2024

CAB230 Assignment 2 Client Side

CAB230

Volcano API – Client-Side Application

Ngoc Anh Duong Nguyen

11479752

5/10/2024

Contents

Introduction 2
Purpose & description 2
Completeness and Limitations 3
Use of End Points 3
/countries 3
/volcanoes 4
/volcano/{id} 4
/user/register 5
/user/login 5
Modules Used 6
Ag-grid-react 6
React-select 6
React-toastify 6
pigeon-maps 6
React-chartjs-2 6
Bootstrap 6
Reactstrap 6
Application Design 7
Navigation and Layout 7
Usability and Quality of Design 8
Accessibility 9
Technical Description 10
Architecture 10
Test plan 11
Difficulties / Exclusions / unresolved & persistent errors 12
User guide 14
References 18
Appendices as you require them 20

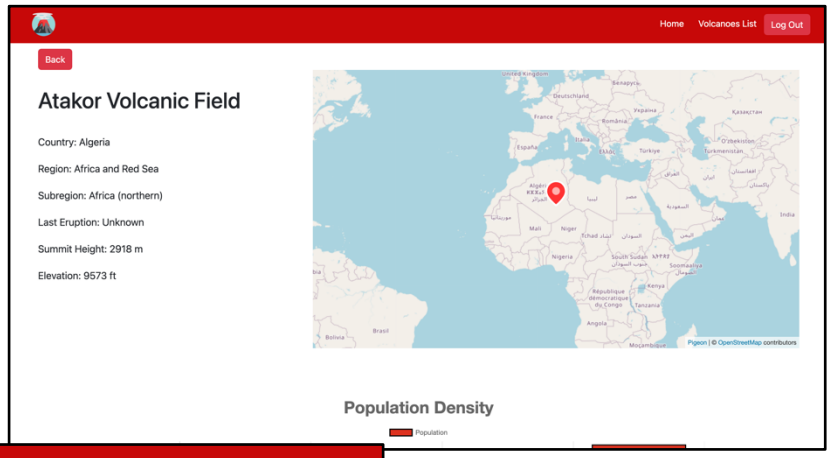
Introduction

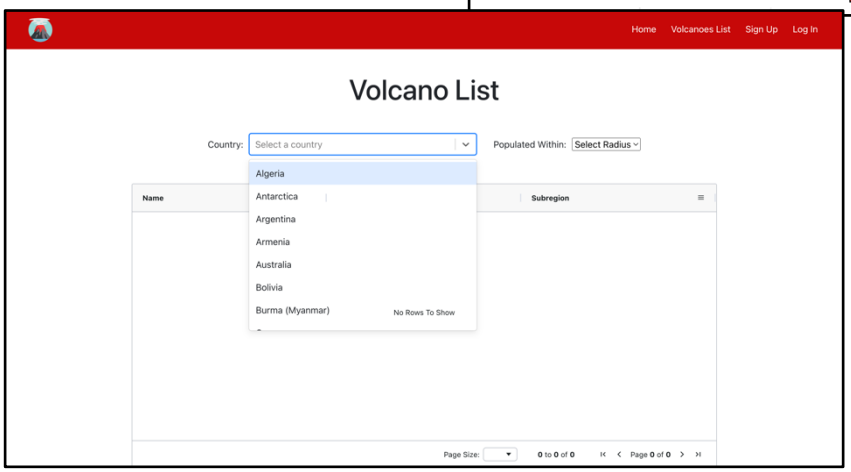
Purpose & description

For this assignment, I made a React-based web application that allows users to explore information about volcanoes in different countries. The application gets the information and facilitates user's request by querying to and from the "Volcanoes of the world" API. The retrieved information is then presented on web pages which are interactive, neat, and simple for an enhanced user experience. Users will be able to browse countries associated with volcanoes, list volcanoes within specific countries, and view individual volcano details. If the user is authenticated (signed up and logged in), they get the see population data around the volcano's radius.

In the development process of my web application, I aimed to differentiate my work by focusing on making the functionality robust and easy, straightforward user experience. For efficiency I made sure the app's navigation was robust. I used the React Router for handling navigation within the application, ensuring smooth transitions between different pages and enhancing the overall browsing experience. I utilised the AG Grid component to display the list of volcanoes in a sophisticated table format, allowing client-side interactivity. I also used Pigeon Maps to display the volcano's location with a marker. To visualise the population around the radius of the I used chart.js as its responsive and customisable.

For the country drop down on the volcano list, I used React-select as it allowed for a smoother, efficient, and more intuitive selection process. I implemented React-toastify to show pop up messages of fulfilled processes or errors. Finally, I used Bootstrap and Reactstrap to provide a consistent and modern design throughout the application. By using the pre-built components and styles, I was able to streamline the development process and ensure a polished appearance for the user interface.





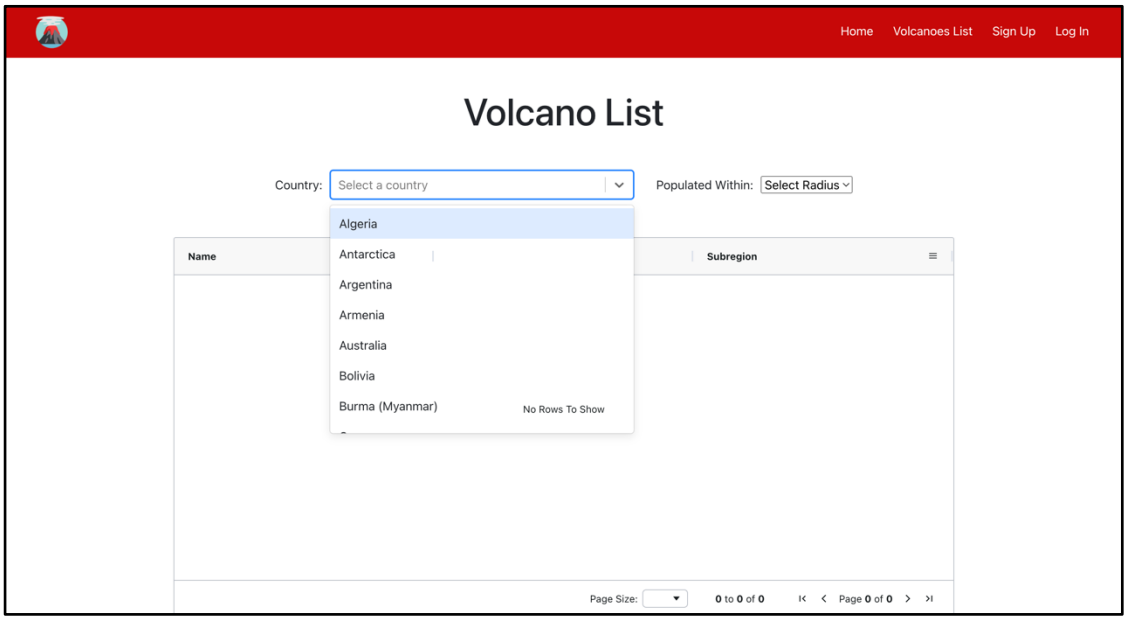
Completeness and Limitations

My application works smoothy and I've handled all the errors that could occur and any edge cases. I have developed the app to handle navigation using React Router to ensure smooth transitions between pages. I've implemented try-catch blocks for error handling when fetching data from the API to ensure the application doesn't break complete if an error occurs during the request (Hansa, 2022). I've also implemented controlled forms for the sign up and log in process, ensuring that the user's input is properly is handled, and only forms with email and passwords in the correct are sent to the API to reduce the number of errors. I've also successfully managed the JWT token's storage in my app, allowing the user to stay logged in even if the application tab is closed. I've handled the logging out process too, and implemented logging out if the user's token has expired. My chosen components, such as the list of countries in the dropdown, and the table for displaying volcano data, align closely with the data they represent, facilitating clear presentation and interaction. Additionally, I've also successfully integrated a map component to visually display volcano locations and a chart to illustrate population density data. Even though I have all the required functionality implemented, I feel that the design of my app is too simple, but that could be worked on if I had more time.

Use of End Points

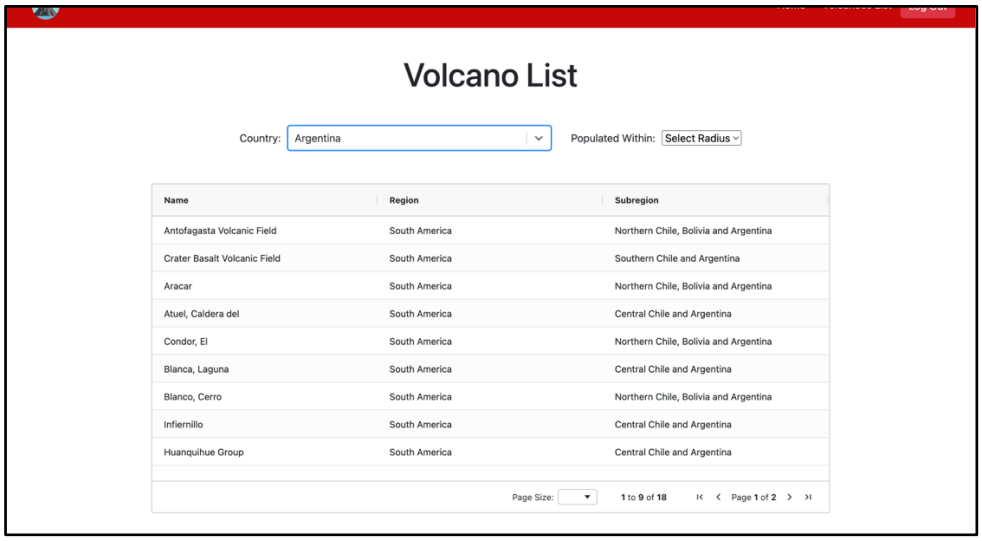
/countries

This endpoint retrieves a list of countries associated with volcanoes from the API. In my app, users can access this data by navigating to the "Volcano list" page, where a dropdown menu populated with country names allows them to select a country of interest. This selection then triggers the display of relevant volcano data associated with the chosen country.



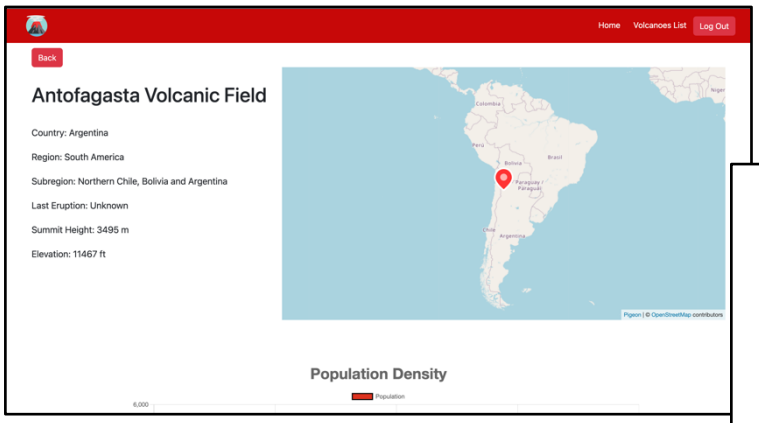
/volcanoes

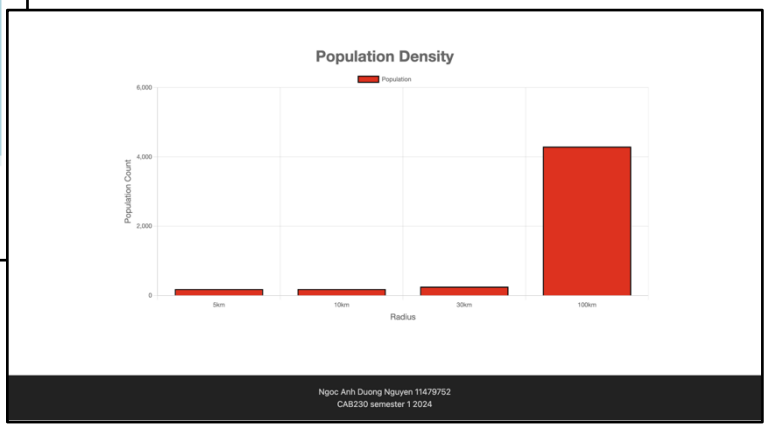
This endpoint returns a list of volcanoes based on the queried country. In my app, users can view this data on the "Volcano list" page's table after selecting a country from the dropdown menu. The name, region, and sub region data of each volcano to display each one in a row. When fetching, I also used the "PopulatedWithin" query to return a filtered list of volcanoes that have at least one person living within the provided radius. If the user is logged in, the user can choose the radius for filtering in the dropdown.



/volcano/{id}

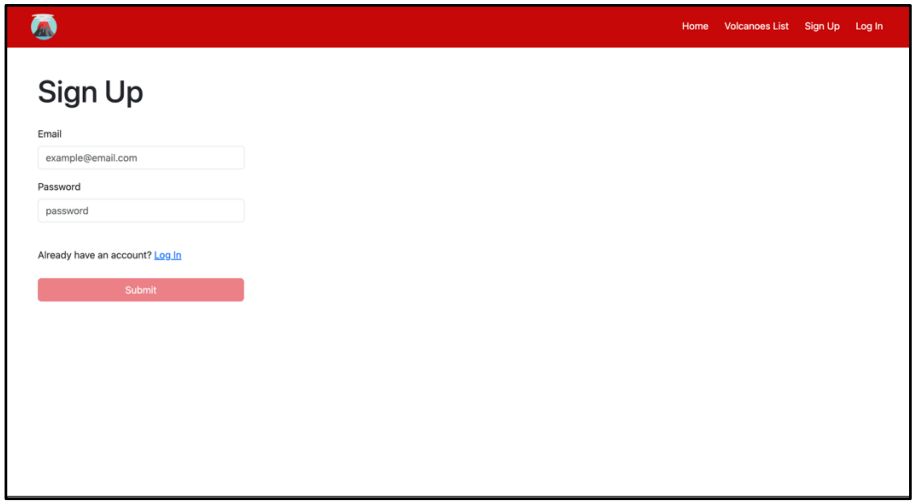
This endpoint provides detailed information about a specific volcano identified by its ID. In my app, users can access this data by clicking on a volcano name in the "Volcano List" table. This action navigates them to an individual volcano page where they can view comprehensive details about the selected volcano, including country, region, subregion, last eruption, summit height and elevation. The latitude and longitude of the volcano is taken to pinpoint its location on the map. If the user is logged in, the population with 5, 10, 30, 100km radius of the volcano is displayed in a chart.





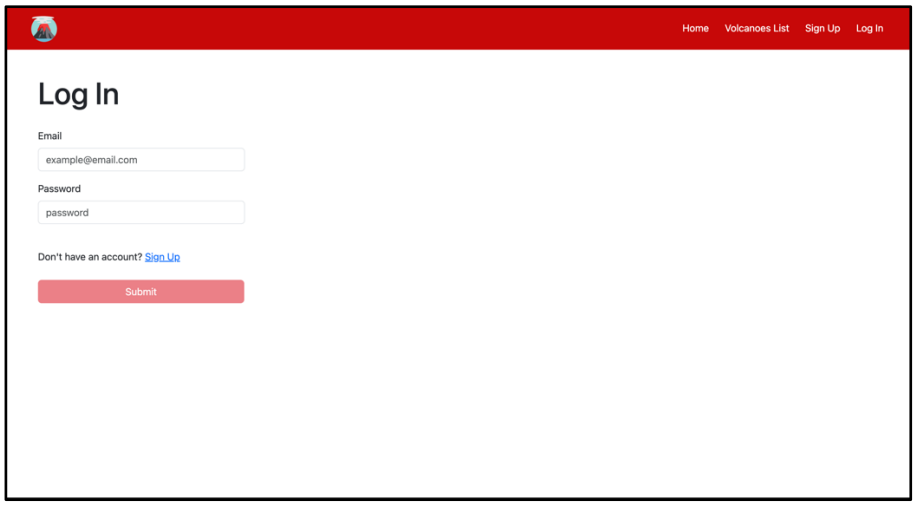
/user/register

This endpoint allows users to register for an account on the app. In my app, users can access the registration form by navigating to the "Sign Up" page. They are prompted to provide their email and password to create an account, enabling them to access authenticated features and data.



/user/login

This endpoint enables users to log in to their accounts. In my app, users can access the login form by navigating to the "Login" page. After entering their credentials, they gain access to authenticated content and additional functionalities within the app with the use of a JWT token.



Modules used

Ag-grid-react

This component is used for displaying and rendering data in a tabular format. It provides fully featured table components, including sorting filtering and pagination that users can interact with on the client-side (Koretskyi, 2021).
https://www.ag-grid.com/react-grid/

React-select

This module is a flexible and customisable dropdown component. It offers features like searching and clicking through the dropdown options, and its async feature is suitable for loading options from a remote source (Watson, n.d.).
https://react-select.com/home

React-toastify

This module is a simple and customisable toast notification. It allows developers to display success, warning, or error messages to users in a visually pleasing a non-intrusive manner (Khadra, 2023).
https://github.com/fkhadra/react-toastify

Pigeon-maps

Pigeon Maps is a lightweight and customizable mapping library. It provides a simple interface for displaying maps and markers, making it suitable for integrating maps into web applications with ease (Andra, 2024). Users can interact with the map by zooming and dragging.
https://github.com/mariusandra/pigeon-maps

React-chartjs-2

Chart.js is a flexible and easy-to-use charting library. It supports various chart types, including bar charts, line charts, and pie charts, and offers extensive customization options for creating interactive and visually appealing data visualisations (Yamanov, 2023).
https://github.com/reactchartjs/react-chartjs-2

Bootstrap

Bootstrap is a popular front-end framework for building responsive and mobile-first websites. It provides a comprehensive set of CSS and JavaScript components, making it easy to create modern and visually appealing user interfaces (W3Schools, 2019).
https://getbootstrap.com/

Reactstrap

Reactstrap is a set of Bootstrap components built for React applications. It allows developers to use Bootstrap's CSS and JavaScript components as React components and it ensures consistency in styles across React-based components and static elements of the website (Geeksforgeeks, 2023).
https://github.com/reactstrap/reactstrap

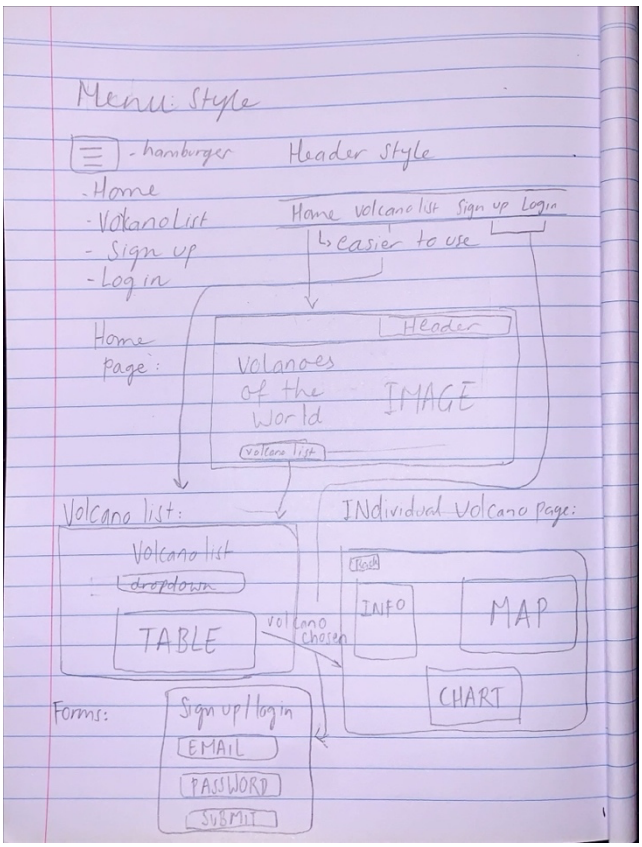
Application Design

Navigation and Layout

In designing the site, I aimed to create a user-friendly and intuitive experience that seamlessly guides users through accessing and interacting with volcano-related data. For navigation, I opted for a straightforward header menu structure consisting of essential menu items such as "Home," "Countries," "Volcanoes," "Register," and "Login". An alternative idea I had was to use a hamburger menu but didn't go through with it as it requires an additional click from the user which isn't ideal for interactivity (Mailchimp, n.d.). For the functionality of the navigation, I used React Router to make my website a single-page application. Using SPA helps improve the speed, responsiveness, and stability of the website (Microsoft, n.d.). Additionally, I incorporated a logical flow between screens, with each page providing clear pathways for users to navigate between different sections of the app.

When the application first loads, the user lands in the home page. The user can the choose to go to volcano list, sign up or log in page in the header, or they can go to the volcano list by clicking the prompted "volcano list" button in the hero. In the volcano list page, users can then delve into a specific volcano by clicking on its row in the table. The app then navigates them to an individual volcano page where details of the volcano are rendered based. When the user signs up, once they have successfully submitted the form, they're automatically directed to the log in page. Once the user logs in, the app reloads and navigates them to the home page, and this redirection happens on logout too. When the user logs out, the application tab reloads, takes the user back to the home page. If the user's token has expired and they try to access and individual volcano page, It will automatically log them out.

Rough sketching:



Usability and Quality of Design

When designing my web application, I wanted it to be simple, well-organised so that only relevant information needs to be communicated to the user, increasing their engagement and satisfaction (Fischer, 2022). Upon critical assessment I found both positive and negative aspects of my design.

For each page, I ensured each component is well organised, with a clean layout that displays the page in a well formatted manner. I made sure to not overload pages with too many texts or images, and spaced each component out so it doesn't look squished together. I made sure to use padding and margin, and align to display the components, and used auto margins to place things in the center so the placement is consistent on different resolution.

I made the navigation clear and intuitive by using a simple header with the menu items displayed on the top right corner so that users can easily access them at the top of the page. On the home page's hero, I added the "volcano list" button to prompt the user to click it if they wanted to access more information on volcanoes. When the user is in the individual volcano page, to go back to the volcano list page, the individual volcano page has a back button, so users don't have to use the browser's back button for quicker load. When the user logs in, the sign up and log in menu items are removed and a log out button is placed. When the user logs out, the menu items appear again.

The application also maintains consistency with user expectations from the other apps by adhering to common design patterns and conventions. I made sure the home page layout was the same as most websites, with the hero and image on top and other content after. I made sure the header stayed at the top of the page and footer stayed at the bottom. For the log in and log out pages, I followed the most common format to display the form for ease of use and navigation.

Using Bootstrap and Reactstrap helped me to keep the text sizing and colour scheme consistent. I used the same font for all the text, I made the headings in each page around the same size. I used a softer red colour scheme for the home, page header, chart, and buttons so there's no overload of bright colours that are harsh to the eyes. The toast notifications are made to be simple with short texts and it's colour coded so success notifications have a green tick mark and errors have a red X sign. This ensures the message is conveyed through colour as well as text. This consistency creates a unified visual identity for the app, enhancing overall aesthetics and user experience.

I found my volcano list, sign up and login page to be a little visually boring as there's a lot of whitespaces where I could've incorporated light colours or images. I could've placed the sign up and log in form in a box and put it in the middle to match with the alignment of the volcano list page. For the header, I could've highlighted or made the menu item change colour if the user is on the specified page for better interactivity.

Accessibility

Analysing my site from the perspective of accessibility, I've assessed its alignment with the lecture content and the W3C priority 1 Accessibility Requirements list:

My design ensures that all information conveyed with colour is also available without colour. All critical information is also presented through context or markup, ensuring that users who cannot perceive color can still access the content effectively (W3C, 2021).

My site includes a lot of dynamic content like the dropdowns, table, individual volcano page information, map, and chart, and I have ensured that the text equivalent changes when the content changes. When the user selects a country or radius from the dropdowns, the table updates the data quickly, and the correct volcano's data is rendered on the individual page (W3C, 2021).

I have taken care to avoid any elements or interactions on the site that could cause screen flickering, which may trigger seizures or discomfort for certain users (W3C, 2021). I made sure the shine effect of the heading in my home page moves slowly and the colour isn't bright, and the shadow effect when hovering over the menu items transitions slowly. I have also made my toast notifications appear for some time so users can read it.

For the table I made sure the column headers are clearly identified and the row information matches it. For the chart, I ensured that the bar graph had a clear heading, legend, and the x and y axis had labels so users can understand the data displayed. Additionally, when the user hovers over each bar, they are shown the exact number of the population for the specified radius.

I prioritized using concise language to ensure the content is easily understandable for all users (W3C, 2021). I also made sure the text were sized and spaced appropriately so it's not hard to read.

For the dropdowns, I used react-select as it allows the user to search for specific countries or scroll through the dropdown list to select. This is helpful because if the user had a country in mind, they don't have to scroll through the long list of countries. They could type it in the dropdown's search bar, and if the country is not on the list, it'll show there are no options based on the search. For the "populated within" radius dropdown, I used a simple click-to-choose one as there's only 4 options.

For the submit in the signup and login page form, I disabled the user from clicking it if they didn't enter a proper email and password and is only able to be clicked if the requirements are fulfilled. If the user doesn't type in an email with a proper format, a message that says "Please enter a valid email" pops up under the email input box to inform them.

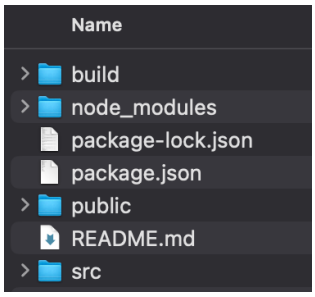
I have provided a non-text equivalent to my icon in the header (W3C, 2021), but I haven't done this for my home page hero image, the table, chart, and map. This could hinder accessibility for users relying on screen readers or those who are visually impaired.

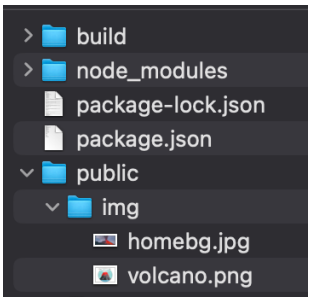
For the organisation of documents to be read without style sheets, the pages are still readable without CSS, but some elements are lost (W3C, 2021). All the text, menu items, icon, dropdown, and user forms are still functional, but the volcano list table doesn't display properly. Even though the user can still select the country from the dropdown, they aren't able to see the list of volcanoes and can't access the individual volcano page.

Technical Description

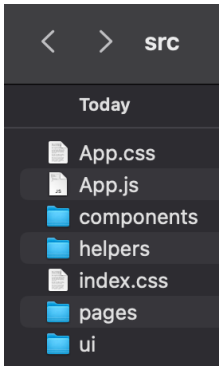
Architecture

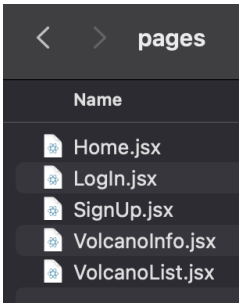
At a source code level, the architecture of my application follows a typical React application structure. There are the mandatory directories and file and folders for the application to run properly like build node_modules, package-lock.json, and package.json. In my public folder, I have another folder named "img" where I kept the images to be rendered in my app. Putting it in the public folder makes it render faster and accessible to all users (Allrounddiksha, 2023).

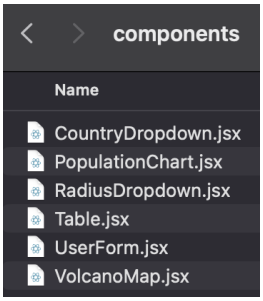


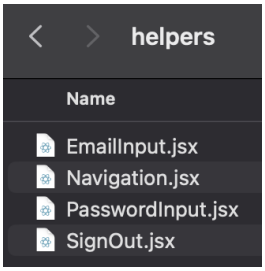


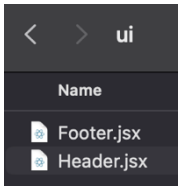
In my React app's src directory, I've structured the codebase to maintain a clear separation of responsibility and prioritised reusability. The main component is the App.js which contains the main structure and logic for the app's user interface (Kods, 2023).











I have split the JavaScript extension files into 4 different folders within the src directory: pages, components, utils, and ui. The pages folder contains all the page-level components representing different views or screens of the application For instance, Home.jsx, LogIn.jsx, SignUp.jsx, VolcanoInfo.jsx, and VolcanoList.jsx represent the home page, login page, signup page, individual volcano information page, and volcano list page, respectively. The component directory contains all the components used in specific pages, like dropdowns and table used in VolcanoList.jsx, the chart and map used in VolcanoInfo.jsx, and user form used in SignUp.jsx and LogIn.jsx. The helpers folder contains files that provide functionality to another specific file, like the navigation.jsx is used in header.jsx to handle the menu items, EmailInput.jsx and PasswordInput.jsx are imported in UserForm.jsx to help with the handling of the user's log in or sign up process. signOut.jsx does the logging out process when the user clicks to log out, or if their token has expired. The ui folder contains the header and footer which are elements that are consistently displayed on every page, and it is imported and used in the App.js.

Test

To test my application, I did manual testing for each task in the table below, and I have attached the screenshots for each test case in the appendix of this document.

Task	Expected Outcome	Result	Screenshot/s (Appendix A)
Click the "Volcano List" button to be redirected	"Volcano List" button on the home page redirects the user to the volcano list page when clicked.	PASS	1.1
Click the "Home" menu item in the header to go to the page	Application navigates the user to the home page	PASS	1.2
Click the "Volcano List" menu item in the header to go to the page	Application navigates the user to the Log In page	PASS	1.3
Click the "Sign Up" menu item in the header to go to the page	Application navigates the user to the Sign Up page	PASS	1.4
Click the "Log In" menu item in the header to go to the page	Application navigates the user to the Log In page	PASS	1.5
Search for countries by typing in the dropdown box's search bar	Countries matching the typed input should be displayed in the dropdown menu	PASS	2
Select for a country by selecting an option from the dropdown items	The list of volcanoes of that country should be displayed in the table	PASS	3
Select a radius from the "Populated Within" dropdown	The table should update to show the volcanoes that have population within that radius	PASS	4
Click on the volcano's name in the table to see more information	The application should direct the user to the individual volcano page and show the information of that volcano	PASS	5
Click the back button on the individual volcano page	The user is directed back to the volcano list page	PASS	6
Click on the volcano list page after device goes offline/API is down	An error toast notification pops up and the country list doesn't load	PASS	7
Click on the country in the dropdown after device goes offline/API is down	An error notification pops up	PASS	8
Click on the volcano's name in the table after device goes offline/API is down	App navigates to the individual volcano page, but nothing is rendered, and error toast notification pops up	PASS	9
Enter an incorrectly formatted email in the signup/login form	A message appears under the email input box to inform the users to enter a valid email	PASS	10
Enter an incorrectly formatted email and password in the signup/login form	The submit button stays disabled	PASS	11
Enter a correctly formatted email and password in the signup/login form	submit button becomes enabled so user can click it	PASS	12
Submit the sign up form with a new account	Success notification appears and user is redirected to the log in page	PASS	13
Submit the sign-up form with an account that's already signed up	Error notification appears	PASS	14
Submit the log in form with an account that's not been created	Error notification appears	PASS	15
Submit the log in form with the correct user details	Success notification appears and user is redirected to the home page	PASS	16

Submit the login form with the incorrect user details	Error notification appears	PASS	17
Submit the signup/login form while offline/ /API is down	Error notification appears	PASS	18
Click the log out button	Success notification appears and is redirected back to the home page	PASS	19
Click on the volcano name in the table when the user's login token has expired	Error toast notification appears, and user is logged out automatically	PASS	20
Go to the log in/ sign up page when user is logged in	User form does not load and only a message is displayed	PASS	21

Difficulties / Exclusions / unresolved & persistent errors /

Major roadblocks:

The most difficult part of developing the app was the logging in process, more specifically handling the token to keep the user logged in if they closed the app, and how they can access the authorised content when they're logged in. I was struggling to decide how to store the token in window.localstorage as it has an easy storage and retrieval process, and I can keep the user logged in if they close and reopen the app (Atkinson, 2023). Another issue I had was how do I fetch the authorised data when the user is logged in, and how to fetch data without the need for authorisation in my individual volcano page. Through searching on StackOverflow, I realised that I needed to get the token if the user is logged in and attach it to the fetch clause. If the user isn't logged in, the auhorisation process and token is ignored.

```
const fetchData = async () => {
  try {

    if (isLoggedIn) {
      const token = window.localStorage.getItem("token");
      headers = {
        Authorization: `Bearer ${token}`
      };
    }

    const response = await fetch(`http://4.237.58.241:3000/volcano/${id}`, {
      headers
    });
    const data = await response.json();

    if (isMounted) {
      setVolcano(data);
    }
```

Another major roadblock I encountered was during the development of the chart for the population density around the volcano. I had troubles making the title, legend, x, and y-axis labels appear in my bar graph, and when the user hovered over the graph it didn't show its exact population number. I searched on StackOverflow to realise that I was not rendering my chart properly. I opted to use React's useRef and useEffect hooks. The useRef hook was used to create a reference to the canvas element where the chart would be rendered. The useEffect hook allowed me to initialise the chart when the component mounts and update it whenever the populationData prop changes.

```
import React, { useRef, useEffect } from "react";
import Chart from 'chart.js/auto';

/* Display the population radius chart, some of the code has
been taken from chart.js */
export default function PopulationChart({ populationData }) {
  const chartContainer = useRef(null);

  useEffect(() => {
    if (!populationData || !chartContainer.current) return;

    const ctx = chartContainer.current.getContext("2d");

    const chart = new Chart(ctx, {
      type: 'bar',
      data: {
        labels: ["5km", "10km", "30km", "100km"],
        datasets: [
```

A minor issue I had was that my fetch functions were seemingly rendering the data twice. I found this out through logging the data in the functions to the console and the data would print twice. I found out that this was due to React's mount and unmount process, so to optimise my code I applied the react cleanup function. It basically uses useEffect to tidy up the code before the component unmounts (Innocent, 2024).

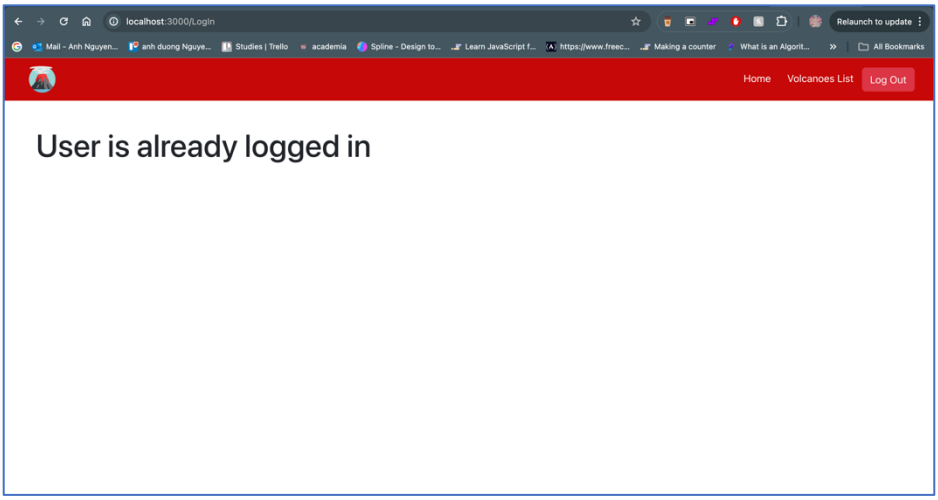
```
useEffect(() => {
  var isMounted = true;

  const fetchCountries = async () => {
    try {
      const response = await fetch(`http://4.237.58.241:30
      const data = await response.json();

      if (isMounted){
      setCountries(data.map(country => ({ value: country,

      }
    } catch (error) {
      console.error("Error fetching countries:", error);
      toast.error("an error has occured");
    }
  };

  fetchCountries();

  return() => {
    isMounted = false;
  };

}, []);
```

This variable is used to track whether the component is still mounted or not.

This conditional check ensures that the state is only updated if the component is still mounted.

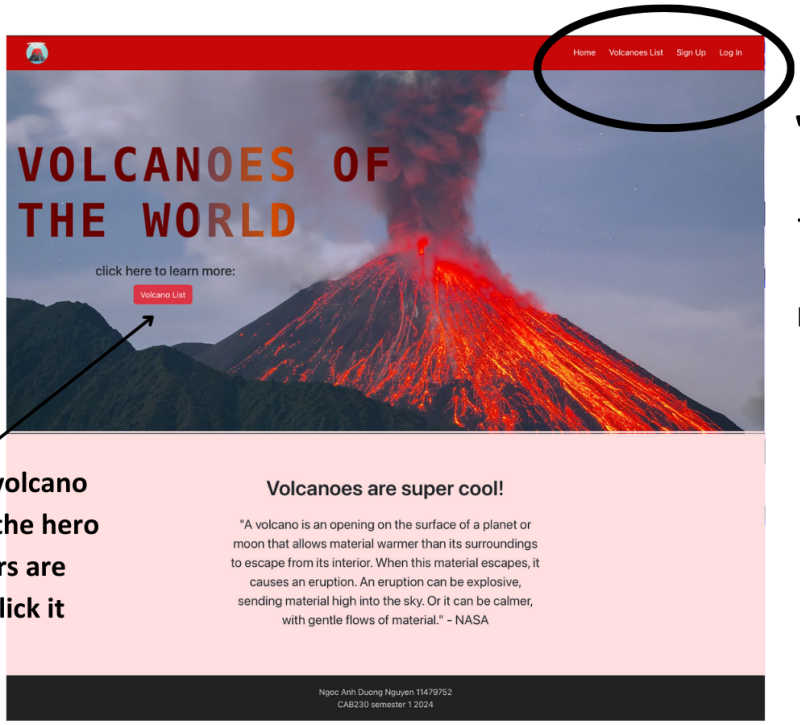
This is the cleanup function for the effect. It runs when the component is unmounted. It sets isMounted to false, indicating that the component is no longer mounted.

I successfully implemented all the functionalities required, but there is a minor functionality I tried to fix. The first one is when the user is logged in, they can, still access the log in or sign-up page by typing it in the browser. I tried to work around this by only letting the users access these pages if they're not logged in, and making them null if they're logged in. Trying to implement this code in some way broke my app, sometimes nothing would render or there'd be runtime errors I couldn't fix. As a work around to this, when the user enters the Log in and Sign Up page when they're logged in, the form won't load, and the page will appear as below.



User guide

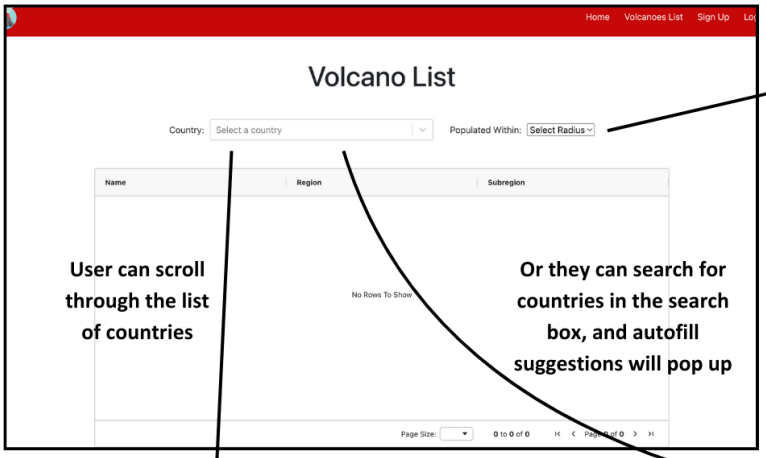
When the user first enters the app, they are directed to the home page.

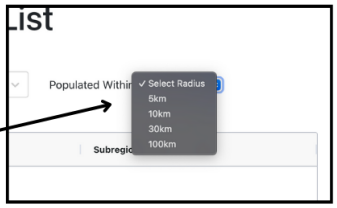


This header is where the menu items are placed and users use this to navigate the app

I placed the volcano list button in the hero so that users are prompted click it

When the user clicks on the Volcano List page, they will see the volcano list table and dropdowns

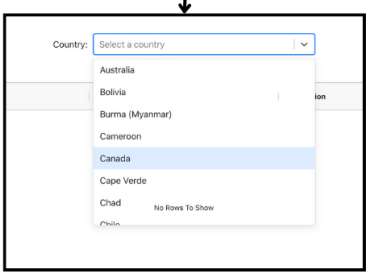


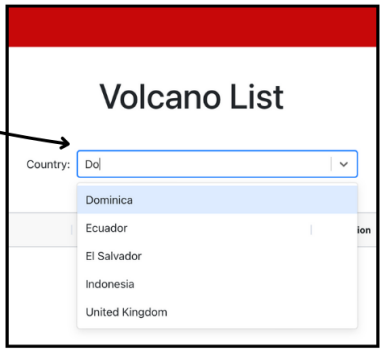


The user can select the radius around the volcano to see the list of volcanoes with population within that radius.

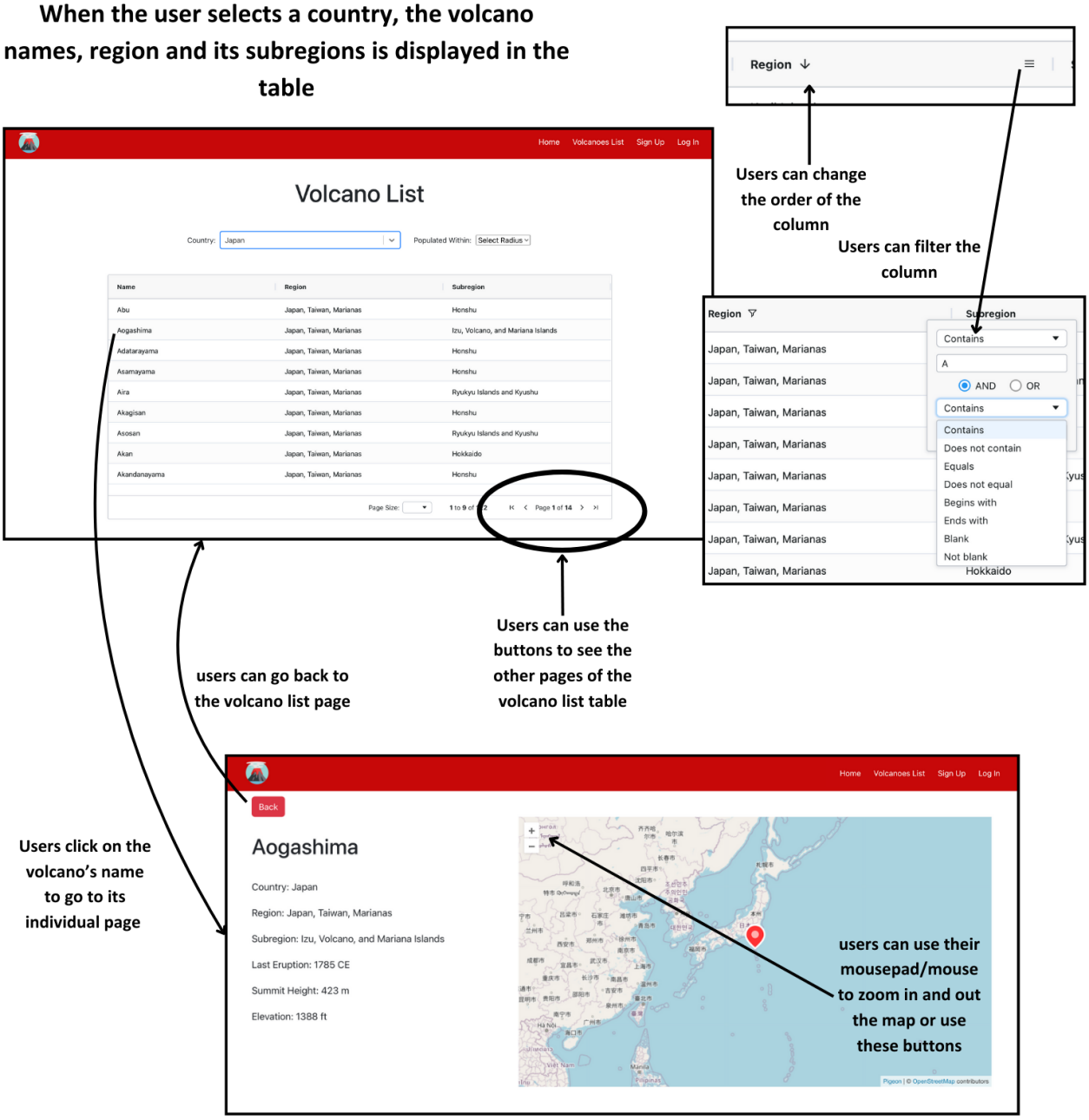
User can scroll through the list of countries

Or they can search for countries in the search box, and autofill suggestions will pop up

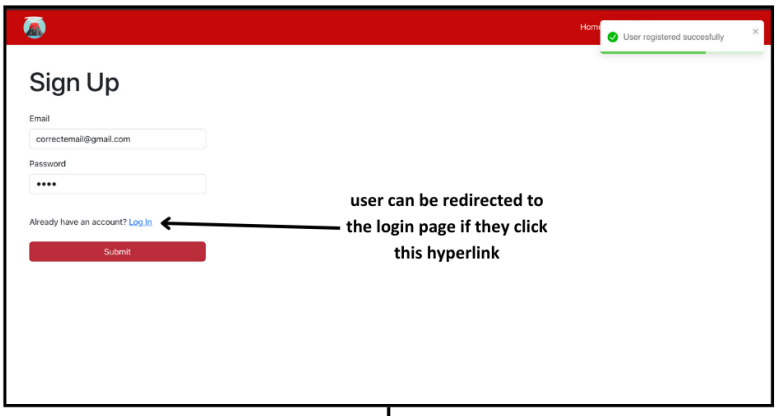




When the user selects a country, the volcano names, region and its subregions is displayed in the table

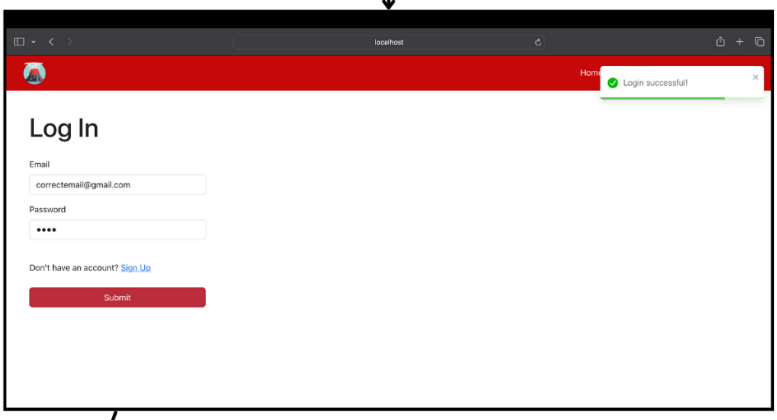


To now view authorised content, the user has to make an account, they have to go to the sign up page and fill out the form.



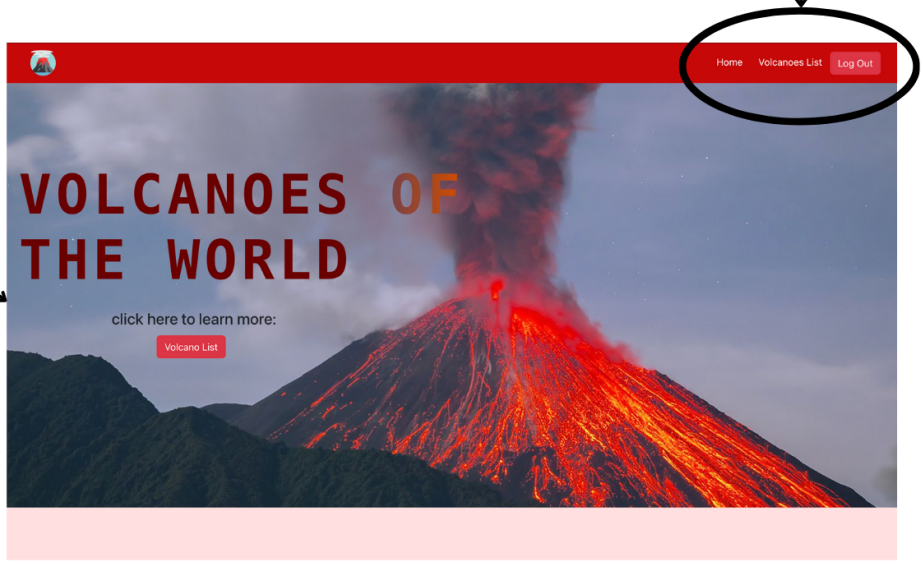
user can be redirected to the login page if they click this hyperlink

A few seconds after succesfully signing up, the user is redirected to the log in page to enter their account details

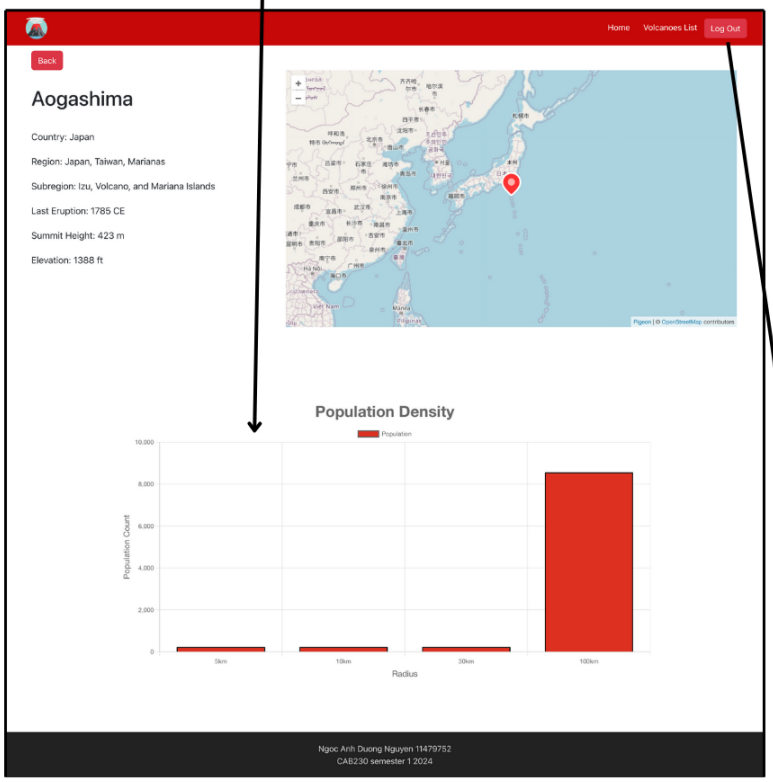


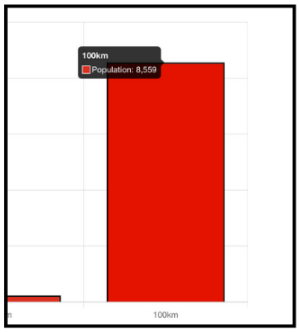
The sign up and log in menu items dissapeared and are replaced with a log out button

The user is redirected back to the home page after logging in



Now when the user goes to an individual volcano's page the population density bar graph is shown





Users can hover over the bars to see the exact population number

When the user clicks the log out button they're logged out and redirected to the home page

References

Allrounddiksha. (2023, July 6). *Where to Put Images in React*. Medium. https://medium.com/@allrounddiksha/where-to-put-images-in-react-a4e8f0aa270e

Andra, M. (2024, January 7). *Pigeon Maps - ReactJS maps without external dependencies*. GitHub. https://github.com/mariusandra/pigeon-maps

Atkinson, S. (2023, November 27). *The Best Place for Your JWTs - Comparing Local Storage and Cookies*. LinkedIn. https://www.linkedin.com/pulse/best-place-your-jwts-comparing-local-storage-cookies-atkinson-lerue/

Fischer, J. (2022, February 25). *The Key Elements of a Well-Structured Website (+ Checklist) - MailerLite*. Www.mailerlite.com. https://www.mailerlite.com/blog/key-elements-well-structured-website

Flaticon. (n.d.). Volcano [Online Image]. In *Flaticon.com*. Retrieved May 8, 2024, from https://www.flaticon.com/free-icon/volcano_1788740

Geeksforgeeks. (2023, September 27). *ReactJS Reactstrap*. GeeksforGeeks. https://www.geeksforgeeks.org/reactjs-reactstrap/

Hansa, U. (2022, May 23). *Implement error handling when using the Fetch API | Articles*. Web.dev. https://web.dev/articles/fetch-api-error-handling

Innocent, C. (2024, February 8). *Understanding React's useEffect cleanup function*. LogRocket Blog. https://blog.logrocket.com/understanding-react-useeffect-cleanup-function/

Khadra, F. (2023, August 12). *React-Toastify*. GitHub. https://github.com/fkhadra/react-toastify

Kods, Y. (2023, July 26). *Understanding the Key Files in a React App*. Nerd for Tech. https://medium.com/nerd-for-tech/understanding-the-key-files-in-a-react-app-1729391ce88b#:~:text=js%20%3A%20This%20is%20the%20main

Koretskyi, M. (2021, October 4). *Get started with AG Grid*. AG Grid. https://medium.com/ag-grid/get-started-with-angular-grid-in-5-minutes-83bbb14fac93

Mailchimp. (n.d.). *Traditional vs Hamburger Menus: Which is Right for Your Website?* Mailchimp. Retrieved May 4, 2024, from https://mailchimp.com/resources/hamburger-menu/#:~:text=Although%20the%20button%20still%20provides

Microsoft. (n.d.). *Choose between traditional web apps and single page apps*. Learn.microsoft.com. https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps

Read, M. (2016). Reventador Volcano Ecuador Erupting Night Red Stock Photo [Online Image]. In *Shutterstock.com*. https://www.shutterstock.com/image-photo/reventador-volcano-ecuador-erupting-night-red-482623717

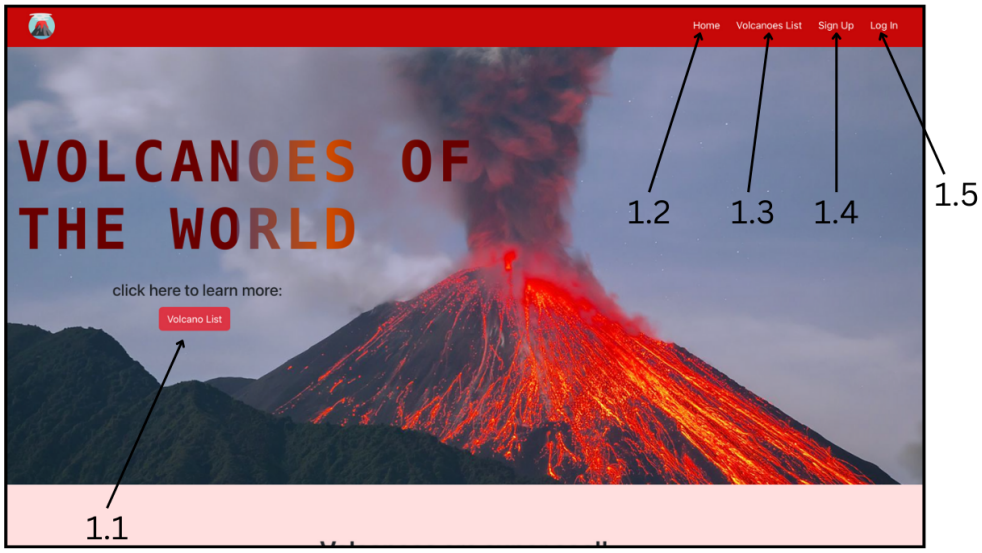
W3C. (2021). *Checklist of Checkpoints for Web Content Accessibility Guidelines 1.0*. W3.org. https://www.w3.org/TR/WAI-WEBCONTENT/full-checklist

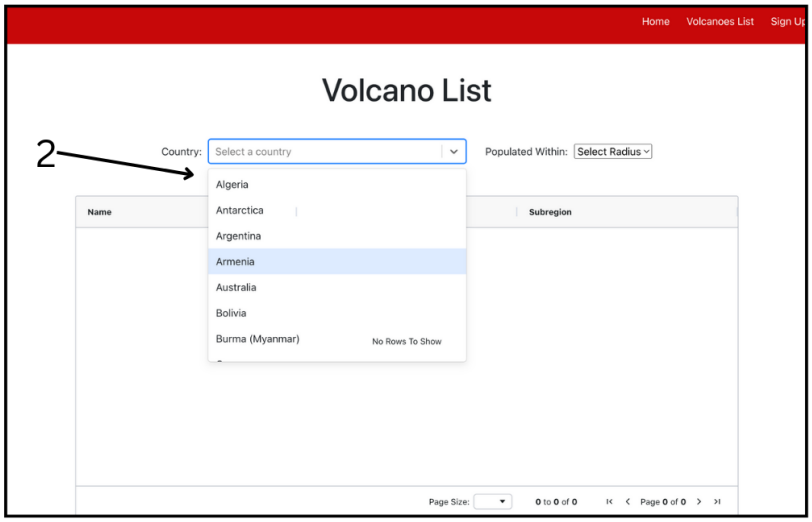
W3Schools. (2019). *What is Bootstrap*. W3schools.com. https://www.w3schools.com/whatis/whatis_bootstrap.asp

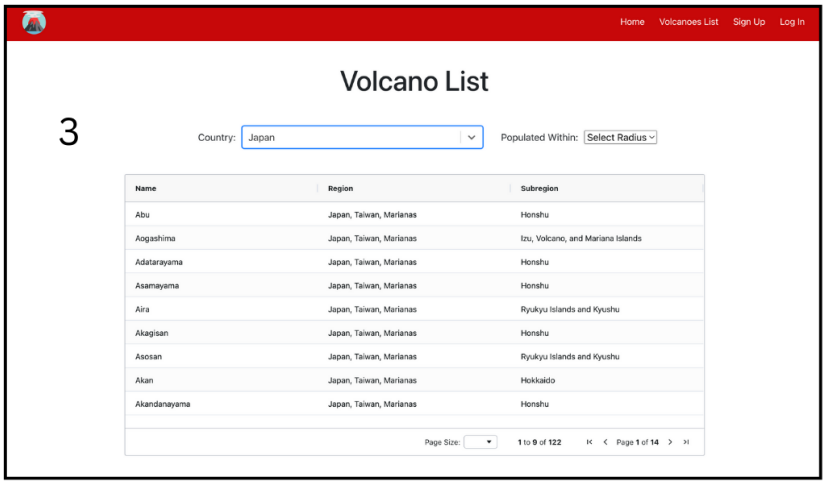
Watson, J. (n.d.). *React-Select*. React-Select. https://react-select.com/home

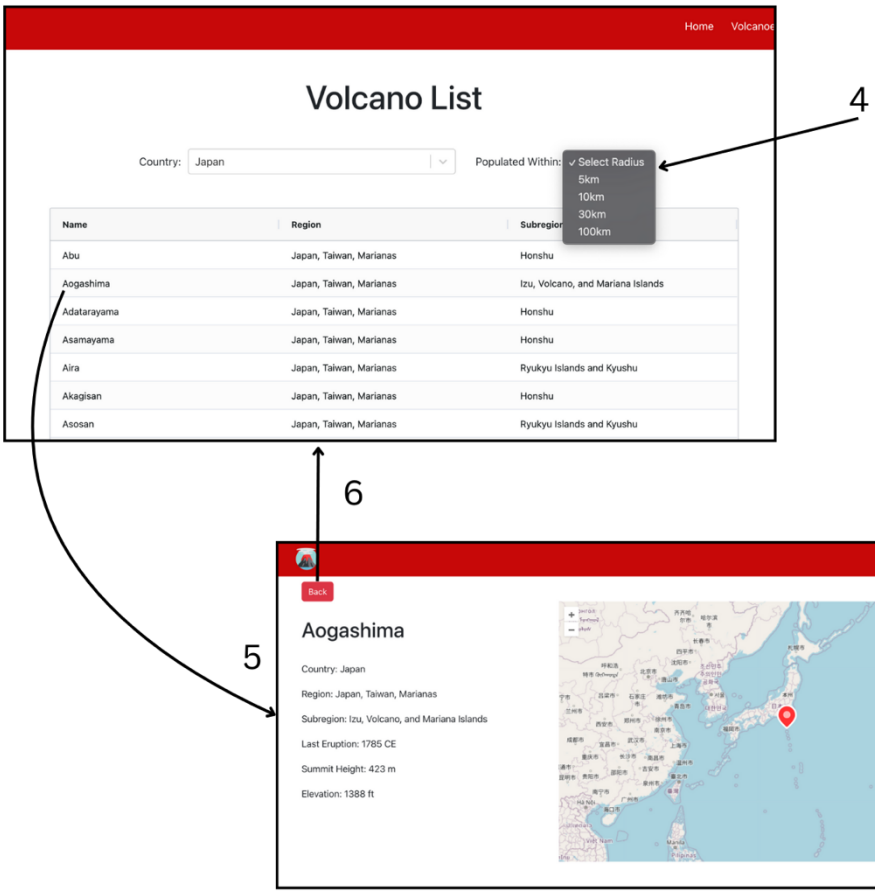
Yamanov, A. (2023, November 4). *reactchartjs/react-chartjs-2*. GitHub. https://github.com/reactchartjs/react-chartjs-2

Appendix A (test case screenshots)

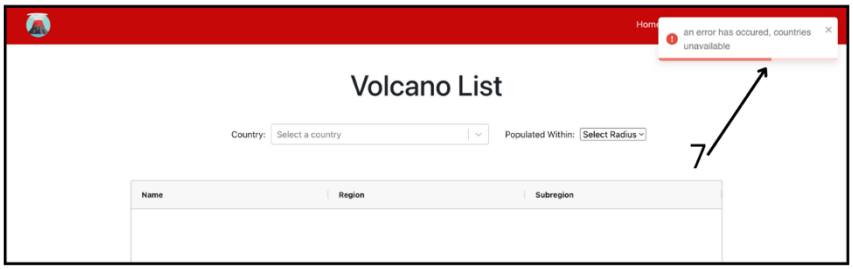


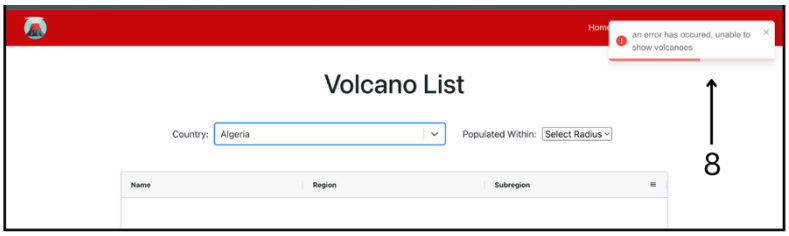





Home
Volcano List
Country: Japan
Populated Within: Select Radius
5km
10km
30km
100km
Name
Region
Subregion
Abu
Japan, Taiwan, Marianas
Honshu
Aogashima
Japan, Taiwan, Marianas
Izu, Volcano, and Mariana Islands
Adatarayama
Japan, Taiwan, Marianas
Honshu
Asamayama
Japan, Taiwan, Marianas
Honshu
Aira
Japan, Taiwan, Marianas
Ryukyu Islands and Kyushu
Akagisan
Japan, Taiwan, Marianas
Honshu
Asosan
Japan, Taiwan, Marianas
Ryukyu Islands and Kyushu
4
6
5

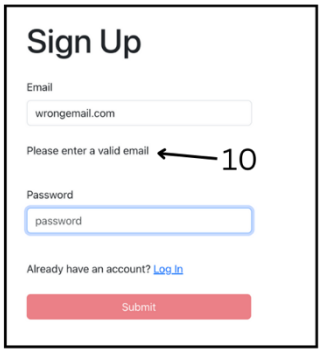
Back
Home
Volcanoes List
Log Out
Aogashima
Country: Japan
Region: Japan, Taiwan, Marianas
Subregion: Izu, Volcano, and Mariana Islands
Last Eruption: 1785 CE
Summit Height: 423 m
Elevation: 1388 ft

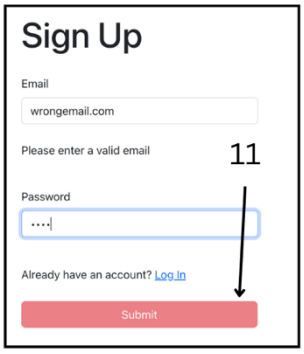

an error has occured, countries unavailable
Volcano List
Country: Select a country
Populated Within: Select Radius
Name
Region
Subregion
7

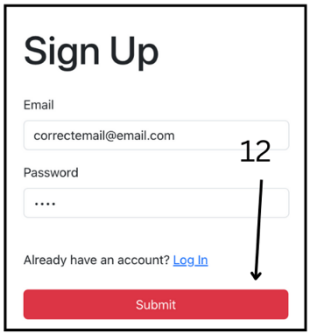

an error has occured, unable to show volcanoes
Volcano List
Country: Algeria
Populated Within: Select Radius
Name
Region
Subregion
8

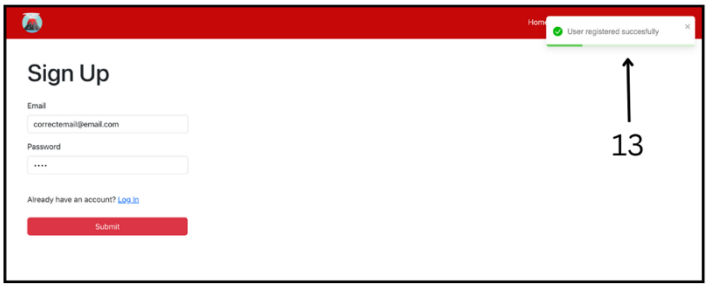

Back
an error has occured, unable to show volcano information
9

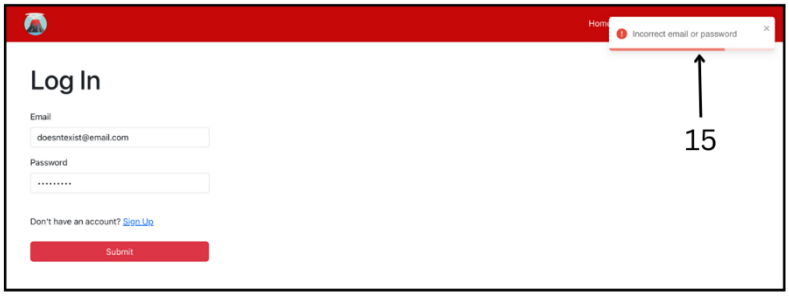
all 3 screenshots apply to the form in the login page (10-12)

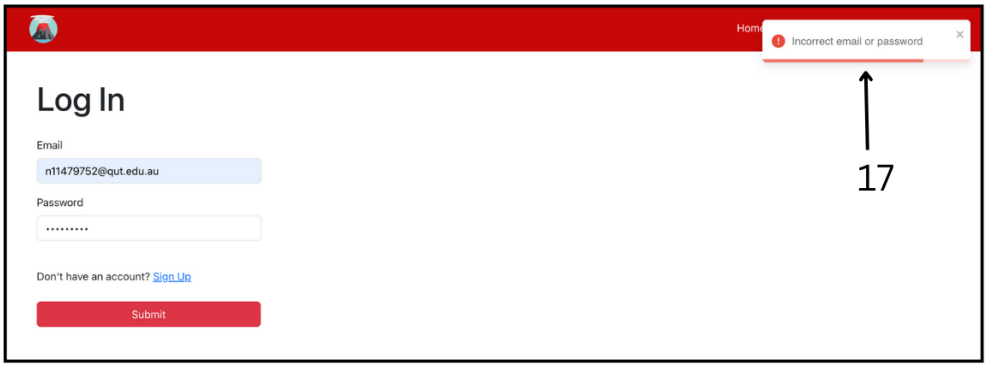


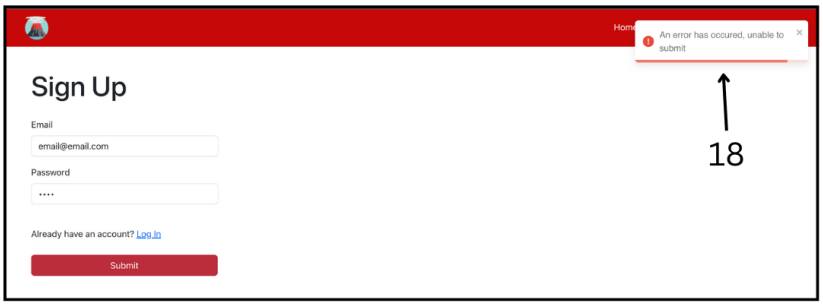


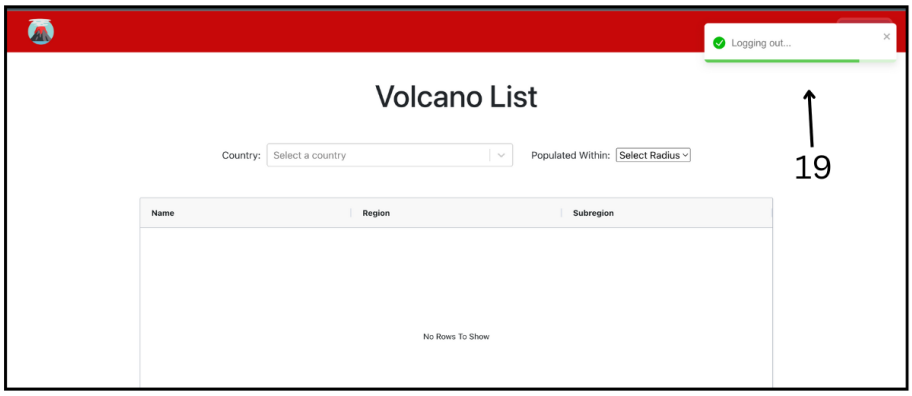


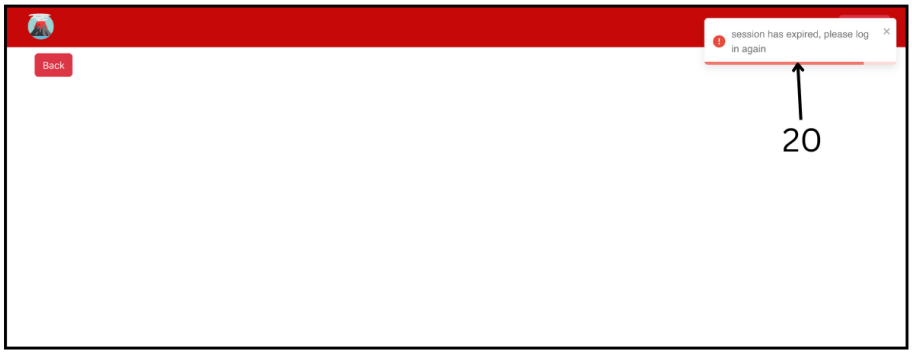




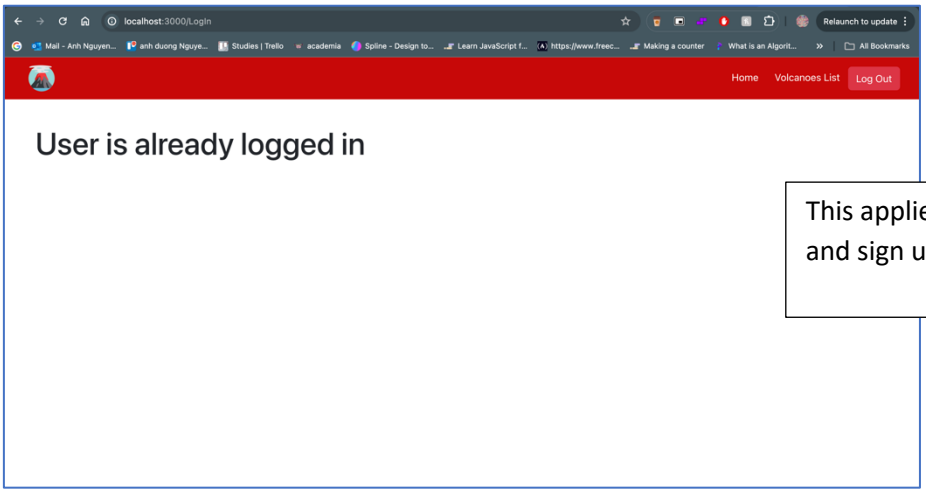

Incorrect email or password
Log In
Email
n11479752@qut.edu.au
Password
Don't have an account? Sign Up
Submit
17


An error has occured, unable to submit
Sign Up
Email
email@email.com
Password
Already have an account? Log In
Submit
18


Logging out...
Volcano List
Country:
Select a country
Populated Within:
Select Radius
Name
Region
Subregion
No Rows To Show
19


Back
session has expired, please log in again
20

21



This applies to the log in and sign up page