific type of quests: modifications and creations. I have labeled them as **existing** and **new.** 
    - With the existing questions, it is possible to make the modifications to preexisting data. Something like an overwrite. We need this to make changes in case some one else marked the data incorrectly, or it has changed recently. AsyncStorage is the way to do this apparently.
      * The other thing is that each of the existing questions have the ability to skip. What I mean by this is that based on the data give, it would be possible for the player to skip the entire quest. I plan on using this mechanism to toggle the showing or hiding of certain quests on the mapping view.
    - With the newer questions, it is possible to start collecting data from scratch. It is useful for noting landmarks and other key locations.
      * Underneath both of these levels, you will see another type of questions. In this case, it refers to the topic of questions you are tackling. The first thing asked is whether you are sure you want to answer these questions.
      * The second thing it looks for are things like bus-stops or homes, intersections, or sidewalks. That was quite entertaining.
    - The other thing is that each of the questions have the **skip** tag available. Through this, it is possible to toggle the question skippable. Based on answers to pervious questions, we can implement a different format each time.

**Displaying the information in List Format. (Failed due to deprecation from .45 native)**

If you remember the navigation stack and the tab stack tutorials, there is one main component for displaying the text: Lists and List Items. The main way you would do this is by adding the information from a json file or an array into this by mapping the array.

There is one big problem I have yet to solve – that being the react native elements not found problem.

You remember the red image that you got for the react natve run android format that we did? Use something like this for a Refresher:



Yup, that same problem is occurring to me as well. This seems to be because of the location of the react native elements. They appear to not be on the haste module map file. This is the same problem occurring on the react native forms.

The solution to this Is found! It works.

Ok, the way you tackle this is by deleting the line of code saying: “import {List, ListElement} from ‘react-native-elements’;

That line should be completely deleted. Also, the location of your JSON file should be specified so that another error does not pop up.

The problem now is that I cannot use one of the files I had previously routed it to. I need to look into that for the future. For example, I call **<prompts />** inside a method. It is a self closing tag, So I had initially thought that there will not be any problems. The problem is as follows: I have a method which I call in the render method as my only call. I route it to a file called ‘prompts.js’ using the tag mentioned above. The problem with this is that whenever I call the file to be instantiated into my code, It immediately fails.

The red screen pops up again, with the notation that it is an improper method of calling this method. It then swaps back and forth with the unable to resolve module portion of the problem.

I then tried using the code inside the file, and what do you know, the scrollview, List, and Listitem imports are all outdated. As such, they are unavailable to gain access to from react-native.

**Opening and using OSM mapping.**

The main part about OSM is the website. (<https://www.openstreetmap.org/>). The tutorial given is quite easy to understand. The main terminology for each key, for example highway = footway, footway = sidewalk. That kind of nomenclature I have yet to gain full control over.

I started to go through this tutorial website: (<https://wiki.openstreetmap.org/wiki/Beginners'_guide>) given to me by Nick. This way, I will be able to understand and get used to the style of OpenStreetMap. The most useful page for me was the **See your work and start using data** page. It is quite interesting.

The second issue at hand is that the tediousness of manually editing the locations. OSM is great, but the main issue is how time consuming the addition of data is. It takes around 5 minutes to completely get the hang of a single data point (like nodes, ways, areas, and relations). I now see the necessity for something like this game that I am making. Street complete based game is quite intriguing.

I just spent over an hour trying to learn OSM nomenclature. It fits pretty well. I immediately got the hang of it. There are a few tags that I still have yet to discover, but it helps me quite a bit. The one problem I am having is that there is no place to just practice and explore.

* For example, the tutorial helped me quite a bit because I could just run around manipulating the nodes and ways, and it would be fine. I would also manipulate the polygons (or areas as I have learnt they are called). Now that the tutorial is over, I cannot find something to do that helps me learn by doing. Since I learn best by doing, that certainly diminishes what I can do.

**Access into the OpenStreetMap game.**

I need a way to display the map. I know that there are problems with the writing mechanism, but if I can gain the ability to access even a simple question, I would have made strides. As such, I need a way to display an icon of a sidewalk or a busstop.

To do this, I have just gained access into the original version of the OpenStreetMap game. This was the one made by Nick Bulton, a PhD student working on this initiative. He gave me access into this. I must now get used to this interface. (<https://github.com/AccessMap/accessmap-uw-native>) is the link I was given access to.

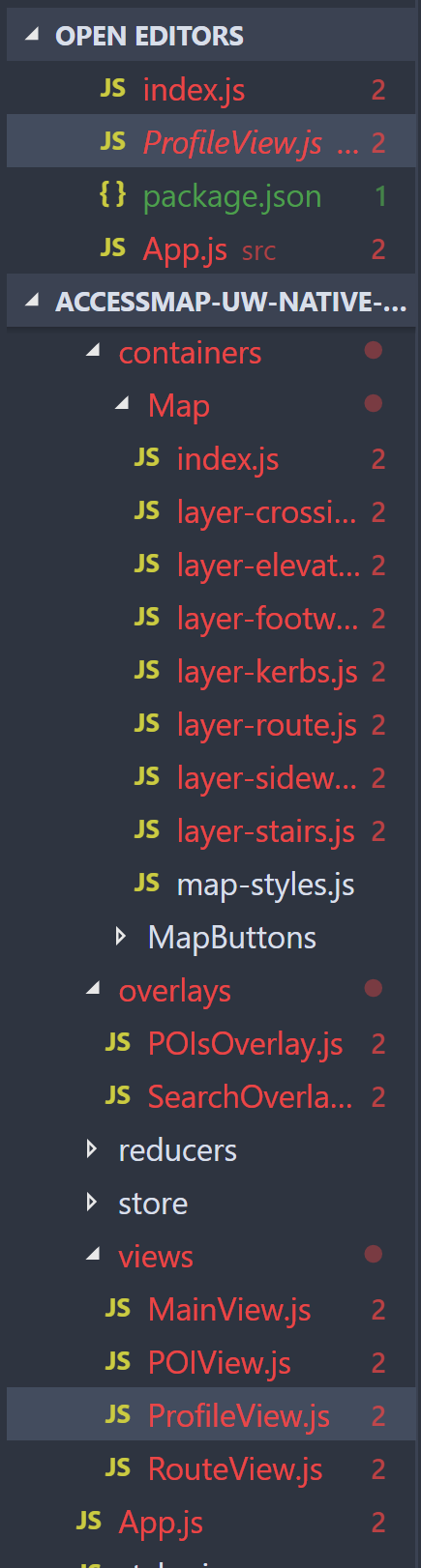
After following through the steps on the read me file, I encountered a problem. The thing is, I downloaded this game onto my computer. Then I tried to follow the steps in the ‘ReadMe’ file. It did not work out for me one bit. I think this has to do with the node modules folder, which I cannot seem to find in the app. I then did an npm install, which should have fixed the problem, but it didn’t. There were **17 vulnerabilities**. I do not think that any of them are really impactful to the game at large.

I installed node onto this copy of the project. It does seem to work. I then npm start and react-native run-android. It works for a few minutes, but later it just fails. The error log states that the keyAlias and keyPassword on the gradle is missing and reported as null. This, if you remember, is what I stored when making an APK I think I should try using these. Let us see where this takes us.

I am looking for the file called ‘accessmap-android-key.keystore’ I cannot seem to find it. I went through the command prompt, file explorer, and others but still cant seem to find it. It says that the file does not exist. As such, I think I should try creating a new .keystore file. This may be helpful to update for the future.

But I don’t want to create it yet. I do not want to make a huge mistake with regards to how the game is set up. So, I think I will just go through the code and try to see how it has been structured. The funny thing about this is that a lot of files are plain js files while only the main ones are structured in a react-native format. I had thought that it would sound funny if I were to use a mixture, but it seems that the usage of regular .js files in an interconnected manner with the React-native .js files is actually encouraged.

Immediately, as I open up the index.js file, 2 errors pop up. I had not even touched this folder or the files in it beforehand. Yet, as I am clicking through each file, I get 2 errors to pop up. It is always 2 or 0, but most have 2 errors. Upon further research, it shows that almost all of the src folder has those two errors.



It seems that the problem is these lines that occur everywhere in the project:



Of course, the class is not always called RouteView, but you get the Idea. All of the files marked in red have 2 errors. It is quite frightening that this error was here the moment I opened the project, yet VS code could not pick it up. Probably it is a missing import somewhere.

I have looked up this error, yet I cannot seem to find any results. It is probably a VS code problem in itself because the ‘type’ key word is being referred to and it does not throw any errors. It may be something bigger than what I had initially thought.

It seems my deduction was correct. I found this online, a template if you will, for what the intended organization for a standard react file is/was:

import \* as React from 'react';

type Props = { /\* ... \*/ };

type State = {

count: number,

};

class MyComponent extends React.Component<Props, State> {

state = {

count: 0,

};

componentDidMount() {

setInterval(() => {

this.setState(prevState => ({

count: prevState.count + 1,

}));

}, 1000);

}

render() {

return <div>Count: {this.state.count}</div>;

}

}

<MyComponent />;

I think that those that wrote this code did not realize how much react-native had changed over time. Apparently, the props literally refer to the properties, and the state literally refers to the state in the component. Since VS code picked it us as an error, it does not seem to understand the newer versions of the code. It is surprising that VS code picked it up as an error, now that I am reading this article: (<https://stackoverflow.com/questions/47853796/what-is-the-syntax-after-extends-component>)

I think what I will do is the following: I will go through each file and list the names of the files with errors below. I will then delete the error above.

I got to this conclusion because I usually do not say (type props={}) in my file. I would usually ignore such things. If this works with the different method, I will be glad of this case. List of files:

Src:

Components:

Default header/index.js

GeoCoderList/index.js

SearchCard/index.js

SearchGeoCoder/index.js

Waypoint/index.js

Containers

Map

Index.js

Layer-crossings.js

Layer-elevators.js

Layer-footways.js

Layer-kerbs.js

Layer-route.js

Layer-sidewalks.js

Layer-stairs.js

MapButtons

Index.js

Overlays

POIsOverlay.js

SearchOverlay.js

Views

MainView.js

POIView.js

ProfileView.js

RouteView.js

App.js

Index.js

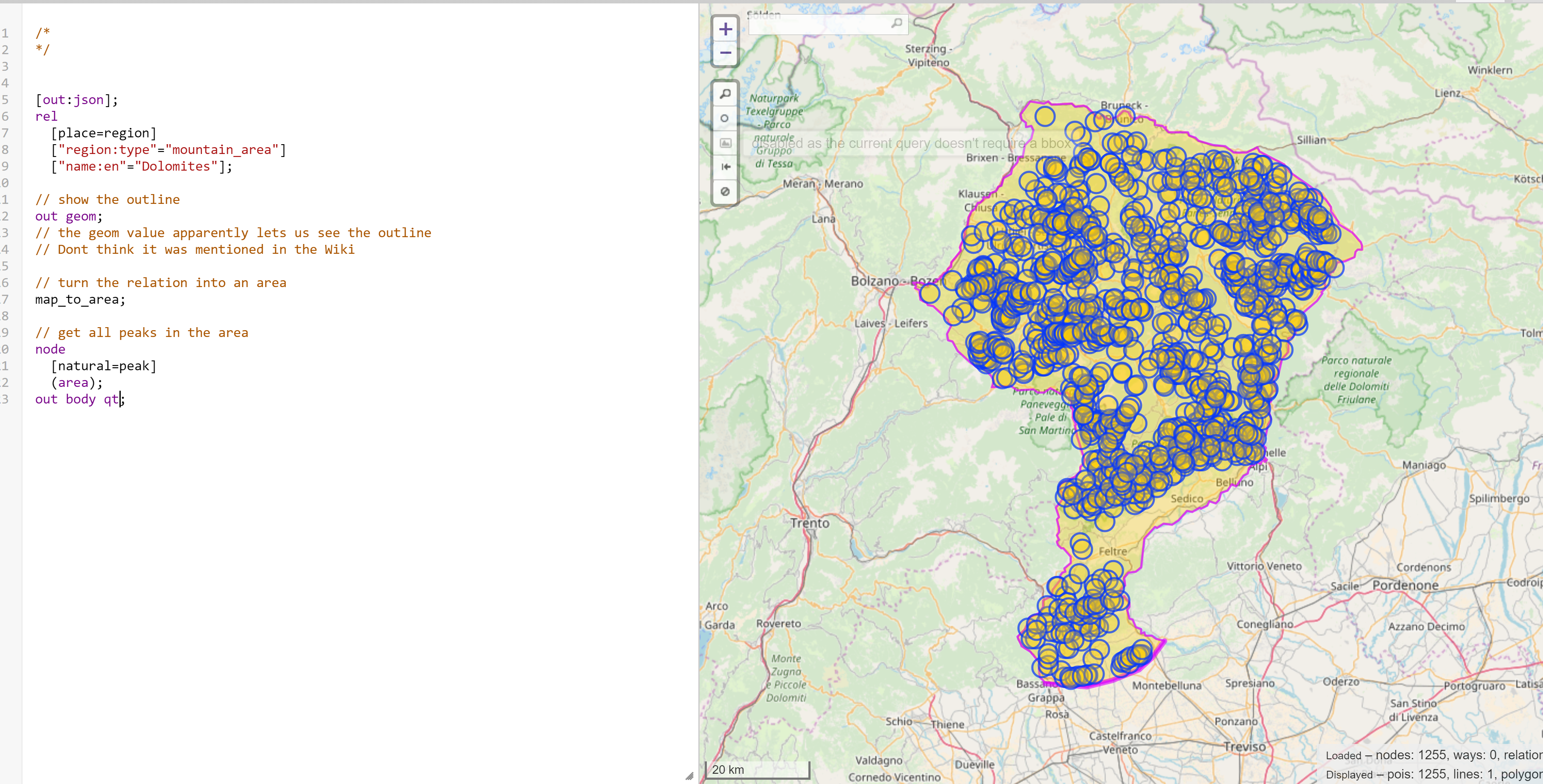
OK, all the errors are fixed, and no new errors are popping up. I can now finally go to understand the way this new project is structured. This will take a while.

**APIs and Overpass Turbo.**

(<https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82>) This is a website which first gave me an overview of how APIs work. In case you have no idea what APIs are, I thought it would benefit you for the future.

I recently met with my professor, Dr. Anat Caspi. She recommended me to look into something called Overpass Turbo. (<https://overpass-turbo.eu>). This is the website for those that learn by doing. Comparatively, there is a website which shows learning by reading (<https://wiki.openstreetmap.org/wiki/Overpass_turbo>) This website shows the minor details for how Overpass turbo works.

This is an example of how Overpass Turbo looks like:



**Note:** I am using a browser called **Vivaldi,** a chrome-based browser for people who want more control and details about web usage. It poses no problems in running Overpass turbo online editor (the first link). I am saying this because the website requires a newer web service/browser in order to function. Examples of this include Opera, Chrome, and Firefox.

*Breakdown:*

I am currently looking through the Overpass turbo wiki and have found this section called exporting results at around 66% of the way down. It looks promising. I found this article which shows a lot about the react native side of things. (<https://blog.hellojs.org/tales-of-a-react-native-beginner-why-graphql-is-the-real-mvp-9169b60b6f92>) Although I think that it is not the method that I will be using, I think it is a great way to be learning the way react native queries work.

It seems that overpass runs on its own special Query language (QL). It also has its own special XML based language. It is slightly confusing since using other third-party QLs would be easier, since they are more commonly used. Its syntax is quite peculiar, since it uses quite a bit of C style syntax. For example, the sentences usually end with a semicolon.

It is quite fascinating to say the least, especially since the language flows from top-down. This means, that the first sentence I use, will be the first command executed (like Java and C). For more info, use these links: (<https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL>), and (<https://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide#The_Overpass_API_languages>)

* The first link goes through all the proper syntax, wording, and methods available in Overpass QL. The second gives a broader perspective on why Overpass QL and Overpass XML work the way they do. I would recommend reading through the second link first. This way you can have a background knowledge going in to the first link.

**Note:** when retrieving data, it seems my laptop has a maximum capacity for handling around recovered. I know this because I crash tested it. As mentioned before, I use **Vivaldi**, a web browser with crashed data like this readily available in graphical and tabular format. My browser tab crashed at about 107.6 mb. Overpass automatically sends a warning label at 1 Mb it seems. If the data recovered is more than 1 Mb, then there will be a pop-up. This means that I would need to make a data channel for getting the data via code. I should not run such large pieces of retrieval data on my browser.

* Note on Data: The data I received apparently comes from a data type called ‘set’ These sets are what my input and output will be. I need to make sure to call on the set I sent the data to. For example, if I sent the data to set ‘A’, I need to call ‘.a’ in the output statement. Otherwise I need to call ‘.\_’
* All the sets I create are automatically given global visibility. To reword, they are global variables. Always.
* I need to use -> syntax for writing to specific sets. For example,

**(**

**node[name="Foo"];**

**node[name="Bar"];**

**)->.\_;**

* The flow must always be “settings first, then queries or blocks”. This will allow for the manipulation/selective-choosing of the data. Before it is called. The calls must occur later, then ‘printed’ in json or xml format
  + Key settings include:
    - Out:
      * This controls the type of output. For example, the common output will always be XML, but you can selectively change the output file type, by saying “out:json” instead
    - Timeout:
      * This controls when (in seconds) the code will ‘eject’ from running. To reword, this setting selects when the code will stop running, incase anything cannot be found.
    - Maxsize:
      * This selects the maximum amount of data that can be transferred. By default, it is a 512 Mb, or .5GB. It seems that most computers will stall at 100Mb, and at peak hours of data usage, data about 2 Gb of data will be unreasonable.
      * This number is actually very large. We will almost never reach even .5 Gb. The server will probably say no to the magnanimity of this data, but it is possible to reach that amount.

*API Trials:*

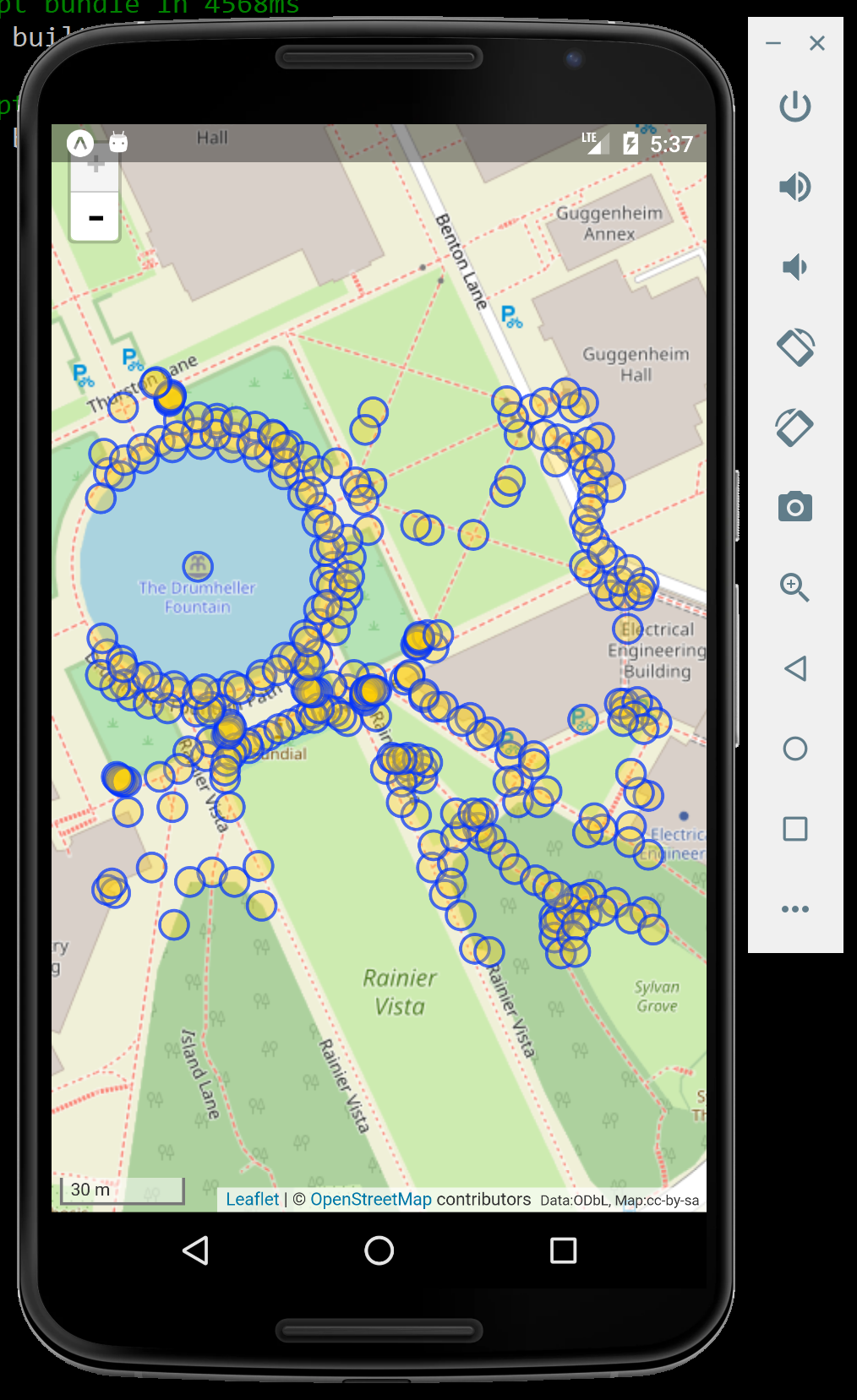
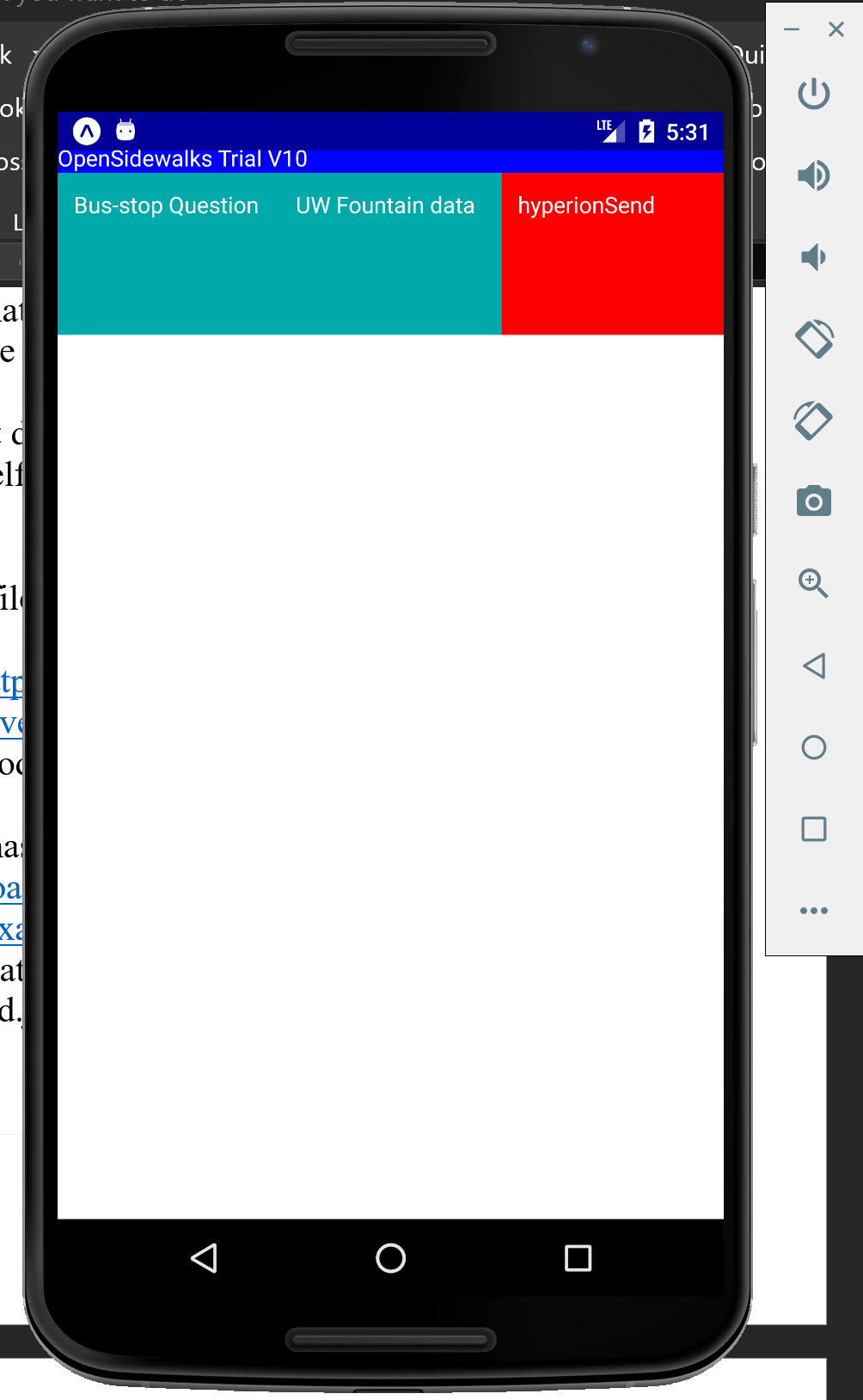
(<https://facebook.github.io/react-native/docs/network>) This link shows how to get APIs from other websites via react-native. This looks promising. I think that this is the way to go. I will start implementing this into the survey after I figure out how to get specific API calls.

* I am also looking into the accessmap-uw-native react native game that was shared with me by Nick. This seems to throw a few errors as mentioned above, but it seems to have resolved itself.
* Ok, I tried the get method on the react-native page, and it works. Barely.
  + I am not saying that the method is outdated, just that this will give unexpected data. This is what I had done:
* var request = new XMLHttpRequest();
* request.onreadystatechange = (e) => {
* if (request.readyState !== 4) {
* return;
* }
* if (request.status === 200) {
* console.log('success', request.responseText);
* } else {
* console.warn('error');
* }
* };
* request.open('GET', 'https://overpass-turbo.eu/map.html?Q=%2F\*\*%2F[out%3Ajson]%3Brel%20%20[place%3Dregion]%20%20["region%3Atype"%3D"mountain\_area"]%20%20["name%3Aen"%3D"Dolomites"]%3B%2F%2F%20show%20the%20outlineout%20geom%3B%2F%2F%20the%20geom%20value%20apparently%20lets%20us%20see%20the%20outline%2F%2F%20Dont%20think%20it%20was%20mentioned%20in%20the%20Wiki%2F%2F%20turn%20the%20relation%20into%20an%20areamap\_to\_area%3B%2F%2F%20get%20all%20peaks%20in%20the%20areanode%20%20[natural%3Dpeak]%20%20(area)%3Bout%20body%20qt%3B');
* request.send();
  + - I used the method described above, and swapped the link in the request.open function with the export/map/interactiveMap link. This would give the interactive map that Overpass generates.
      * I call this method based on a button press event. I did this so that I could make sure of when the method is called. It prints out ‘success’ and then the data when it works. Otherwise, it warns of an error.
      * This get statement that I describe above in the black box, is using a specific URL. I need to find a way to get the access to the API itself on the computer. If I can keep this method and then do multiple queries via my phone, that would be the dream.
    - The end result, or the data which I was able to access, was a lot more surprising. The thing is, I got a whole html file instead of a xml or json file that I was expecting. I also did not receive the ability to manipulate anything else. I just received the file, in what looks like a read-only capability.
      * This also means that I probably cannot add anything to the file itself. It also does not display on the phone. It just gave me the raw data. I need to test this to see if this is true
      * The reason this is a hinderance is because I don’t want to go through and type on the html document, saying “place icon here”. If I can just display this as a map with geojson based coordinates, then I can just use lat-long data to add the quests. This method which I think I will be forced to do seems unproductive to me. I would rather have a better working method than that.
    - This was completely unexpected. I need to investigate how to manipulate the data. I think the files Nick gave me should help.
      * On the accessmap-uw-native file, the container folder (the folder which contains the map) shows to me the mechanism the map is being used.
      * I still don’t understand this method that they have. I see that there is something called mapboxGL, but other than that, there is no reference to the map anywhere. These containers seem to contain the headers and other items. Everything that is not the map.
      * It seems that the mapboxGL is a node modules file located here: ‘@mapbox/react-native-mapbox-gl’. This resides in the node modules folder that is built by the project.
    - The Mapbox file, under node modules folder, seems to be located under the index.js file. It refers to itself as a react-native component, which I can understand and would like to use. The problem is that I don’t really see where they are getting this map data from. This looks like there might be already downloaded data taken from the OSM webpage.

**WebView in React-native**

As I mentioned before, I managed to get data for a HTML file on react native. The main problem was that I could only get HTML data, not xml or otherwise. So I thought to myself, if I can manage to show this data in an interactive web-view, that would substantially improve the efficiency of things I have to do.

As such, I went into research. I wanted to find how to load a HTML file onto the app.

* I first started with the local HTML files.
  + I copy-pasted the information I had received on my console via the query method. This gave me a template.
  + I then went online, and found this link: (<https://stackoverflow.com/questions/38481445/react-native-write-into-file>) which helps me write to files, and this link: (<https://stackoverflow.com/questions/42851296/react-native-load-local-html-file-into-webview>) which would help me load HTML files via a module called WebView.
  + The result went just as expected – it failed terribly. I cannot explain how many errors I had run against.
    - Note: after loading the project again, I had lost the error messages I had received. I apologize for not being able to depict this data in picture form.
  + The main reason for this failure is that all the images and settings variables that the query refers to, only exists on the net – the website itself.
  + So, I thought to myself, “If I cannot depict this via a local file, would it not be easier for me to show the map itself online, as if I am showing the website itself?”
* So, I went onto the Website-based HTML files.
  + This made me run into this article <https://medium.com/spritle-software/embed-responsive-webpage-on-a-react-native-app-8cd418714bde> of someone literally doing just that, but with the same module <WebView> that I was using.
  + The author of the previous article has her source code stored in this location: <https://github.com/spritlesoftware/load-webview-on-reactnative/blob/master/WebViewExample/index.android.js>. As you can see, she uses an older version of react native. This is clearly visible when you see that her file was called index.android.js, instead of app.js.
  + I tested this code out, and it works.
  + 
  + The first phone depicts what the code used to look like. Now, this next phone is showing what it now looks like.
    - As you can see, the dots depict the nodes that were available in the location of UW. I did not use GPS location to set this. If I had the capacity to do something like this, it would be great. However, this is just a static URL of a location I had initially queried on the internet.

My next steps are the following:

* I need to: (ordered from most importance to least)
  + Figure out how to retain the information in the map.
  + figure out how to merge phone one and two’s displays, so that I at least have a header going into the game
  + use GPS location to automatically track if this will work on a phone.

**Merging the 2 displays made.**

The main thing I needed to do was figure out why the aforementioned two phone screenshots did not look alike. I looked into the previous two links (on the blog and its source code for WebView) and saw that it showed a header, while mine had none.

This problem arises from the styles presented in each view. I think what has happened is that the initial ‘flex’ amount for the styles placed in WebView is 1. When I placed a <View> tag encapsulating the previous WebView, which initially had a flex value of 1, it caused a confliction with the <View> tag’s flex value, which was at a default 0.

To remedy this, I changed the <View> tag which encapsulates the WebView to have a flex value of 1. I have pasted the code below for details.

<View style ={{flex:1}}> Annotation: 2

<View style={styles.myView}>

<Text> </Text>

<Text style = {styles.myText}>OpenSidewalks Trial V10</Text>

</View>

<View style = {styles.container}>

<TouchableOpacity

onPress = {this.onPress1}

style = {styles.v1}

>

<View>

<Text style={{color:'white'}}>{this.state.questTypeNew.busstop.name}</Text>

</View>

</TouchableOpacity>

<TouchableOpacity

onPress = {this.onPress2}

style = {styles.v1}

>

<View>

<Text style={{color:'white'}}>UW Fountain data</Text>

</View>

</TouchableOpacity>

<TouchableOpacity

onPress = {this.onPress3}

style = {styles.v3}

>

<View>

<Text style={{color:'white'}}>hyperionSend</Text>

</View>

</TouchableOpacity>

</View>

<View style={{flex:1}}> Annotation: 1

<WebView

ref={WEBVIEW\_REF}

automaticallyAdjustContentInsets={true}

source={{uri: DEFAULT\_URL}}

javaScriptEnabled={true}

domStorageEnabled={true}

decelerationRate="normal"

startInLoadingState={true}

scalesPageToFit={this.state.scalesPageToFit}

/>

</View>

</View>

As you can see, the flex value of the <View> tag labeled Annotation: 1 is 1. That was the code described initially. All I had to do was mirror this in the <View> tag labeled Annotation: 2, by labeling this to also have flex: 1.

I think what happens is that when the two types of styles collide, it causes a disturbance, which makes it such that the WebView no longer functions properly.

Problem solved!

**Getting geolocation of a person.**

With the basic UI Layout complete, I decided to try my had at getting the location-coordinates of a person. Based on this, I started researching. If I could get this done, I would have something to show for myself based on the time I had spent in the UW – 7 weeks.

So I came across this react-native documentation: <https://facebook.github.io/react-native/docs/geolocation>, which looks like it was made for me. I looked through it to see how to access the GPS location of a person.

There are three main steps to follow. I will now say where I got stuck and what I had done.

* One, the ejection from CRNA. I usually use CRNA unless I absolutely need to eject, because I have gotten used to it. It seems that the only way to access the GPS location is by doing so. I did. This time I chose the regular React-native based format.
  + I tried testing out the ejection, to see if it was achieved successfully. Spoiler alert, it wasn’t. It seems that my mistake was to use the **react-native run-android** method from the get-go on the cmd. What I had to do was do a **npm run start** in the beginning. It was only then that I should have done the run-android method.
  + After this, it was a successful ejection. Yay!
* Two, I needed to use something called the PermissionsAndroid API, which is inbuilt from the react-native. I imported this from react-native, and got to work. I made this basic function, which I modeled after this function:

async function requestCameraPermission() {

try {

const granted = await PermissionsAndroid.request( Annotation: 3

PermissionsAndroid.PERMISSIONS.CAMERA,

{

'title': 'Cool Photo App Camera Permission',

'message': 'Cool Photo App needs access to your camera ' +

'so you can take awesome pictures.'

}

)

if (granted === PermissionsAndroid.RESULTS.GRANTED) {

console.log("You can use the camera")

} else {

console.log("Camera permission denied")

}

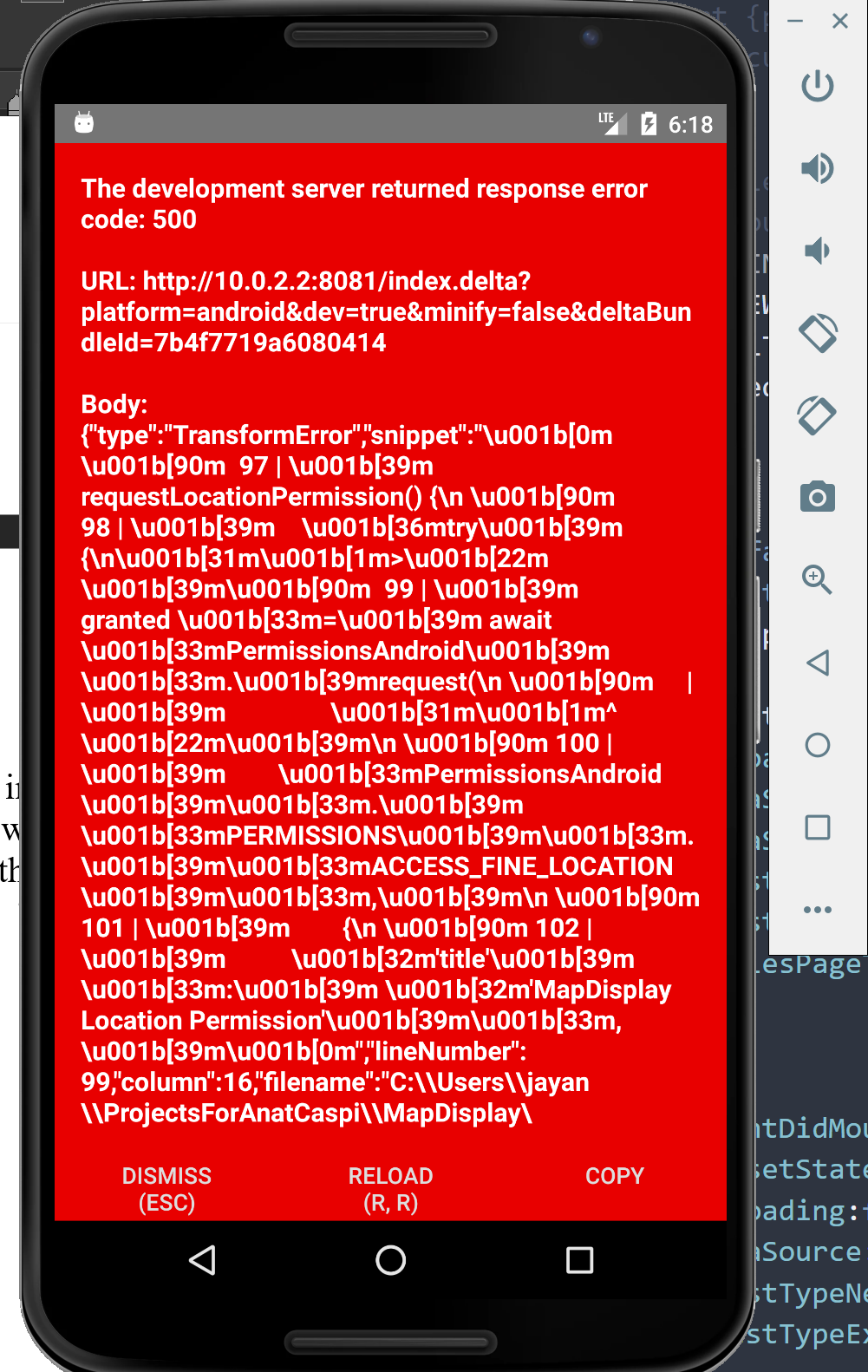
} catch (err) {

console.warn(err)

}

}

* + I then implemented this into the code. Spoiler Alert! It did not function at all. There was no pop-up at all. I could not understand what was going on. So I tried a few things.
    - The first thing I tried was to change this into a function inside the export default class method. This functioned terribly. I met the old friend development error: 500



* + - This problem is apparently a Syntax error according to the error displayed above. Since the await key-term, marked Annotation: 3 is restricted to only async functions, I cannot put this on a separate function inside the class/object. It must stay outside the class/object. I am thinking of exporting the function alongside the rest of the object to the index.js method, to see if it is possible to make the permissions function work properly.
      * Just tested. Doesn’t work. I don’t know why this is the case. I will try a few more things. I may have to give up on the geo-location for now.
      * I say this because I only have 2 days left in the internship. Although I can work on this afterwards, I have been told to create a git-hub repo to upload this to. It should work, but incase it doesn’t, that is my next plan.