CAB230 Assignment one­­

Mitchell Robert Benjamin

N11099887

CAB230 Assignment one

VolcanoNerd.com

Contents

[Introduction 3](#_Toc102596191)

[Purpose & description 3](#_Toc102596192)

[Completeness and Limitations 3](#_Toc102596193)

[Use of End Points 3](#_Toc102596194)

[/countries 3](#_Toc102596195)

[/volcanoes 3](#_Toc102596196)

[/volcano/{id} 3](#_Toc102596197)

[/user/register 3](#_Toc102596198)

[/user/login 3](#_Toc102596199)

[Modules Used 3](#_Toc102596200)

[Application Design 3](#_Toc102596201)

[Navigation and Layout 3](#_Toc102596202)

[Usability and Quality of Design 3](#_Toc102596203)

[Accessibility 3](#_Toc102596204)

[Technical Description 3](#_Toc102596205)

[Architecture 3](#_Toc102596206)

[Test plan 3](#_Toc102596207)

[Difficulties / Exclusions / unresolved & persistent errors / 3](#_Toc102596208)

[Extensions (Optional) 3](#_Toc102596209)

[User guide 3](#_Toc102596210)

[References 3](#_Toc102596211)

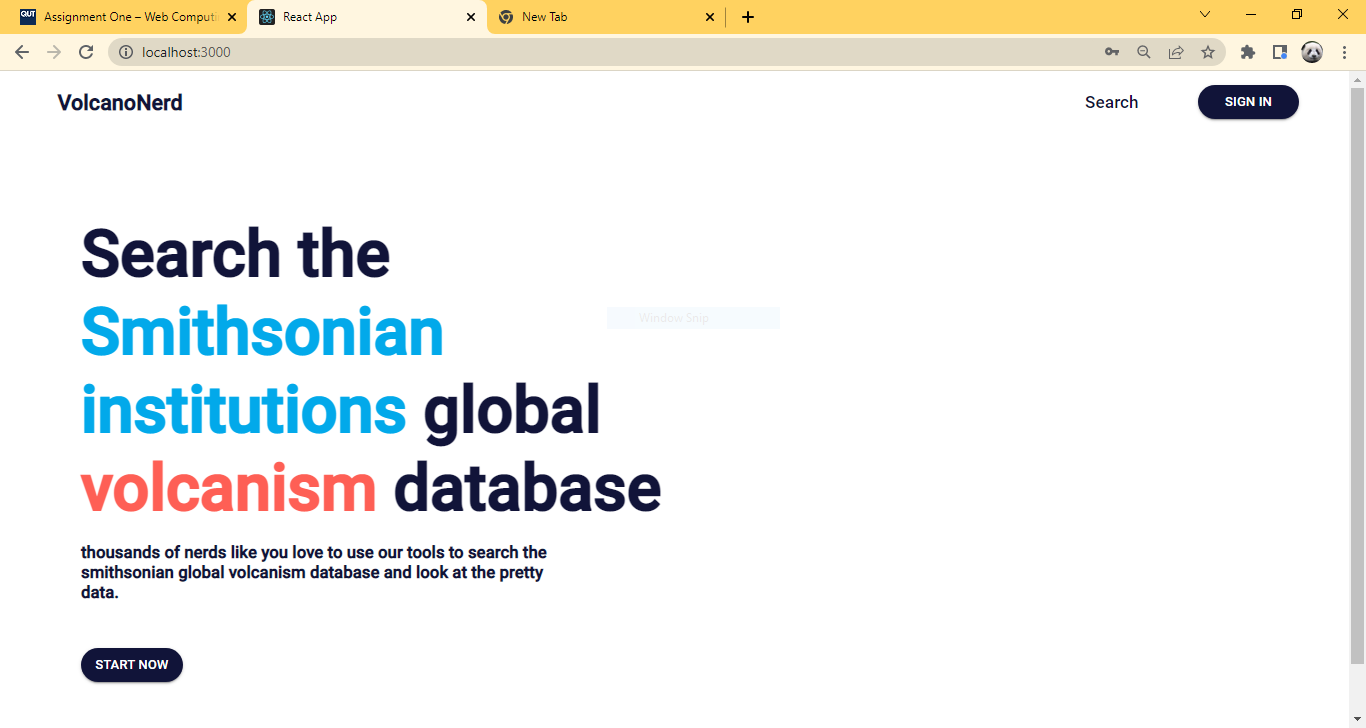
# Introduction

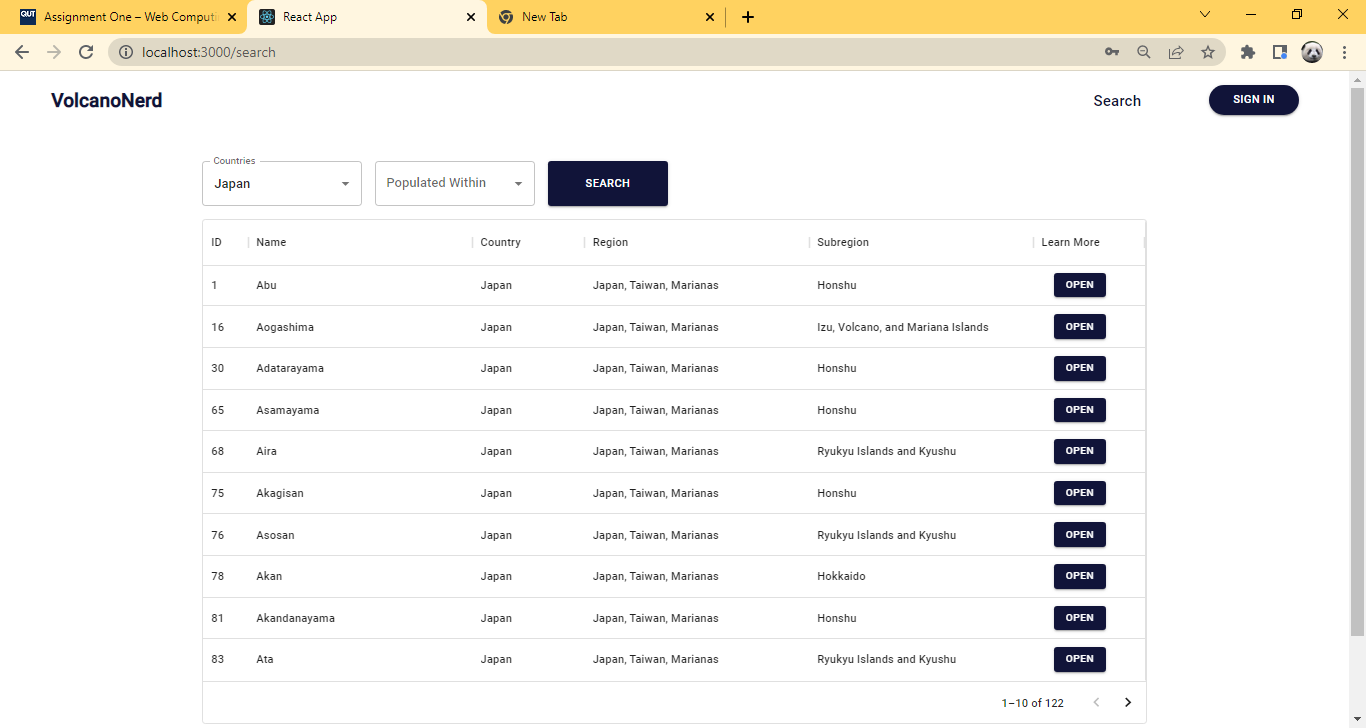
## Purpose & description

VolcanoNerd.com is a website designed to simplify accessing volcanic data for people who love volcanoes! You can search by country as well as population radius.

In my implementation of this assignment, I focused on design using the MUI framework to work with material design in react, using coolors.co for assistance with color selection.

For my CSS I elected to make use of CSS modules to make styles easier to deal with.





## Completeness and Limitations

During this assignment I implemented all the endpoints, utilised react router for navigation, made use of controlled inputs in my login and register form as well as implementing map and graphing components in my volcano page, one item which I failed to implement was I originally wished to have a hero element like the one on stripe.com with a tilt spanning the header and a moving colour background.

I was unable to implement this not due to lack of time but lack of skill or understanding.

# Use of End Points

### /countries

I fetch then use the output to populate a MUI autocomplete component with the list of countries supplied

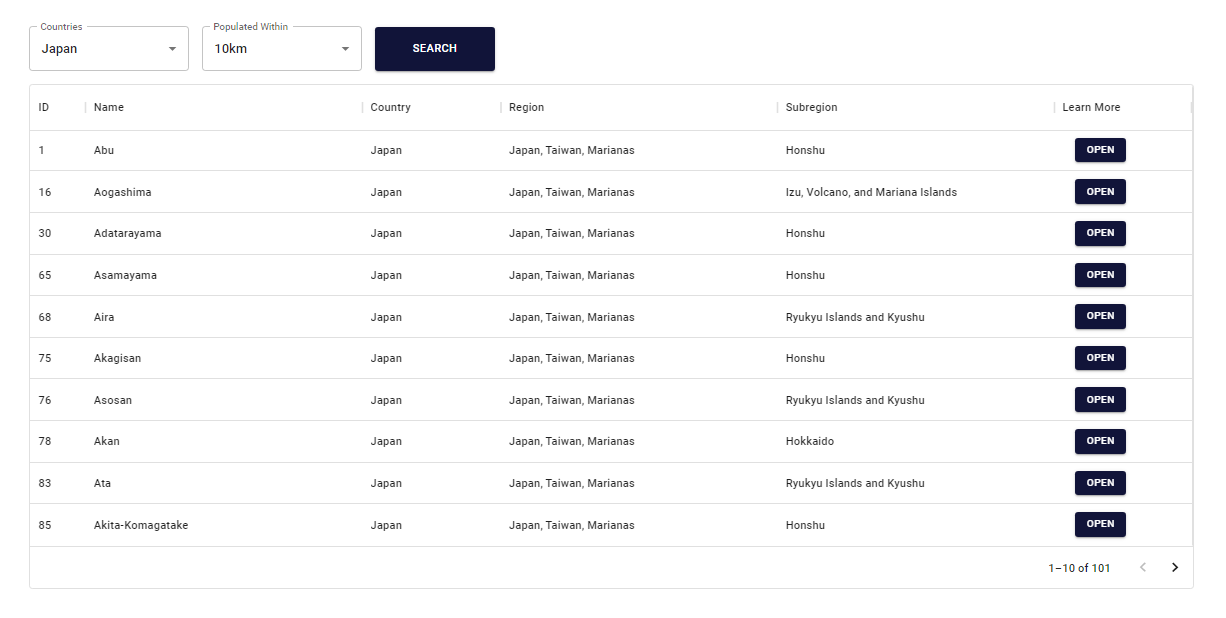
Graphical user interface, application

Description automatically generated

Originally, I kept a local copy of the JSON file rather than constantly making requests, but I thought that a new volcano may be discovered, and a new country added so I decided to refactor to this current approach.

### /volcanoes

A quirk of the fetch statement I use is that rather than implement separate fetch depending on whether user selects a populated distance or not I just pass an empty string in populated distance in the case of a lack of selection and the endpoint is smart enough to figure it out I am not sure if this is best practice or not as I am new to using fetch, but it works so I left it in.



### /volcano/{id}

I make use of pigeon maps to display the coordinates in a visual manner

I also have a graph available for signed in users.

One issue I encountered was how to use the fetch request based on whether a user was logged in, as I found that passing in a null value in the token parameter failed the entire request rather than simply provided the information available to unauthenticated users.

I ended up defining an options variable above the fetch that used a ternary operator to assign make it so an authorization header was provided if a JWT token was in session storage but not using that header in the opposite case.

const options =

*//if token is found, send authorization header otherwise send no authorization header*

        sessionStorage.getItem**(**"jwt"**)** != null

            ? {

**method:** "GET",

**headers:** {

**"Content-Type":** "application/json",

**Authorization:** `Bearer ${sessionStorage.getItem**(**"jwt"**)**}`,

                },

            }

            : {

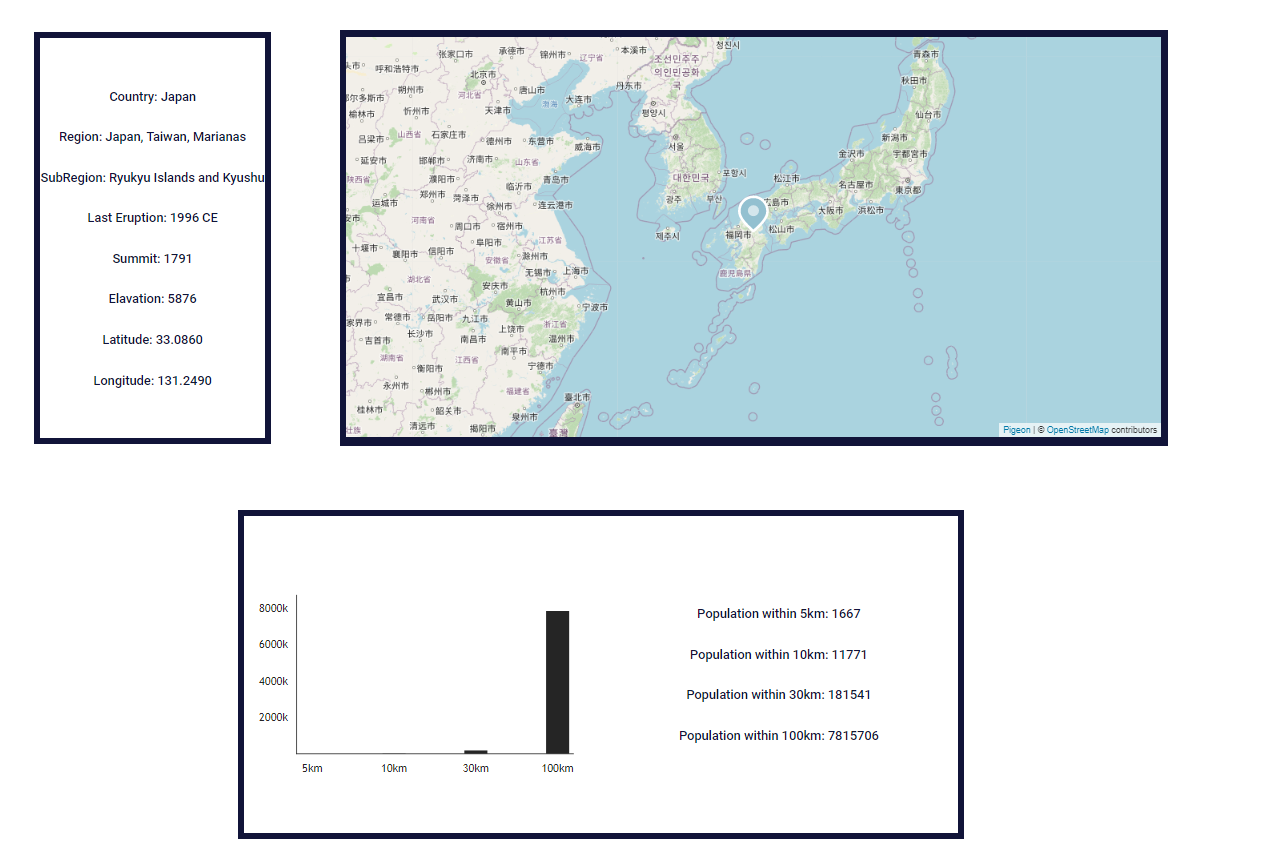
**method:** "GET",

**headers:** {

**"Content-Type":** "application/json",

                },

            };



### /user/register

Uses useState to create a controlled form based on MUI paper the fetch statement posts the username and password then I save the JWT token to session storage because I don’t know how to handle it the right way yet.

Once a user registers I automatically log him in.

Once a user is logged in the sign in button changes to a log out button, currently I am doing my logged in checks by seeing if the key “jwt” is null in session storage, but I think I should probably add state to track this instead.

Graphical user interface, application

Description automatically generated

### /user/login

The login form is quite like the register form the difference being rather than register the user and log him in we just log him in straight away. All that was stated above is true here as well.

I took inspiration from the google log in cards for the UI design of these cards

Graphical user interface, application

Description automatically generated

# Modules Used

## Material UI, MUI

A UI framework based on google material design, I use this for most of my UI elements including the cards and buttons

https://www.MUI.com

<https://www.npmjs.com/package/@mui/material>

## Victory Chart

A graphing library that I use to draw my population visualization on the volcano page.

<https://formidable.com/open-source/victory/>

<https://www.npmjs.com/package/victory-chart>

## Pigeon Maps

The map library I use to visualize the coordinates gathered from the endpoint.

<https://pigeon-maps.js.org/>

https://www.npmjs.com/package/pigeon-maps

# Application Design

## Navigation and Layout

Navigation is based around the navbar at the top of all pages, the design of the navbar took inspiration from the stripe.com navbar.

The only page with navigational nesting is the search page which allows users to go deeper and inspect more detailed information about an individual volcano

Letter

Description automatically generatedOriginally, I wanted the navbar to have a full color background and an icon of a volcano next to the logo which also acts as a link to the home page, I decided against these choices as I felt the choice I went with was more modern.



## Usability and Quality of Design

I tried to make design a focus on this project and drew heavily from inspirations such as stripe.com and google for my colors and overall design.

Originally, I had a demo of the volcano page complete with map and stats on the home page, but I did away with it in the final design as I felt it made the page cluttered. I also wanted an icon of a volcano on the navbar but did away with this as well.

A concern I had when working on the navbar was the lack of a specific “Home button” having decided to make the logo the home button, I was concerned users would be confused, having looked at other sites, I decided that users would know intuitively that they could navigate to the home page by pressing the logo due to their prior experiences.

Another major consideration was the volcanos page, I am still unsure as to whether I have taken the best approach to the design of this page, I attempted to make the clutter as minimal as possible and yet I still feel the design of this page strikes as somewhat dated.

As for colors and typography, I have pre-defined pallets and font families so this was never a large issue.

## Accessibility

* Provide a text equivalent for every non-text element – alternatives to images, symbols, scripts, graphical buttons, sounds, audio and video files and so on.

This is easy as the only image on the site is the map and this info is able to be accessed via the text information panel.

* Ensure that all information conveyed with color is also available without color, for example from context or markup.

All buttons are represented as thick and contained, all other elements are distinctly styled similarly.

MUI takes care of this for me luckily

* Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document.

To speak truthfully I have no idea how this would work since react code isn’t what shows up in devtools in the browser, and considering the fact I don’t know I highly doubt I have implemented this.

* Ensure that text equivalents are updated when dynamic content changes.

This is handled by state change I believe.

* Avoid causing the screen to flicker.

Screen does not flicker as far as I am aware

* Use the clearest and simplest language appropriate for a site's content.

I believe I am doing this

* For tables, identify row and column headers – clearly differentiated from the data.

This is done for us in MUI tables

# Technical Description

## Architecture

I’ve divided the source folder into 4 folders plus the index file, in alphabetical order these are

Components which contain the code which I have seen fit to abstract away from the pages they exist on.

These components are Bar chart, which I’m using to graph the population data.

Data Table holding the MUI data table that displays the results of calling the volcanoes endpoint

And Navbar which sits atop all pages by way of including it above all routes in the index file.

The next folder is Hooks, this folder only contains my useFetch hook, this is used to populate the page directly after load.

After this we have our routes, this holds all the pages for our application these are imported into the index file and passed into route components to be rendered by react-router.

Finally, we have styles, I have decided to use CSS modules in order to deal with my styles, I find this more pleasurable to work with as I am able to define styles on a page by page basis while also having global styles for things like typography and color. Each route and component have their own module.

Please see the following screenshot for a visual representation of the above information

Graphical user interface

Description automatically generated with medium confidence

## Difficulties / Exclusions / unresolved & persistent errors /

I had many difficulties while working with pigeon maps, the map will not load if provided null values in coordinates, I had to provide meaningless coordinates and update them when the fetch request returned.

This caused issues with the map popping from the wrong area to the correct one, I managed to solve this by creating logic to delay rendering of the page until the fetch comes back to us successfully.

Doing it this way solved the problem for the most part, but I still have some occasional pop in issues.

The second major issue I faced with pigeon maps is the hack that I’m using with the marker, rather than define the state of the marker uniquely I used just passed in the state of the centre of the map.

This meant I was unable to give users the ability to pan the map.

I had some issues with finding charting libraries that did what I needed, I tried graph.js and some other popular options but luckily, I was able to find victory graph.

An issue I have still yet to figure out is how to better visualise the population, as of now many volcanoes have such a small amount of people within 5 and 10 kilometres that these bars are too small to have any meaning since the 100km bar is so massive in comparison.

I have partially remedied this by providing the information in text format, but this is something I may wish to make clearer in the future.

# Extensions (Optional)

In the future I would consider adding other geographical features such as mountains or rivers or other items which may be catalogued in a similar way.

I also wanted to improve my management of tokens so that tokens could persist beyond sessions, unfortunately this is beyond my skull level at this time.

As I said earlier in the paper, I wised to have a curve on my home page like the one found on stripe.com, hopefully in the future I can elevate my skills to this level.

# User guide

The root directory is the home page, users can either click the get started button or go to the search tab in the nav bar.

From the search menu populate the country and populated within fields with your desired values and click search.

From here you can click open on any volcano entry in the grid to go to a page specific to that volcano providing detailed population data among other things.

If you wish to login or register, go to the sign in button in the top right, if you wish to make a new account click create account and enter your details, after you create a new user, you will be automatically logged in as that user.

If you just wish to sign in, just enter your registration details on the first sign in page.

Once signed in navigate back to volcano to see the full info.