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)

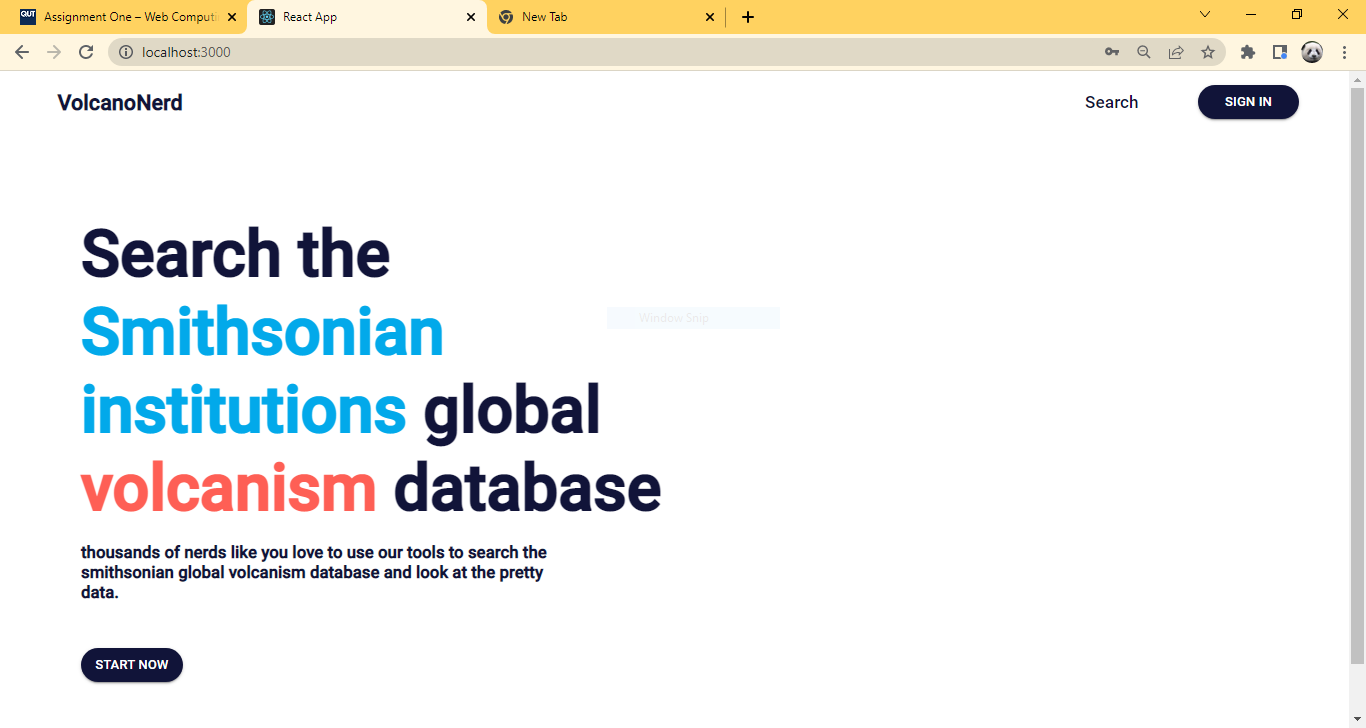
# Introduction

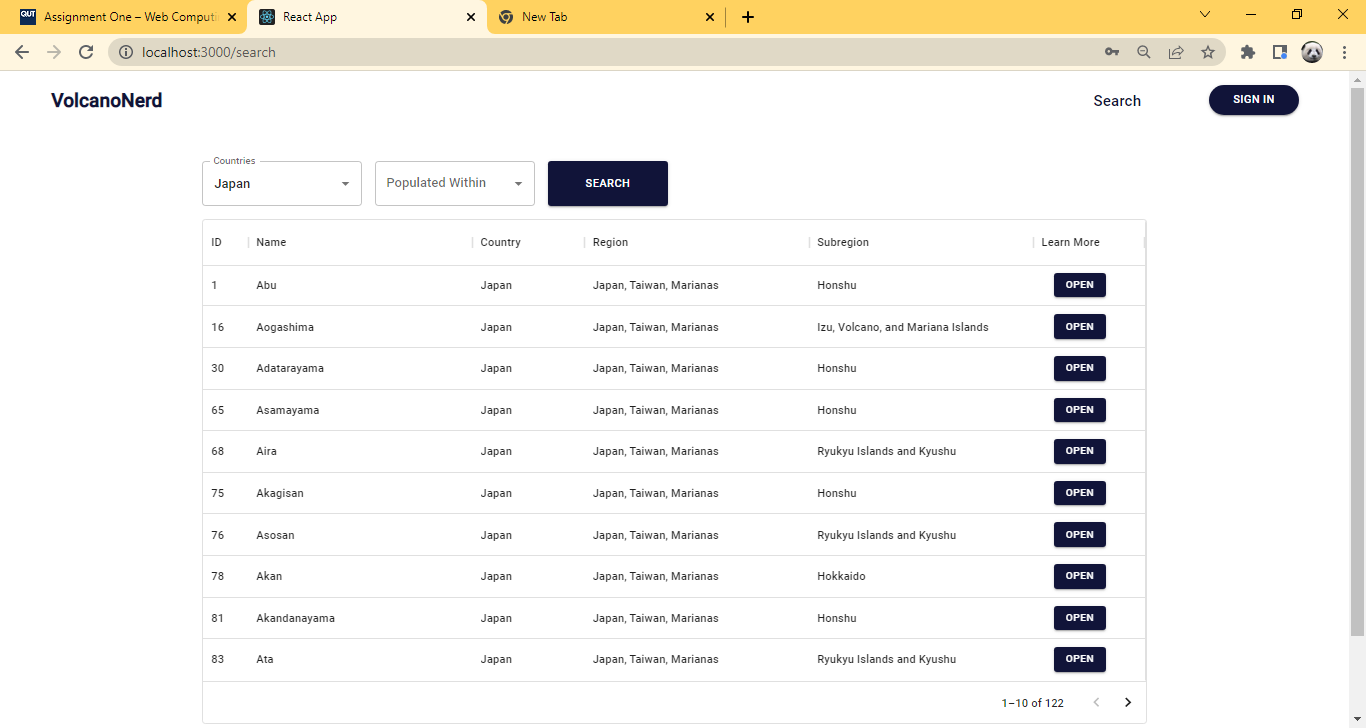
## Purpose & description

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 make use of my useFetch custom hook to query this endpoint, I then use the output to populate a MUI autocomplete component with the list of countries supplied

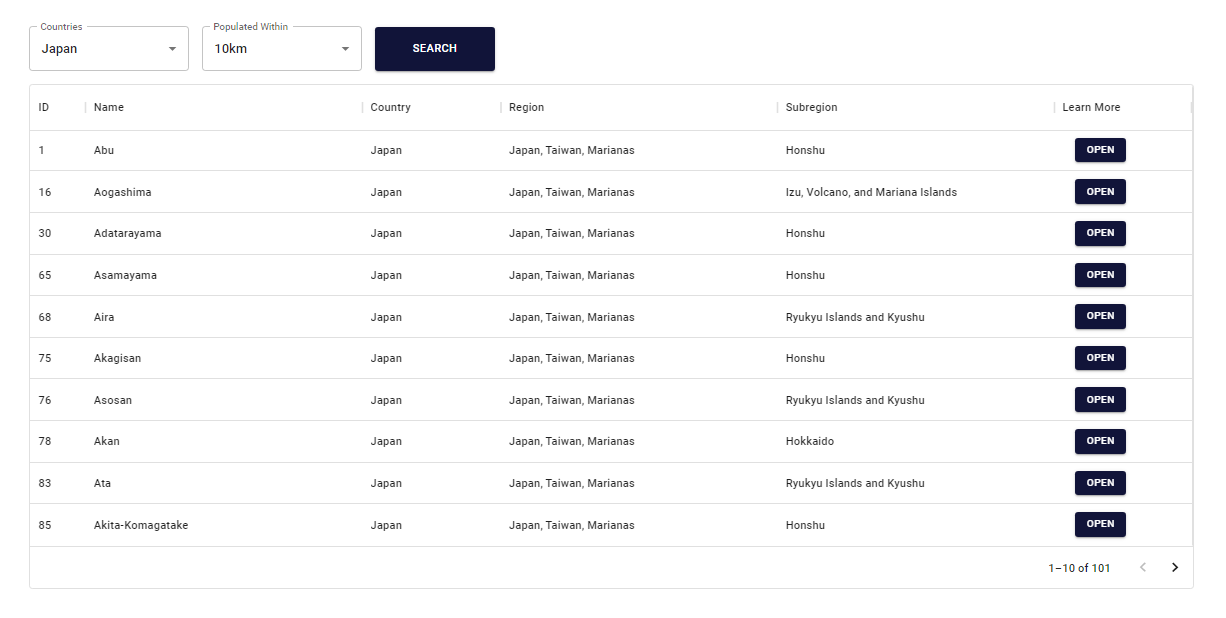
Graphical user interface, application

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 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"**)**}`,

                },

            }

            : {

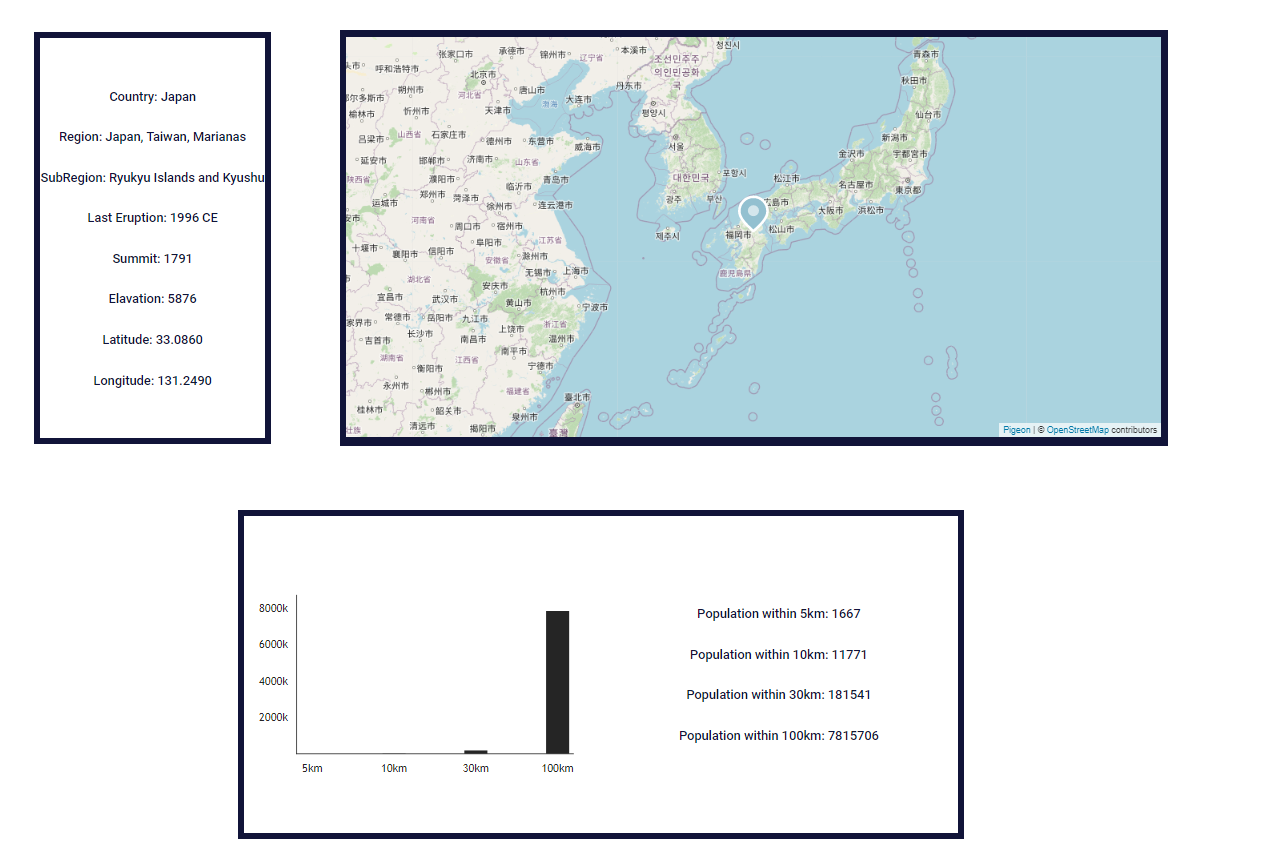
**method:** "GET",

**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

## Usability and Quality of Design

## Accessibility

# Technical Description

## Architecture

## Test plan

## Difficulties / Exclusions / unresolved & persistent errors /

# Extensions (Optional)

# User guide

# References