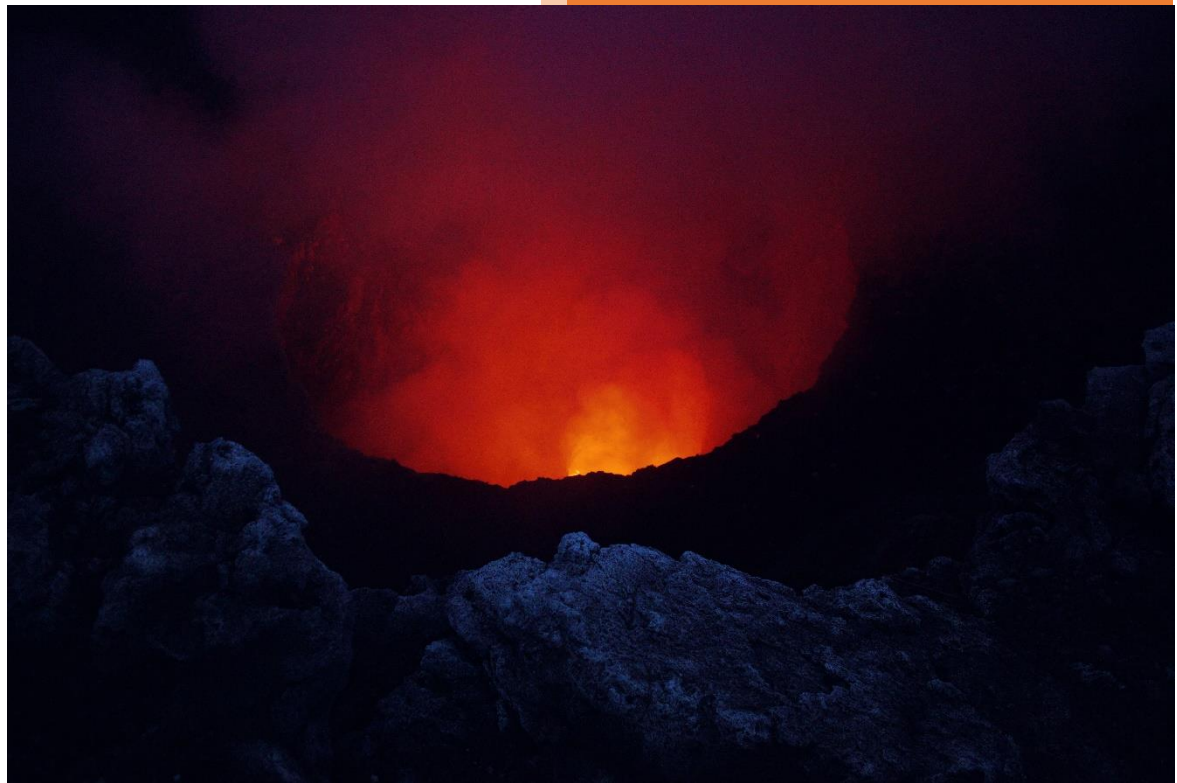


2022

CAB230 Assignment 1 Client Side



CAB230

Volcano API: Client-Side Application

Harrison Leach

N11039639

5/18/2022

Contents

Introduction	2
Purpose & description.....	2
Completeness and Limitations.....	3
Use of End Points	3
/countries.....	3
/volcanoes.....	3
/volcano/{id}	3
/user/register	4
/user/login	4
Modules Used	4
Ag-grid-react	4
Reactstrap	4
React-chartjs-2	4
Pigeon-maps	4
Application Design	5
Navigation and Layout	5
Usability and Quality of Design	5
Accessibility.....	6
Technical Description.....	6
Architecture	6
Test plan.....	7
Difficulties / Exclusions / Unresolved & Persistent Errors	7
Extensions	8
User guide	8
References	12
Appendix	12

Introduction

Purpose & description

This report details the volcano web application that was developed using Facebook's *React Environment*, a JavaScript library that assists in web development. The application enables the user to choose a country and a radius (in km). The country then displays a list of volcanos within the intended region. The user can select a volcano from this list which will navigate them to an individual volcano page showing information specific to that volcano. Extra information such as the population density surrounding the volcano is available for users that have registered and are logged in. All information was available through a provided REST API.

Within the application, some features that go beyond the scope of the main task were implemented. In the volcano list page, instead of using a search bar to input a country, a table using ag grid, was displayed. A specific country could be chosen within this table and upon clicking the search button, the list of volcanoes would be acquired.

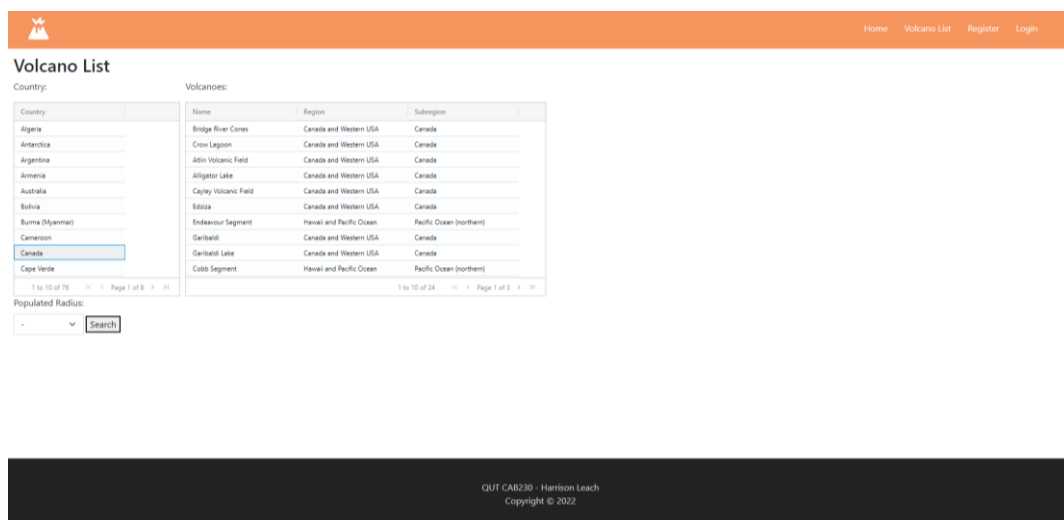


Figure 1: Volcano List Page

In the registration page, two password fields are included to confirm the password is what the user has intended it to be. If fields do not equal each other, an error will be prompted to the user to check passwords and enter once more.

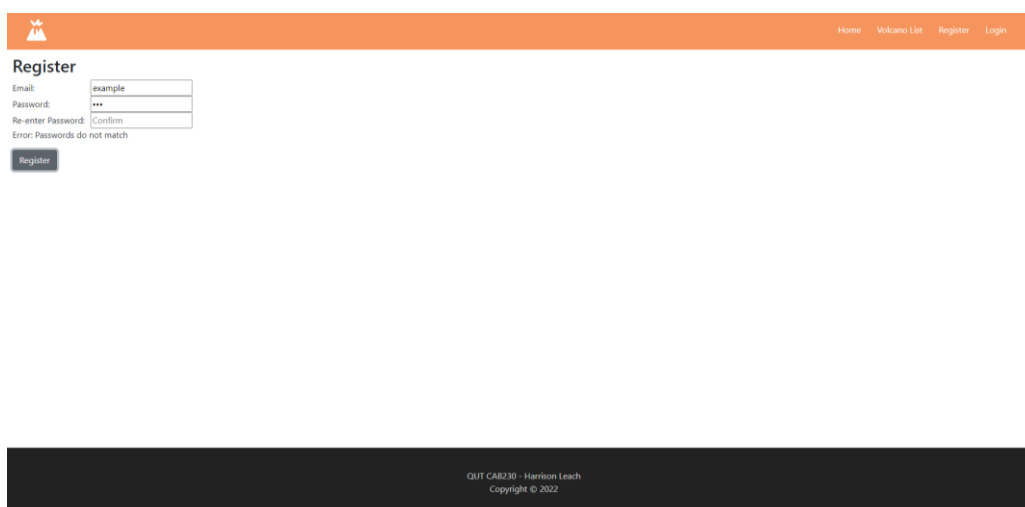


Figure 2: Registration Page

Completeness and Limitations

Regarding functionality, the web application encompasses all given endpoints for their intended purpose. Login and registration work well and include error handling to assist the user if a mistake is to occur. With the use of authentication, the individual volcano page is responsive to the user's login status. Extra functionality of changing the navigation bar to allow the user to logout was attempted. Although this function does work, there are issues, which along with the endpoints will be discussed later in the report.

Table, map, and chart components were included to display lists, location, and population density respectively. The table, with the assistance of Ag grid, has filtration and sorting methods for users to be specific with their searches. Although further styling could've been included, there is a general structure the react app follows and avoids feeling cluttered. The website also takes the form of a single page application (SPA) that is navigated using react router.

Use of End Points

</countries>

The `/countries` end point is vital to the searching process of the volcano. The JSON is mapped through and then displayed on the Ag grid table, enabling users to select a country.

</volcanoes>

Depending on what country has been selected, it becomes the parameter for the `/volcanoes` endpoint. This endpoint is used to obtain the list of volcanoes associated with the country in question. The query parameter, `populatedWithin` is given via dropdown beneath the country list for further filtration, it is not required.

Volcano List

Country: Volcanoes:

Country	Name	Region	Subregion
Algeria	Bridge River Cones	Canada and Western USA	Canada
Antarctica	Crow Lagoon	Canada and Western USA	Canada
Argentina	Atlin Volcanic Field	Canada and Western USA	Canada
Armenia	Alligator Lake	Canada and Western USA	Canada
Australia	Cayley Volcanic Field	Canada and Western USA	Canada
Bolivia	Edizae	Canada and Western USA	Canada
Burma (Myanmar)	Endeavour Segment	Hawaii and Pacific Ocean	Pacific Ocean (northern)
Cameroon	Garibaldi	Canada and Western USA	Canada
Canada	Garibaldi Lake	Canada and Western USA	Canada
Cape Verde	Cobb Segment	Hawaii and Pacific Ocean	Pacific Ocean (northern)

1 to 10 of 76 < > Page 1 of 8 >

Populated Radius: Search

Figure 3: Countries & Volcanoes Endpoints in Application

</volcano/{id}>

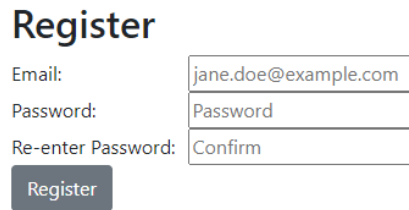
The `/volcano/{id}` endpoint was used to retrieve the information that is displayed on the individual page. In addition, if the user is logged in, population density will be returned as well, otherwise a message is left encouraging the user to login.



Figure 4: Individual Volcano Page

/user/register

The `/user/register` endpoint is a POST method that takes an email and password that, given the email is not already taken, creates a new user.

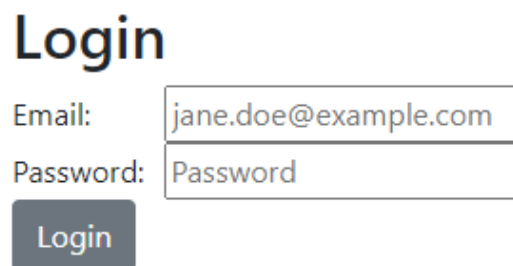


The registration form is titled "Register". It contains three input fields: "Email:" with the value "jane.doe@example.com", "Password:" with the value "Password", and "Re-enter Password:" with the value "Confirm". Below these fields is a dark grey button labeled "Register".

Figure 5: Registration Forms

/user/login

`/user/login` is a vital endpoint as it provides authentication to the app. If the given email and associated password match a registered user, a token is returned. This token can be used to give access to authenticated users only.



The login form is titled "Login". It contains two input fields: "Email:" with the value "jane.doe@example.com" and "Password:" with the value "Password". Below these fields is a dark grey button labeled "Login".

Figure 6: Login Forms

Modules Used

Ag-grid-react

Module to provide fully featured table components, including sorting and filtering.

<https://www.ag-grid.com/react-grid/>

Reactstrap

Module that uses bootstrap inspired components for compatibility within a react app.

<https://reactstrap.github.io/?path=/story/home-installation--page>

React-chartjs-2

Used to display given data in a graphical format, bar chart, pie chart, etc.

<https://www.npmjs.com/package/react-chartjs-2>

Pigeon-maps

Module that provides a map of the world that can be navigated by the user, with markers that can be set with given coordinates.

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

Application Design

Navigation and Layout

The layout of the application is heavily inspired from assignments specification. The home screen was designed to be a simple page with a title and a hero image as the centerpiece. The header includes a logo for the app and the navigation bar. The navigation bar is how the user gets to most pages, the one page the navigation bar does not take the user to is the volcano page. This is done purposely to lead the user to utilise the list to get to the specified volcano. The list initially had a search input for the country, however, to prevent user error, it was decided to give the explicit list of selectable countries in the form of a table.

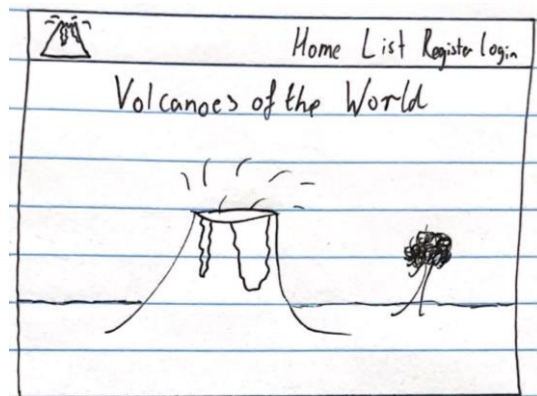


Figure 7: Initial Sketches of Home Page

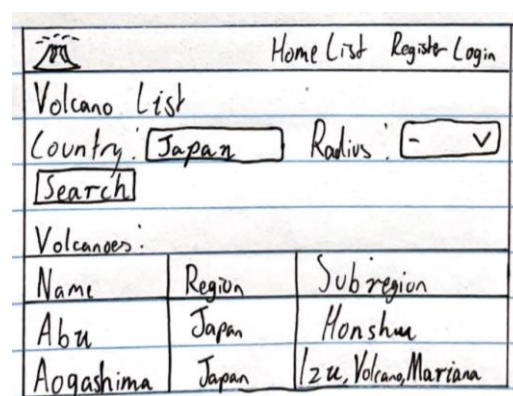


Figure 8: Initial Sketches of Volcano List Page

The initial plan for the volcanoes page was to show everything vertically, making the user scroll down to view the bar chart. However, there was a lot of white space, so the bar chart section was aligned to the right of the map as a better alternative. Register and login pages were always intended to be simple form pages as they currently are.

Usability and Quality of Design

This analysis will be broken into the 6 ingredients for a successful website, first beginning with content. For a user who has an interest in volcanoes, this application certainly satisfies their needs. Components such as the map and bar chart manage to find the right balance between quality verse quantity when displaying data. For a casual internet user that came across this application, there would most likely not be as much of an interest. This is okay though as this website was made specifically for a targeted niche.

Navigation has been mentioned above, but to reiterate the navigation is simple for the user with minimal links.

The visual design is simple but perhaps lacking with placement of components. From the perspective of a typical web browser this website is not necessarily impressive. A card layout would've been a better approach to go about placing components on the webpage. There is certainly consistency and structure, its just missing a more professional approach to the design. A minor issue that should be changed in the future is the navigation bar's links when hovered over have blue text. The colour pallet for the rest of the application is quite suitable, but the blue in this case is not fitting.

In the performance aspect, the app meets the specification to a high standard. It can perform all required functions and given there is a secure connection, the results return almost immediately.

Compatibility is very important for real world applications, as mobiles, different screens, etc. need to be accounted for. For this assignment, varying screen sizes have been considered (if a screen is too small the user can still access all parts of the page by scrolling down).

Finally, interactivity, in the case of this assignment. There is strong error handling, which means plentiful feedback/communication to the user. For majority of cases the application has been designed to be in control but let the user feel like they have the control. It is also adaptive in the sense that if a user requires to see the population density and have gone through the authentication steps, the app rewards them by displaying what they wanted.

Accessibility

In response to Priority 1 Accessibility Requirements. Providing text equivalents for symbols, audio, videos, etc. are very important for users that are visually or hearing impaired. As majority of the website is text, this is okay, but it is clear the importance of alternative text equivalence and it would be a requirement in an upgrade of this react app.

Bar charts and the map never went through a stage where greyscale versions were considered unfortunately.

The website is not dependent on styling sheets. Obviously, it helps a lot but without the CSS, the application still runs.

Due to text equivalents not being a part of the application, content cannot update when dynamic changes occur yet.

The only occurrence of a flicker in the screen is when the user must refresh the page after logging in. Therefore, it is so important to solve this issue in a possible future update of the application. This must be prevented as it could cause possible seizures to people who suffer from epilepsy.

It would be suitable to say that the language used is neither too complex nor too simple for the typical volcano enthusiast. But it's understandable how this is important as excluding a user by intimidation with language is not the experience web developers are trying to convey when creating websites.

Regarding tables, row headers are not present in the country and volcano lists. It could be argued that the name of the country or volcano is a *primary key* which acts as the header of each row.

Technical Description

Architecture

The architecture of the application's source code follows the structure of a typical react application. The public folder contains another folder for images/icons used, as well as the html page index, that the react app *attaches* itself to through the reactDOM. Node_modules simply contain all the modules the application depends on. Package-lock and package, are JSON files that are essential to keep however, were not modified for this assessment.

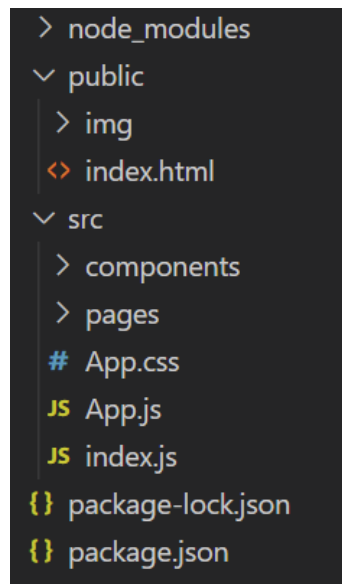


Figure 9: Application Source Code

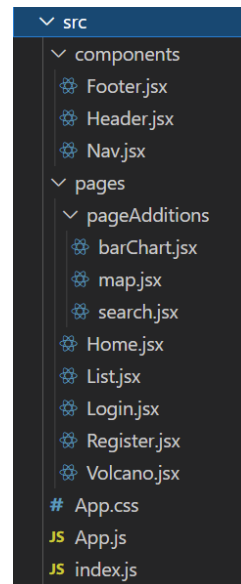


Figure 10: src Architecture

The src folder holds majority of the code, including components (header, footer, navigation bar) and pages (home, list, register, login, and volcano). Within pages, there's three 'page additions', bar chart, map, and search. Search has a connective relationship with the list page as it passes country and radius data for the list page to create a list of volcanoes. App.css is the style sheet of the app, app.js is the code that routes all pages together and index.js displays app.js with the reactDOM.

Test plan

Task	Expected Outcome	Result	Comments	Figure
Select Volcano List in navigation bar	User is taken to Volcano List	Pass		12
Filter country list	Country list is filterd	Pass		13
Select country and population radius and press search button	List of volcanoes is updated	Pass		14
Sort and filter volcano list	Volcano list is sorted and filtered	Pass		15
Select a volcano	User is taken to the Volcano page	Pass		16
Press the back button	The user is taken back to the list page and the country is retained	Pass	Only country is remembered	17
User registers correctly	A message appears stating create user	Pass		18
User logs in correctly	A message appears stating logged in as: {username}	Partial	Page must be refreshed	19
	Navigation bar updates to show logout	Pass	for a change to occur	
User returns to volcano page having logged in	Population Density becomes available	Pass		20
User presses the logout button	User is navigated to the log in page	Partial	Page must be refreshed	21
	Navigation bar shows register and log in	Pass	for a change to occur	
User tries to register without completing the form	An error appears to complete form	Pass		22
User tries to register without matching passwords	An error appears to match passwords	Pass		23
User tries to log in without completing the form	An error appears to complete form	Pass		24
User tries to login with incorrect password	An error appears to match passwords	Pass		25
User is on Volcano List Page (Offline)	No results appear	Pass		26
	Troubleshooting suggestions appear			
User is on Volcano Page (Offline)	No results appear	Pass		27

Difficulties / Exclusions / Unresolved & Persistent Errors /

During the development stage, there was always somewhat of a struggle to add each component of functionality. Whether it was the map, table or logging in, with some perseverance and research, each feature was added to the website.

Considering all error cases that could occur using the react app, proved to be quite tedious. For the login/register pages, incorrect form applications lead to the error situations. The error handling for these scenarios is satisfactory. In comparison to the volcano list page where errors occur due to several conditions such as a query returning nothing, no connection to the server or manipulation of the URL. When this happens, a list of possible causes to the issue is appended to the screen, this is a bit vague and makes troubleshooting more difficult for the user than it could be.

When it comes to bugs that have not been completely resolved, pigeon maps have an issue where if the user reaches the edge of the map and goes past this point, a grey segment can be seen. This is slightly unprofessional and after some time of trying to find a solution it was decided the map should be initially zoomed in, in hopes of trying to *cover up* the issue.

Whilst attempting to have a navigation bar that is responsive to the user's authentication, inadvertently created some errors that at this point are still outstanding. As a result, the user now must refresh the page for the website to recognize the user has logged in with a token. This unfortunately makes the SPA concept seem redundant in a sense.

Extensions

Aside from fixing issues stated above, the volcano page's error handling could give more insightful prompts to the user and pressing the back button could even recall population radius as well.

A potential upgrade to this project, would be to obtain images of these volcanoes. The purpose of the images would be to present the user with a slideshow next to the map to give the user a greater understanding of the location around the volcano.

User guide

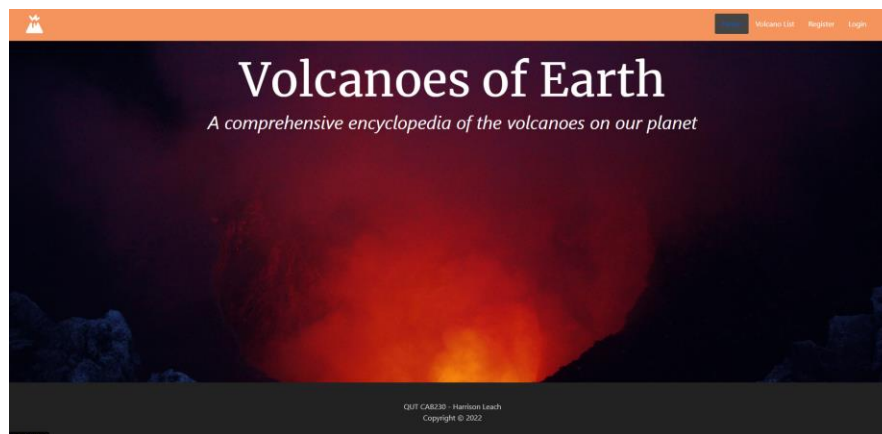


Figure 11: Home Screen

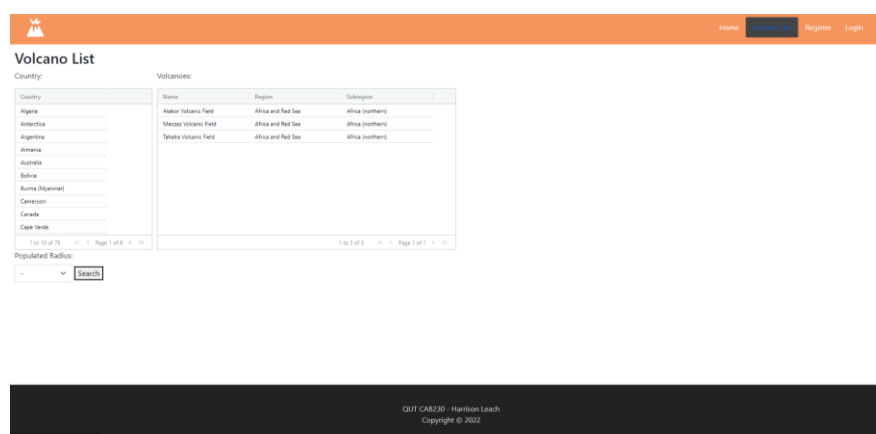


Figure 12: Volcano List (Initial State Always Shows First Country, Algeria)



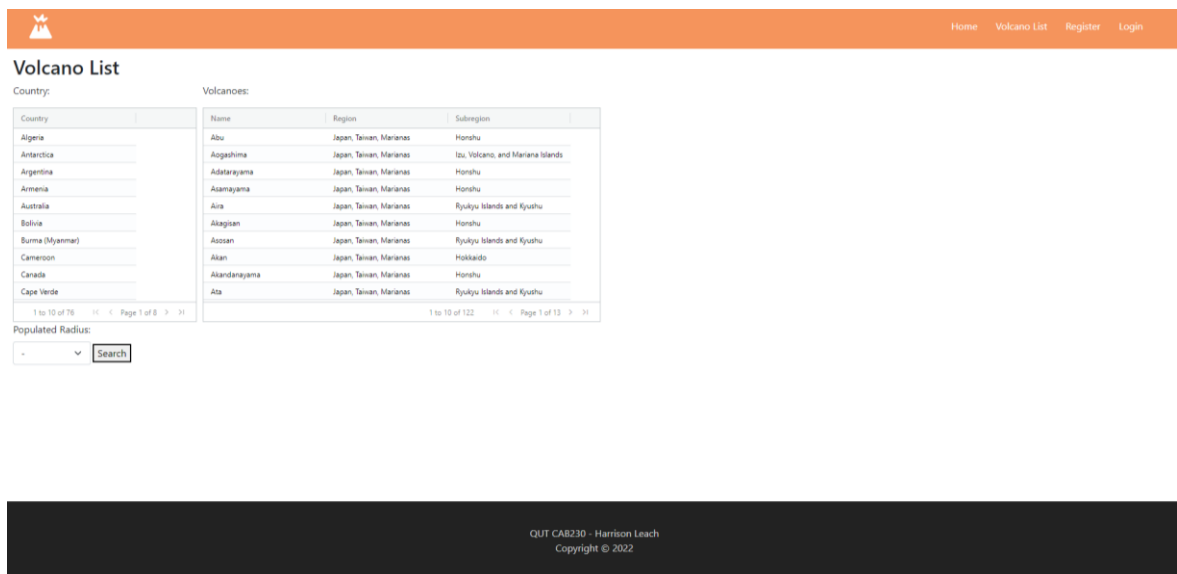


Figure 17: Pressed Back from Volcano Page (Remembers Country)



Figure 18: Registration - User Created (User: example, Password: 123)



Figure 19: Login - User Logged In (Navigation Bar Updates Upon Refreshing)

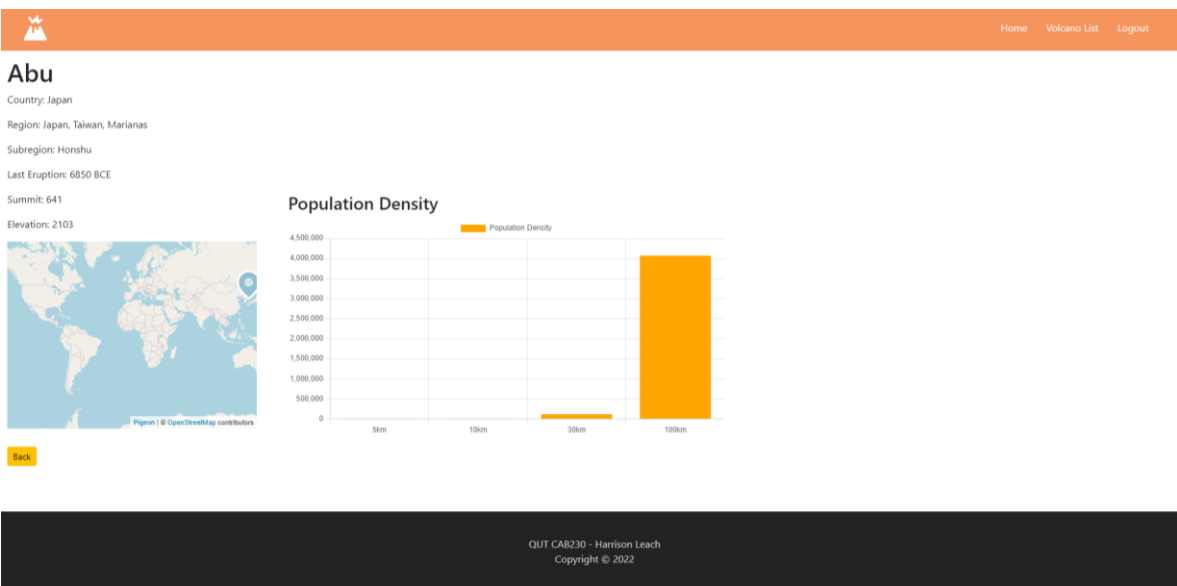


Figure 20: Returning to Volcano Page Having Logged In

[Home](#)[Volcano List](#)[Register](#)[Login](#)

Login

Email:

jane.doe@example.com

Password:

Password

Login

Figure 21: Logging Out (Navigation Updates Upon Refreshing)

References

Twitter Emoji. Volcano Emoji Icon in Flat Style [Image]. Icon Scout. <https://iconscout.com/icon/volcano-eruption-mountain-natural-disaster-emoji-symbol>

FREE SVG. (2019). Black icon of a volcano [Image]. Volcano Silhouette. <https://freesvg.org/volcano-silhouette>

rawpixel. Free volcano lava photo, public domain CC0 image [Photograph]. Images. <https://www.rawpixel.com/image/5917161/free-volcano-lava-photo-public-domain-cc0-image>

Appendix



Register

Email:

Password:

Re-enter Password:

Error: Request body incomplete, both email and password are required

Register

Figure 22: Registration - Fields Not Completed



Register

Email:

Password:

Re-enter Password:

Error: Passwords do not match

Register

Figure 23: Registration - Passwords Don't Match



Login

Email:

Password:

Error: Request body incomplete, both email and password are required

Login

Figure 24: Login - Fields Not Completed



Login

Email:

Password:

Error: Incorrect email or password

Login

Figure 25: Login - Passwords Don't Match

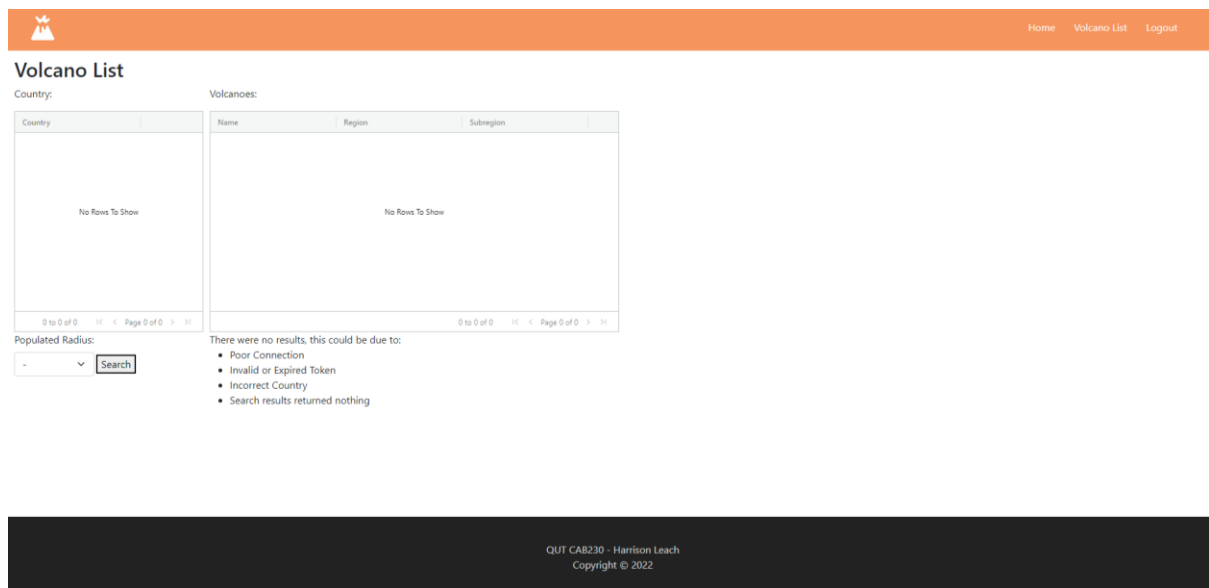


Figure 26: Volcano List Page (Offline)



Figure 27: Volcano Page (Offline)