2024

# CAB230 Assignment 2 Client Side

CAB230

Volcano API – Client Side Application

<Jason Zhang>

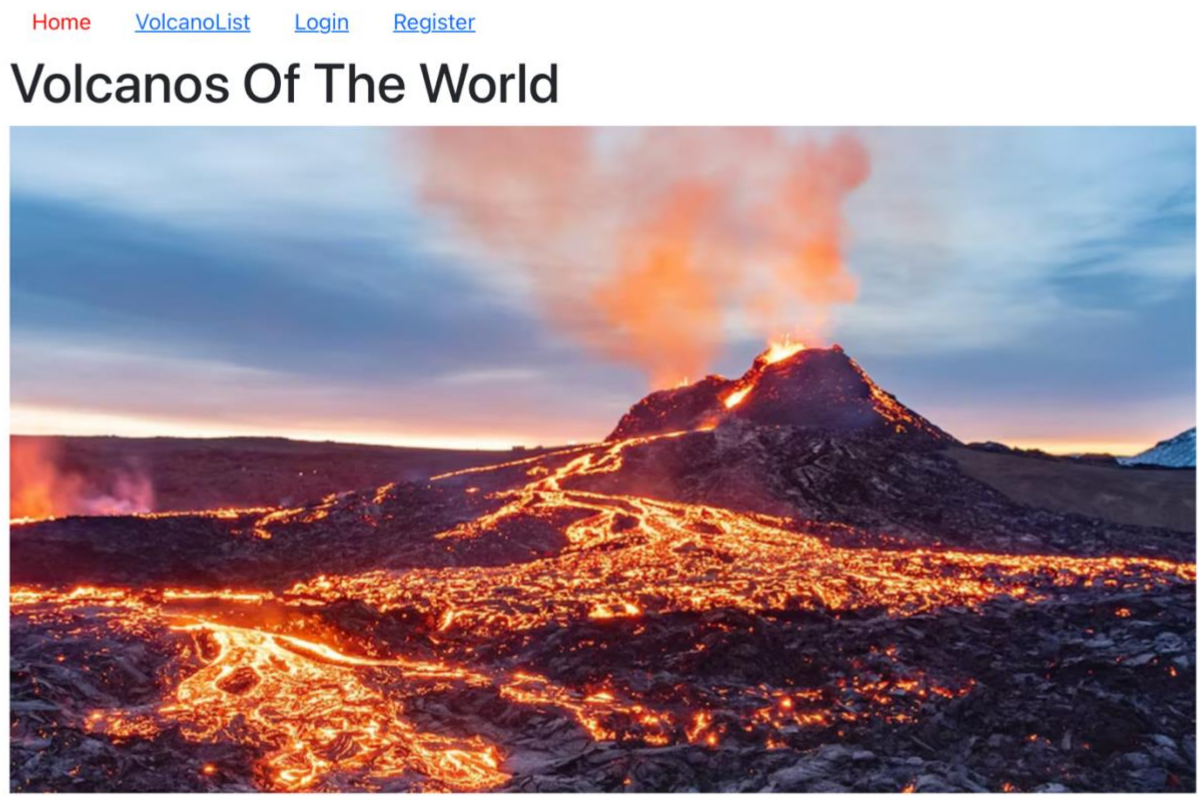
<n11255838>

10/5/2024

## Contents

## Introduction

### Purpose & description



This react-based web application is used to allow users to view and analyze data about volcanoes through a REST API about volcanoes. Specifically, it includes the following features. First, users can register and log in to this web app and see a large volcanic photo displayed on the homepage. Second, users can search for relevant volcanic information by selecting a country and region, and optionally filter the list of volcanoes to those that have people living within a specified radius. When a user clicks on a volcano in the list, more specific information about the volcano will be displayed, including its country, region, elevation, etc., and its location will be marked on the map.

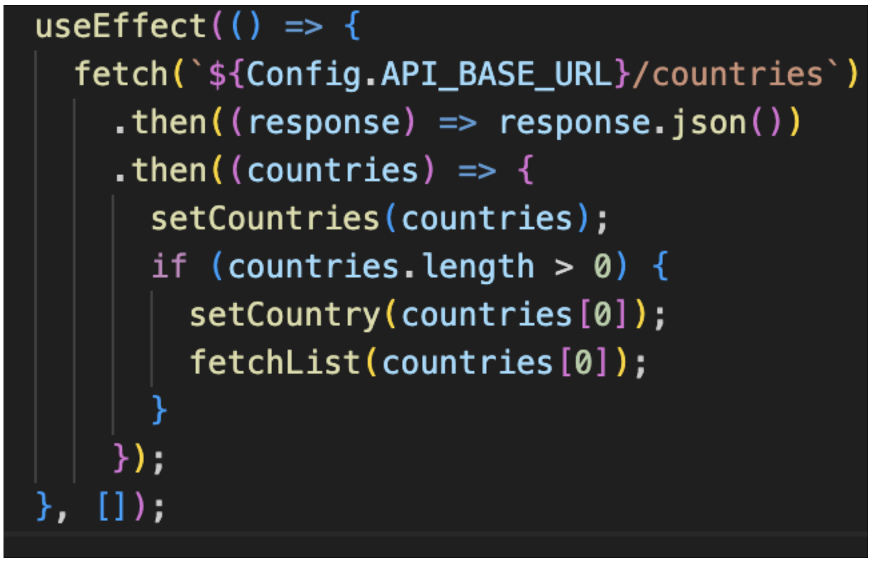
When rendering a chart related to population density, I have used Chart.js, which is a popular JavaScript library for creating charts. When developing the navigation bar, I have used Bootstrap, which is a popular CSS framework that can help speed up front-end development.

## Completeness and Limitations

- Data and data endpoints handling: The data obtained from the data endpoint is displayed accurately and clearly.
- Navigation: Navigation bar is handled using React Router, controlled forms are used for inputs. The chosen components and the data displaying are close alignment.
- Overall design: The overall design of the website is well thought out and it is easy to use. However, there is no back button designed to allow users to return directly to the previous directory; users can only return to specific pages by clicking on one of the four options in the limited navigation bar.
- Map feature: A map component is used to display the location of the volcano and a chart to show the population density data.
- Table: Using AG Grid to show the list of volcanoes on the volcano list page.

## Use of End Points

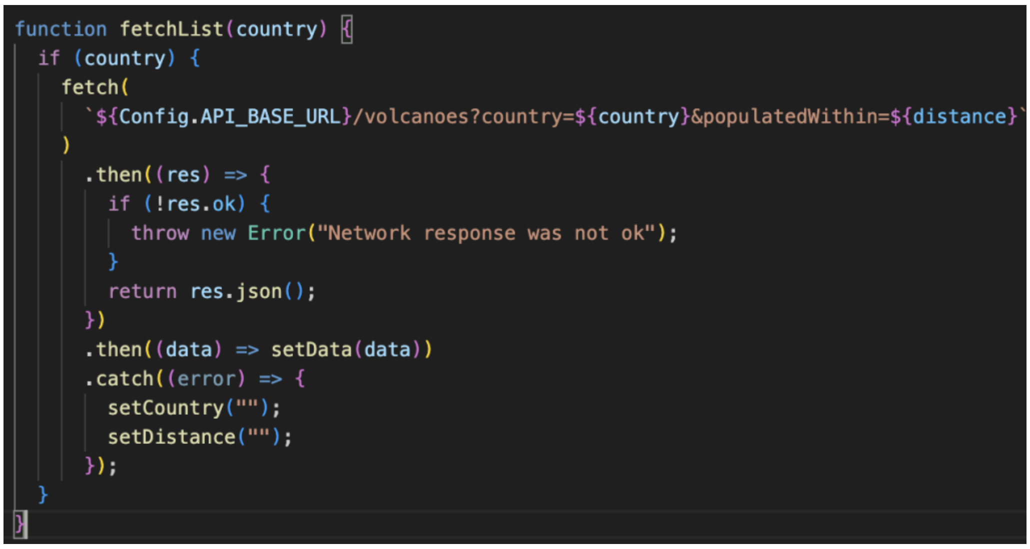
### */countries*



```
useEffect(() => {
  fetch(`${Config.API_BASE_URL}/countries`)
    .then((response) => response.json())
    .then((countries) => {
      setCountries(countries);
      if (countries.length > 0) {
        setCountry(countries[0]);
        fetchList(countries[0]);
      }
    });
}, []);
```

It is responsible for fetching the list of countries through the countries endpoint, which is used to display a list of volcanoes for a selected country within a specified distance.

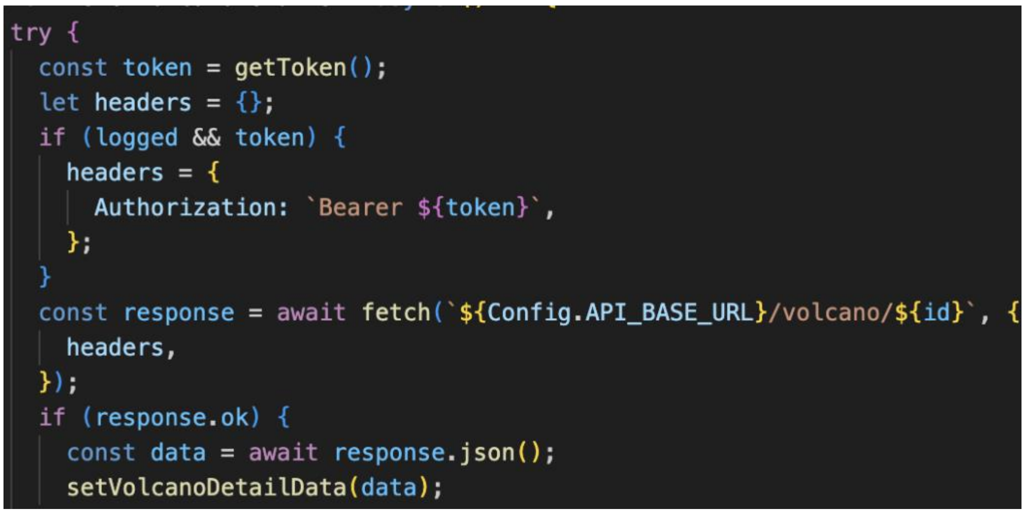
*/volcanoes*



```
function fetchList(country) {
  if (country) {
    fetch(
      `${Config.API_BASE_URL}/volcanoes?country=${country}&populatedWithin=${distance}`
    )
      .then((res) => {
        if (!res.ok) {
          throw new Error("Network response was not ok");
        }
        return res.json();
      })
      .then((data) => setData(data))
      .catch((error) => {
        setCountry("");
        setDistance("");
      });
  }
}
```

The volcanoes endpoint has a mandatory query parameter: country. It also has an optional query parameter: populatedWithin. This is used to fetch a list of volcanoes based on the selected country and a certain distance.

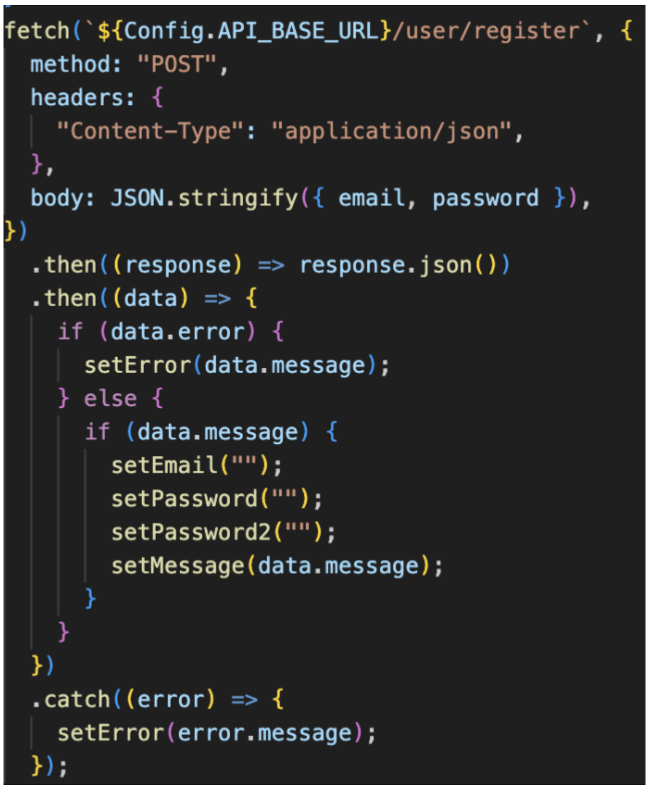
*/volcano/{id}*



```
try {
  const token = getToken();
  let headers = {};
  if (logged && token) {
    headers = {
      Authorization: `Bearer ${token}`,
    };
  }
  const response = await fetch(`${Config.API_BASE_URL}/volcano/${id}`, {
    headers,
  });
  if (response.ok) {
    const data = await response.json();
    setVolcanoDetailData(data);
```

The volcano endpoint has a mandatory ID path parameter. This volcano ID
can be obtained from the /volcanoes endpoint. This is used to get the details of a specific volcano from the API.

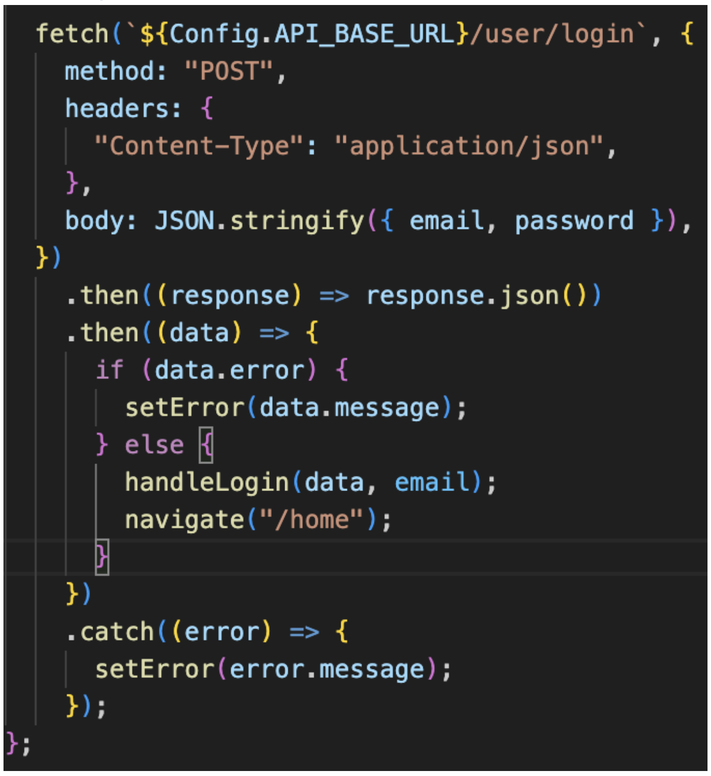
*/user/register*



```
fetch(`${Config.API_BASE_URL}/user/register`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify({ email, password }),
})
  .then((response) => response.json())
  .then((data) => {
    if (data.error) {
      setError(data.message);
    } else {
      if (data.message) {
        setEmail("");
        setPassword("");
        setPassword2("");
        setMessage(data.message);
      }
    }
  })
  .catch((error) => {
    setError(error.message);
  });
```

This is a part of a function that handles user authentication. It uses the Fetch API to make a POST request to a register endpoint.

*/user/login*



```
  fetch(`${Config.API_BASE_URL}/user/login`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ email, password }),
  })
    .then((response) => response.json())
    .then((data) => {
      if (data.error) {
        setError(data.message);
      } else {
        handleLogin(data, email);
        navigate("/home");
      }
    })
    .catch((error) => {
      setError(error.message);
    });
};
```

This is a part of a function that handles user authentication. It uses the Fetch API to make a POST request to a login endpoint.

## Modules Used

### *Ag-grid-react*

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

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

### *Pigeon-maps*

Module to make maps without any external dependencies.

https://github.com/mariusandra/pigeon-maps#readme

### *Chart.js*

Module to provide a simple yet flexible way to embed interactive and animated charts into a web page.

https://www.chartjs.org

### *React-chatjs-2*

It is a React wrapper for Chart.js, making it easier to integrate Chart.js into React applications.

https://github.com/reactchartjs/react-chartjs-2

### *Bootstrap*

Module to provide a solid foundation for developing responsive websites quickly and with less effort, ensuring that web applications look and work well across all devices.

https://getbootstrap.com/

### *Reactstrap*

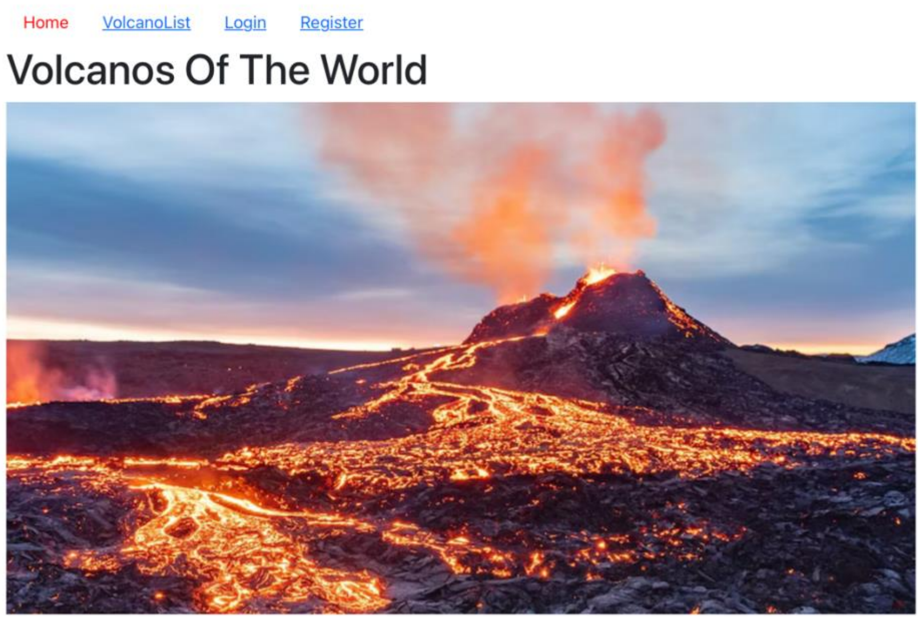
A library that brings Bootstrap`s core functionality to React application.

https://github.com/reactstrap/reactstrap#readme

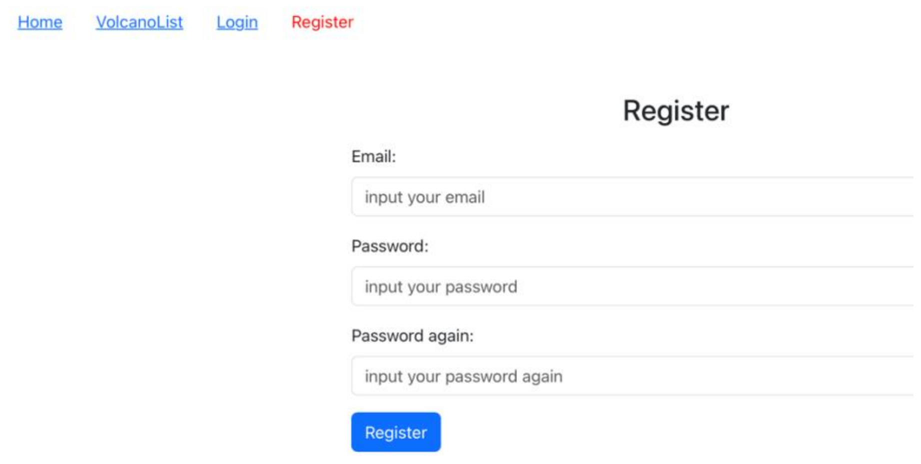
## Application Design

### Navigation and Layout

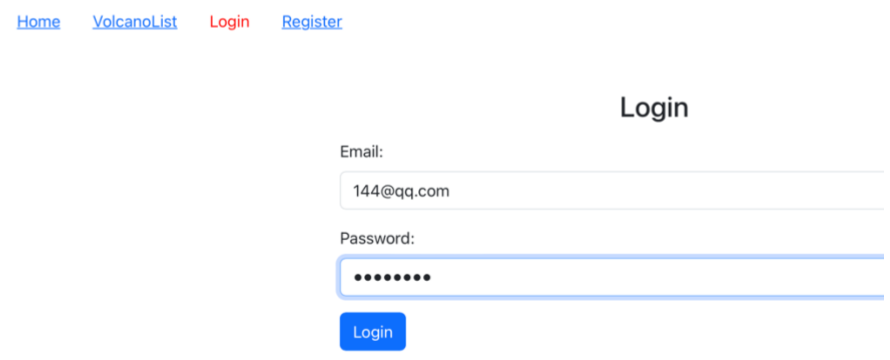
When users first enter this web application, they come to the home page, which includes a picture of a volcano and four navigation keys in the top left corner of the page, including home, volcano list, login, and register.



Afterward, users will need to click 'register' to sign up as new users. This takes them to the registration page, where they need to enter their email address and input the same password twice to ensure that what they are entering is indeed the content they intend to use as their password.



After registration is complete, users need to click the login button in the top left corner to log in. At this point, they need to enter the email address and password used during registration.

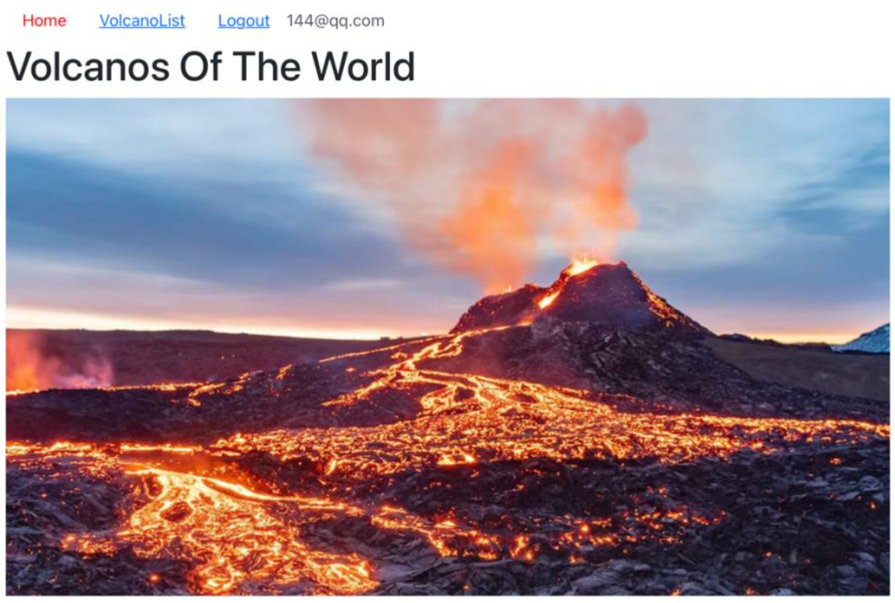


Login

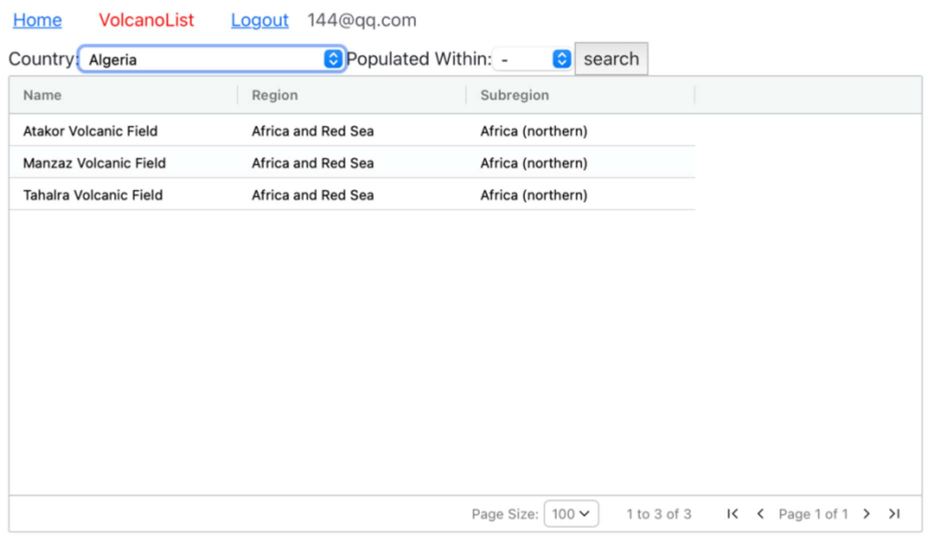
Email: 144@qq.com

Password: ••••••••

After clicking login, the page will automatically redirect to the home page, and the navigation bar will display 'logout' along with the username, replacing the positions of 'login' and 'register'.



When users click on 'volcano list', the information about the volcanoes is displayed here in table format. The default page shows the names of the countries arranged in alphabetical order starting with the first letter. Users can select a country using the dropdown list above the table, and there is an optional 'Populated With' field that allows users to also select a population density radius as a filter.



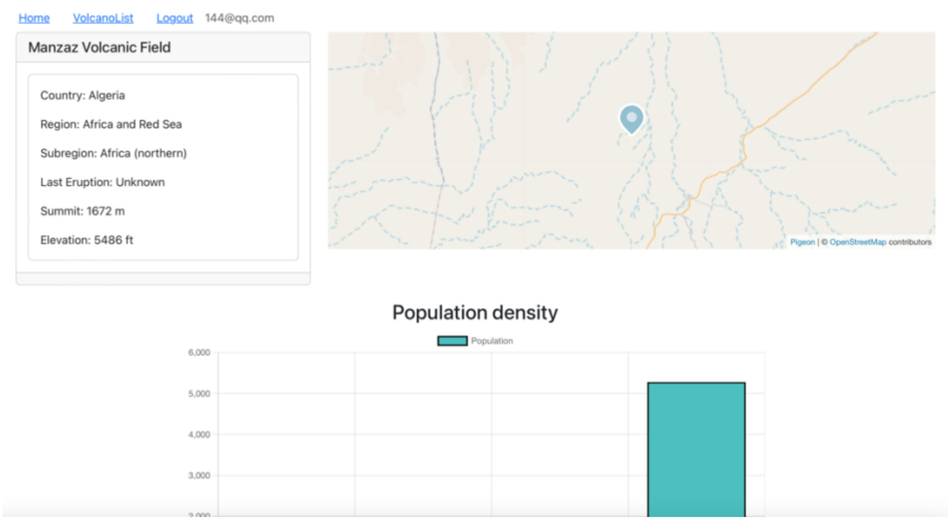
Home VolcanoList Logout 144@qq.com

Country: Algeria Populated Within: - search

| Name | Region | Subregion |
|---|---|---|
| Atakor Volcanic Field | Africa and Red Sea | Africa (northern) |
| Manzaz Volcanic Field | Africa and Red Sea | Africa (northern) |
| Tahalra Volcanic Field | Africa and Red Sea | Africa (northern) |

Page Size: 100 1 to 3 of 3 Page 1 of 1

After clicking on a volcano in the table, users will be redirected to a new screen. This screen displays detailed information about the volcano and its geographical location on a map, as well as the population density shown in chart form.



Home VolcanoList Logout 144@qq.com

Manzaz Volcanic Field

Country: Algeria

Region: Africa and Red Sea

Subregion: Africa (northern)

Last Eruption: Unknown

Summit: 1672 m

Elevation: 5486 ft

Up to this point, all the functionalities of the web application have been demonstrated. When clicking 'logout', the screen will revert to the initial homepage state.

## Usability and Quality of Design

- Content: The web content involved here is simply fetched from a volcano API, so the accuracy and timeliness of the content are guaranteed. Moreover, we have formatted the fetched content into tables and maps, allowing users to read and digest the data more intuitively and easily.
- Organisation: The layout of the entire webpage is simple and clear, with the navigation bar located in the top left corner and arranged in this order. Clicking on any of the buttons will display its content directly in the middle of the page, making the content of the entire web application consistent enough so that users can easily find what they are looking for and understand how to operate it to achieve their goals.
- Navigation: As mentioned earlier in the navigation and layout section, when designing the navigation process of this web app, I made it as simple as possible so that users can clearly know what to do and what to do first. Especially when browsing the volcano list, when users hover their mouse over a volcano, that row will darken, indicating that they can click to learn more about that volcano's specific information. Another point worth noting is that after users log in successfully, I replaced 'login' and 'register' with 'logout' and the username in the navigation bar. This makes the login process very logical and helps prevent errors.
- Visual design:  Although the overall colour scheme of this web app is quite monochromatic, this prevents the page elements from becoming overly complicated and cluttered due to an excess of colours, which could make it difficult for users to find the information they are looking for. The layout of various elements on the page is rational and compact, making it easy for users to navigate and search for content.
- Performance: This is designed as a single-page application(SAP), the major benefit of this approach is the speed. Most resources are loaded initially. Once loaded, the application is usually very responsive to user input and navigation - can support rich UI interactions and shared state between screens. (*The Single Page App, n.d.)*
- Compatibility: This web app uses Reactstrap and Bootstrap modules, which enable it to be compatible with any device, including computers, smartphones, and tablets, as well as with any browser.
- Interactivity: When designing this web app, I also considered its interactivity with users, especially during registration and login. If the content entered by the user does not match the expectations, such as not entering a correctly formatted email address or entering different passwords during registration, there will be relevant error messages to guide the user on how to complete the operations correctly. This way, users will not be left in a predicament, unsure of how to proceed with using the web app.

## Accessibility

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

There is a volcano image in my web app. And there are currently no solutions implemented to address this principle in this web app. In order to meet this requirement, I should use 'alt' attributes to provide text alternatives that describe the purpose and the content of this image.

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

Currently, this web app only incorporates basic blue and white colours, both of which are simple and basic tones, thus satisfying this requirement.

3. **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.**

Currently, the textual content in this web app relies heavily on CSS for standardization and layout. Without the CSS rules, all content may become jumbled and difficult to read. In order to address this issue, we should use proper and logical HTML tags like **<h1>** to **<h6>** for headings, **<p>** for paragraphs, and **<ul>/<ol>** for lists. This ensures content is structured and comprehensible even without CSS.

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

The dynamic content involved in this web app is merely simple data about volcanoes, so its dynamic changes will not affect the existing text equivalents. Therefore, the current app meets this requirement."

5. **Avoid causing the screen to flicker.**

Since the current web app does not contain dynamic images or videos, there will be no flickering when browsing the pages.

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

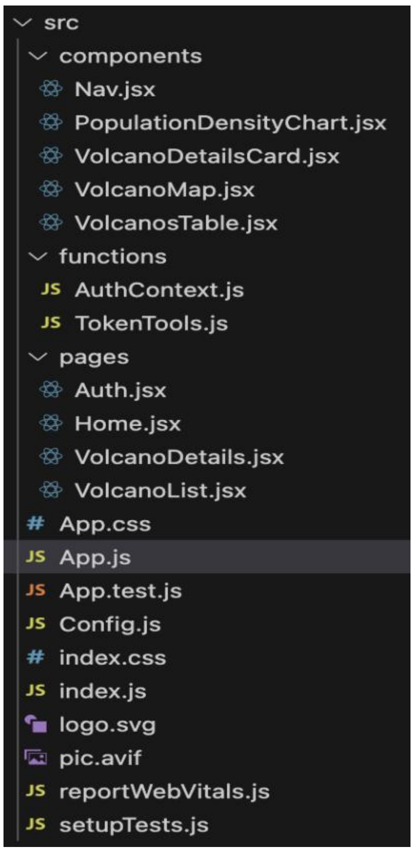
The textual content of the current web app is short and easy to understand, so it is already sufficiently clear and simple.

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

Thanks to the powerful capabilities of the AG Grid module, the table it generates is very clear and easy to understand. The rows and columns are well-defined, and each column header uses a different font and colour from the content, making it easy to distinguish between them.

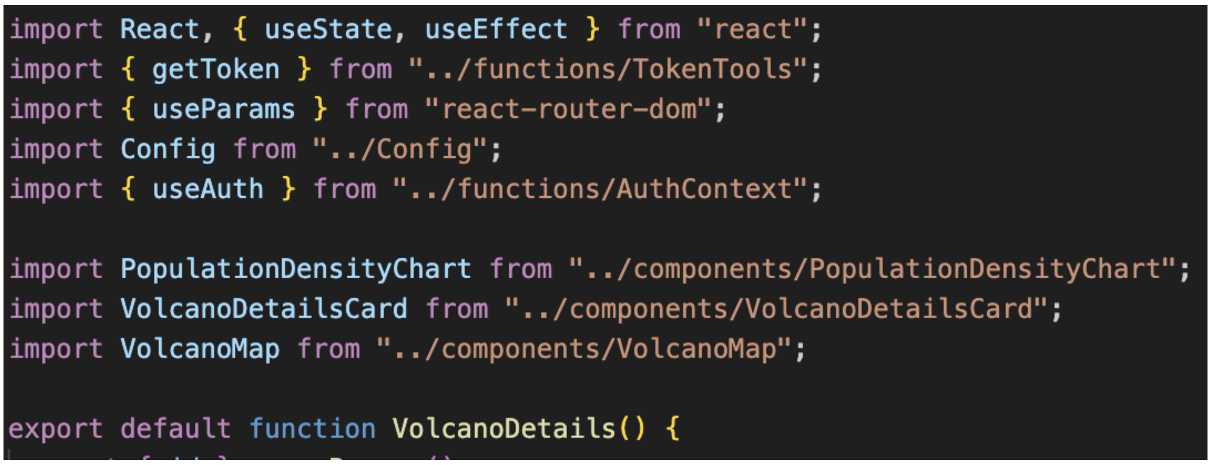
## Technical Description

### Architecture



```
src
  components
    Nav.jsx
    PopulationDensityChart.jsx
    VolcanoDetailsCard.jsx
    VolcanoMap.jsx
    VolcanosTable.jsx
  functions
    AuthContext.js
    TokenTools.js
  pages
    Auth.jsx
    Home.jsx
    VolcanoDetails.jsx
    VolcanoList.jsx
  App.css
  App.js
  App.test.js
  Config.js
  index.css
  index.js
  logo.svg
  pic.avif
  reportWebVitals.js
  setupTests.js
```

In the src directory, I created three folders: components, functions, and pages. The 'components' folder is used to store all the functional components needed for the app; 'functions' primarily includes features for handling tokens and managing login status; finally, 'pages' is used to implement the main pages of the app and their functionalities. I combined the register and login pages into 'auth' because their code is very similar.



```
import React, { useState, useEffect } from "react";
import { getToken } from "../functions/TokenTools";
import { useParams } from "react-router-dom";
import Config from "../Config";
import { useAuth } from "../functions/AuthContext";

import PopulationDensityChart from "../components/PopulationDensityChart";
import VolcanoDetailsCard from "../components/VolcanoDetailsCard";
import VolcanoMap from "../components/VolcanoMap";

export default function VolcanoDetails() {
```

The import statement is used to include functions, objects, or values from other files into the current file, so that the current file can use it to implement relative function where needed.

## Test plan

| Task | Expected Outcome | Result | Screenshots (see Appendix ) |
|---|---|---|---|
| Register new user | The new user is successfully register | Pass | 01 |
| Input incorrect email format | Error message prompt | Pass | 02 |
| Input incorrect password format | Error message prompt | Pass | 03 |
| Input different password when registering | Error message prompt | Pass | 04 |
| Login failed | Error message prompt | Pass | 05 |
| Login successful | Redirect to home page and show username at navigation bar | Pass | 06 |
| Search volcano with filter | Filtered volcanoes information present in table | Pass | 07 |
| Display volcano detail, show location in a map, show population as a chart (Authorized) | Volcano details, the location map and the chart are presented | Pass | 08 |
| Display volcano detail, show location in a map (Unauthorized) | Volcano details and the location map are presented | Pass | 09 |

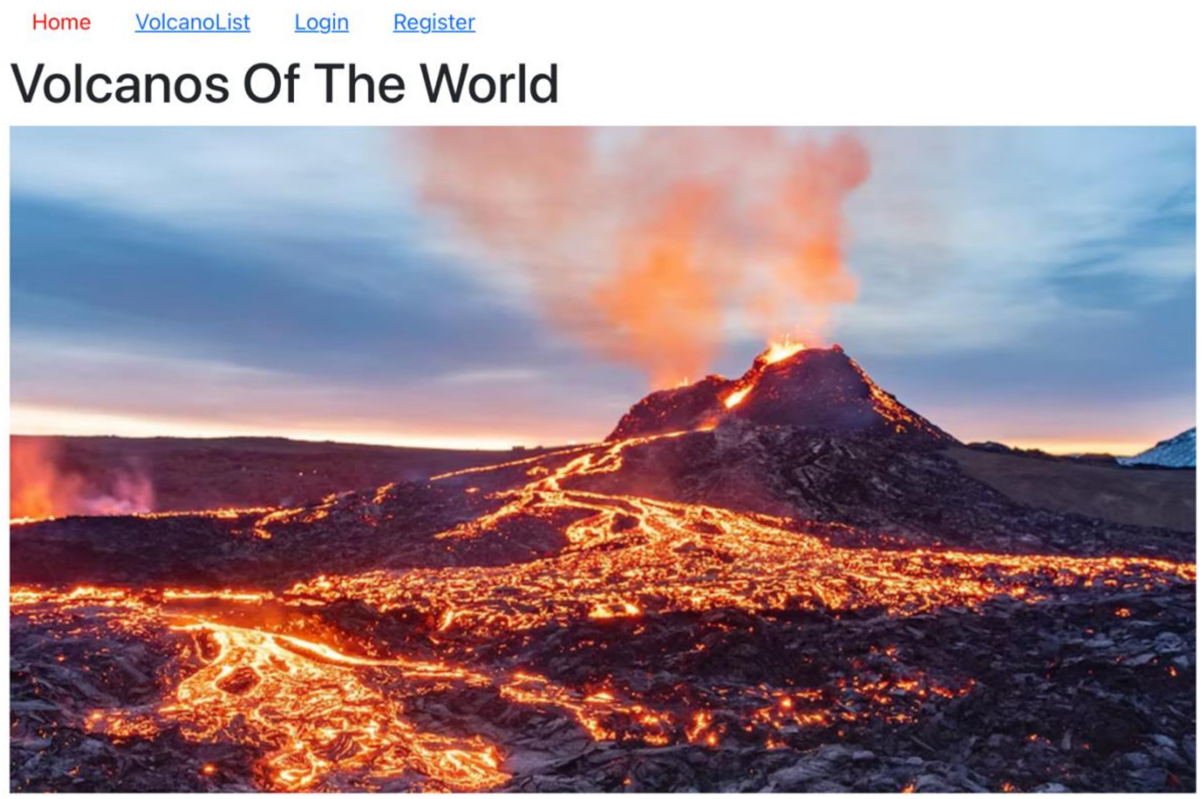
## Difficulties / Exclusions / unresolved & persistent errors /

In the assessment specification, it was mentioned that we need to decide whether to display the population density chart based on the user's login status. This requires a feature to determine the user's login status. Initially, I was unsure how to design this feature because I needed to create a global state for the user's authentication status, so that this state can be accessed from any component in the application without having to pass props down through multiple levels of components. Later, I revisited the video from live lecture 4 by the teacher and referred to the 'state-context-demo' case study, which helped me solve this problem.

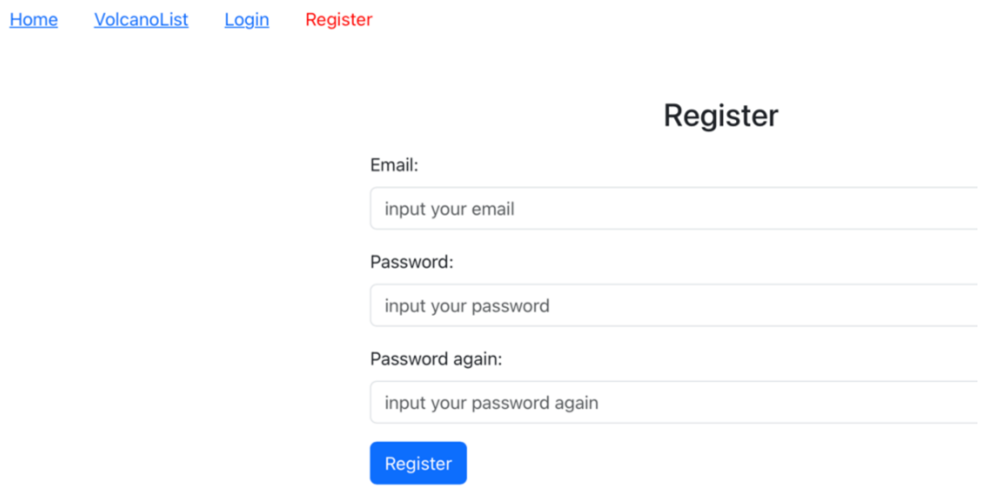
However, the drawbacks of current web application is, as I mentioned before, there is no back button designed to allow users to return directly to the previous directory, furthermore, when users have to return to specific pages by clicking on one of the four options in the limited navigation bar, the previous search results will disappear. I have not managed to find a solution to solve this drawback.

## User guide

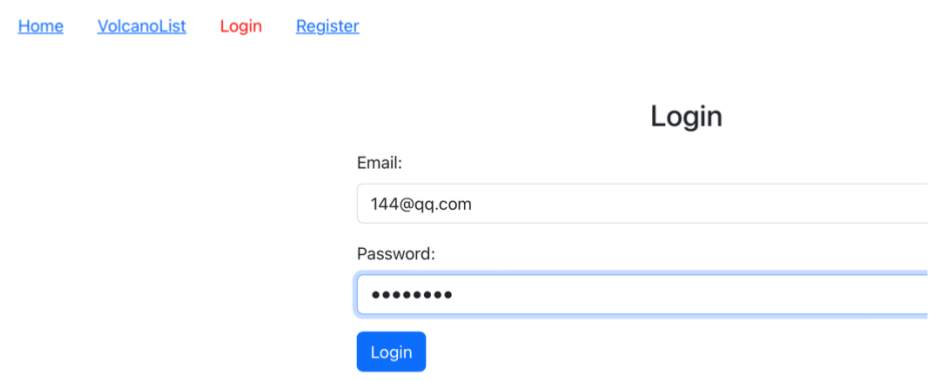
When it is your  first time entering this web application, you will come to the home page.



Afterward, clicking on ‘register’ to register a new user. You will need to use the correct email format and for the password, the length is at least 8 characters.



After registration is complete, you need to click the login button in the top left corner to log in. At this point, you need to enter the email address and password used during registration.



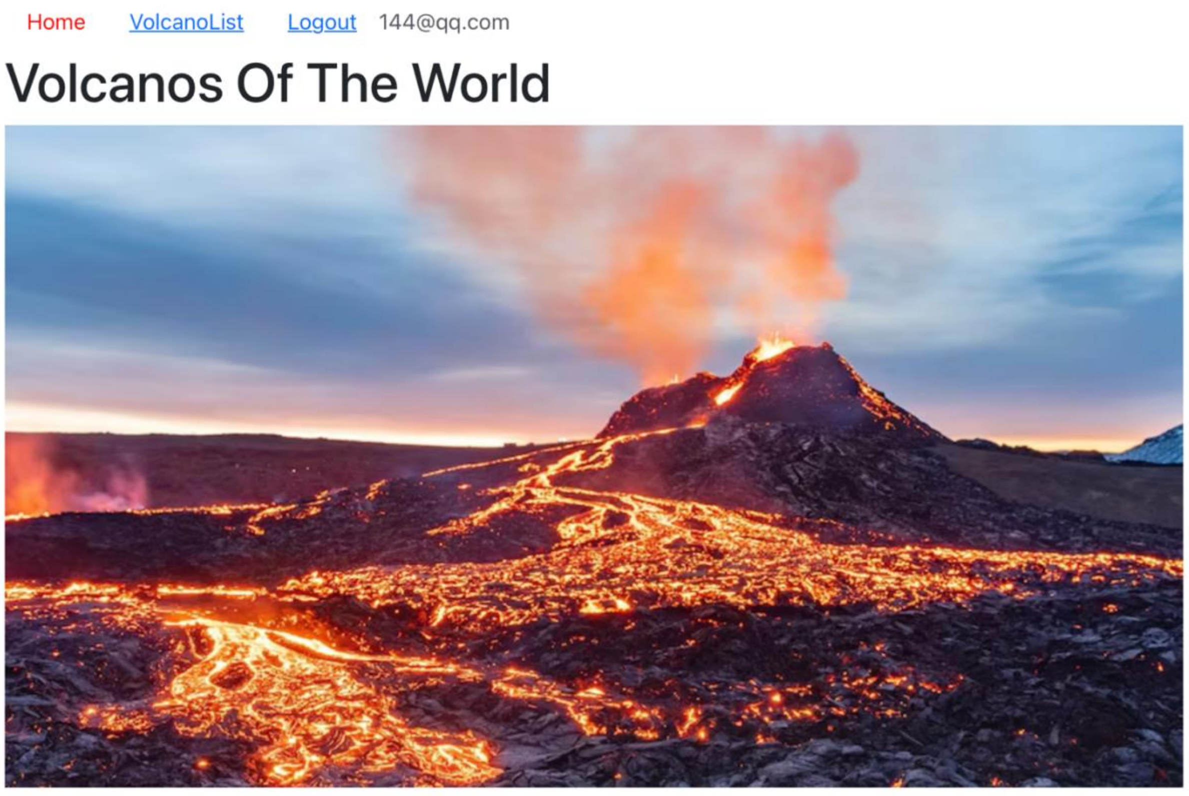
Email:

144@qq.com

Password:

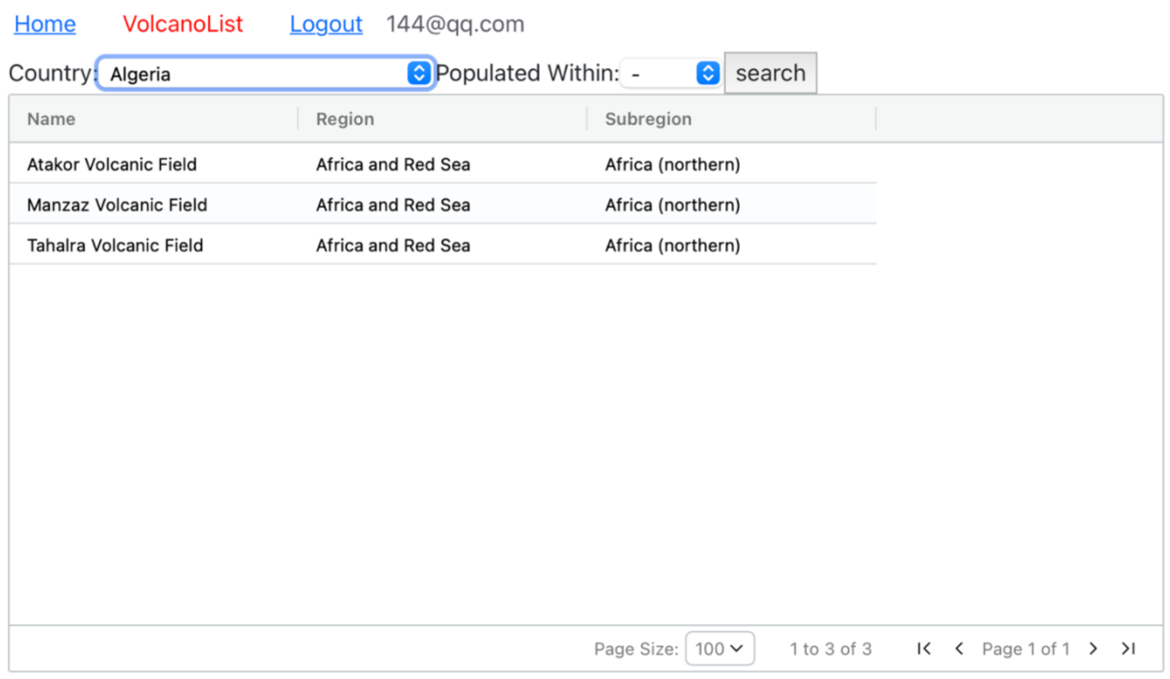
•••••••••

After clicking login, the page will automatically redirect to the home page.



Volcanos Of The World

Now click on 'volcano list', the information about the volcanoes is displayed here in table format. The default page shows the names of the countries arranged in alphabetical order starting with the first letter. You can select a country using the dropdown list above the table, and there is an optional 'Populated With' field that allows you to also select a population density radius as a filter.



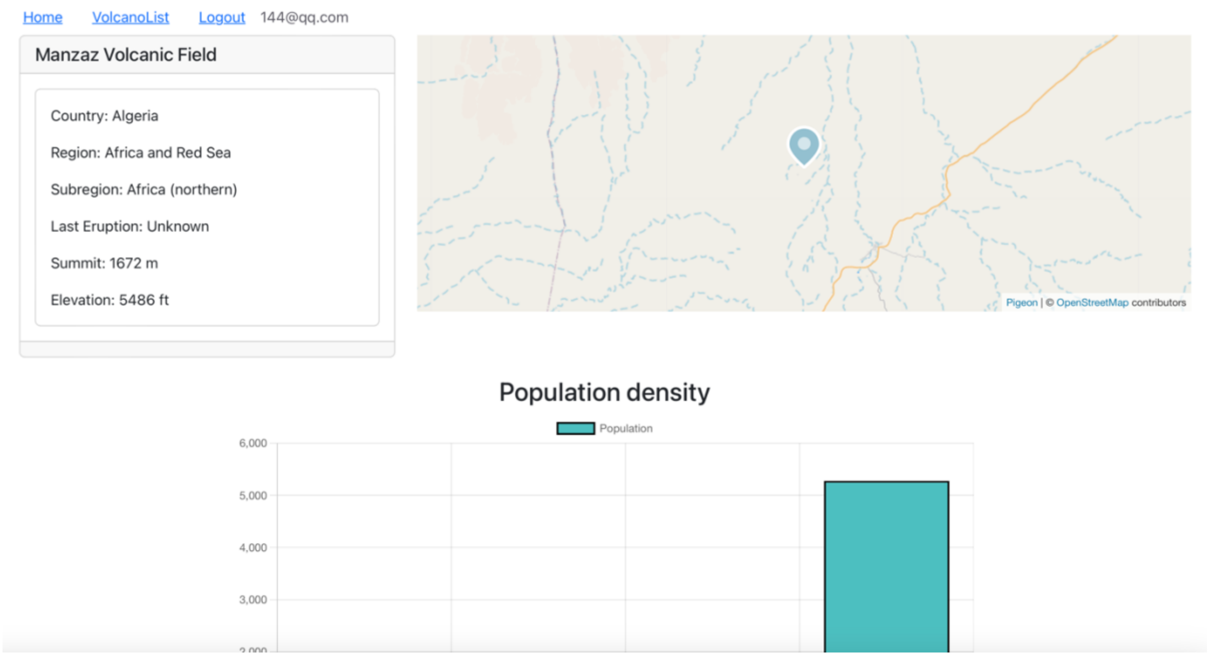
Home VolcanoList Logout 144@qq.com

Country: Algeria Populated Within: - search

| Name | Region | Subregion |
|---|---|---|
| Atakor Volcanic Field | Africa and Red Sea | Africa (northern) |
| Manzaz Volcanic Field | Africa and Red Sea | Africa (northern) |
| Tahalra Volcanic Field | Africa and Red Sea | Africa (northern) |

Page Size: 100 1 to 3 of 3 Page 1 of 1

After clicking on a volcano in the table, users will be redirected to a new screen. This screen displays detailed information about the volcano and its geographical location on a map, as well as the population density shown in chart form.



Home VolcanoList Logout 144@qq.com

Manzaz Volcanic Field

Country: Algeria

Region: Africa and Red Sea

Subregion: Africa (northern)

Last Eruption: Unknown

Summit: 1672 m

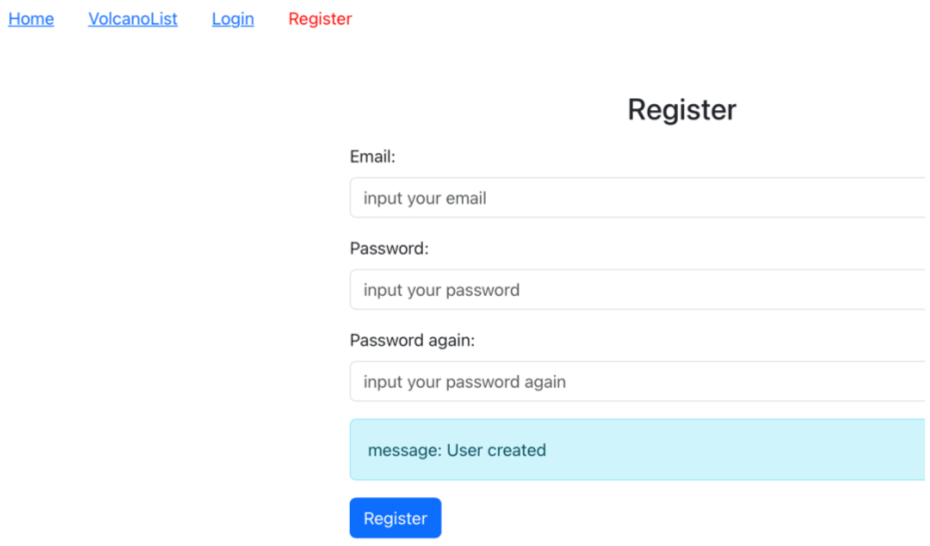
Elevation: 5486 ft

Up to this point, all the functionalities of the web application have been demonstrated. When clicking 'logout', the screen will revert to the initial homepage state.

## References

*The Single Page App, CAB230 Lecture 4.1 page 4*.
https://canvas.qut.edu.au/courses/16668/files/3625015/download?wrap=1

## Appendix



Register

Email:

input your email

Password:

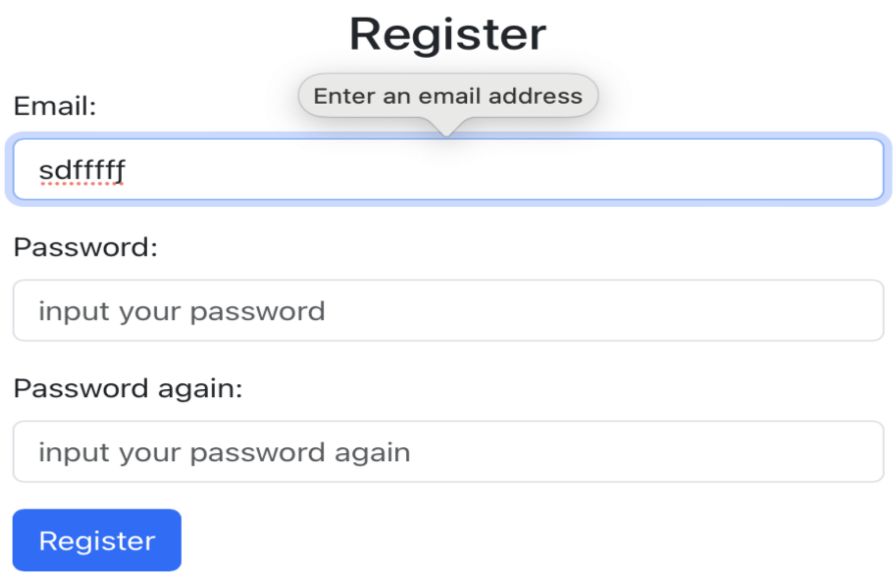
input your password

Password again:

input your password again

message: User created

(pic 01)



Register

Email:

Enter an email address

sdfffff

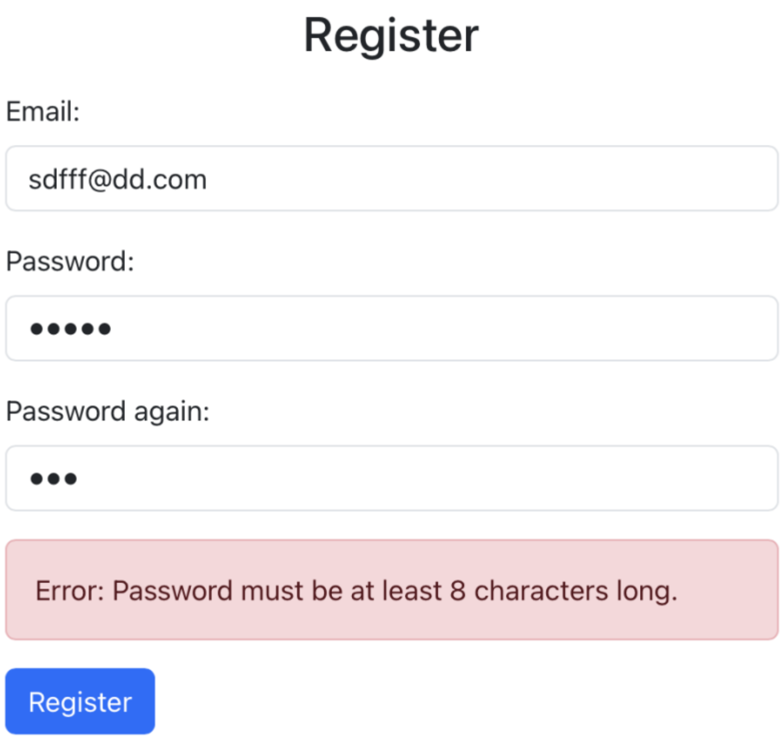
Password:

input your password

Password again:

input your password again

(pic 02)


Register

Email:

sdfff@dd.com

Password:

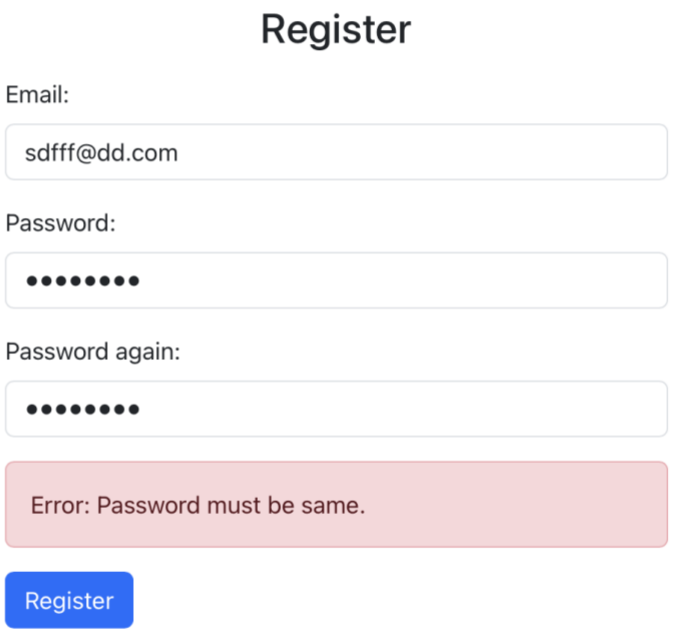
●●●●●

Password again:

●●●

Error: Password must be at least 8 characters long.

(pic 03)


Register

Email:

sdfff@dd.com

Password:

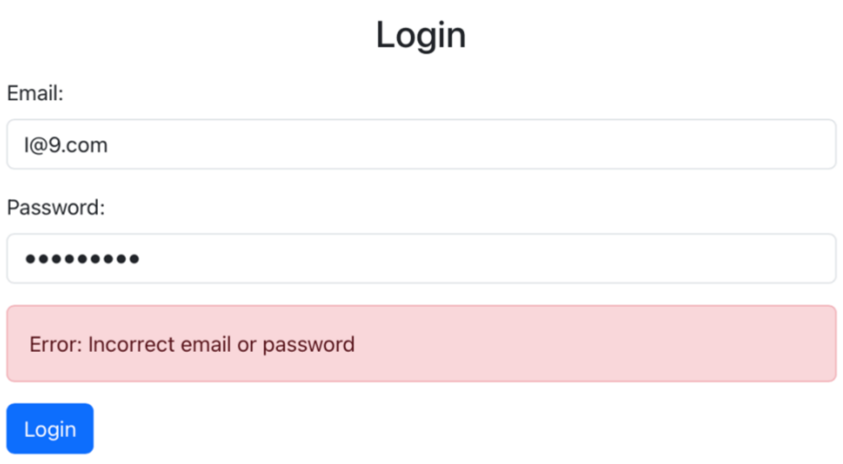
●●●●●●●●

Password again:

●●●●●●●●

Error: Password must be same.

(pic 04)



Login

Email:

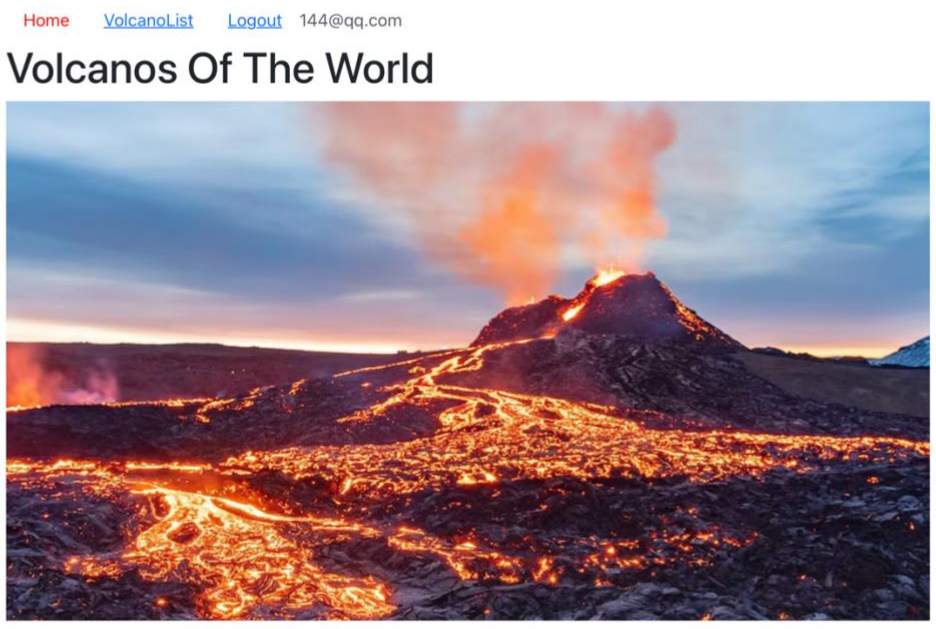
l@9.com

Password:

••••••••••

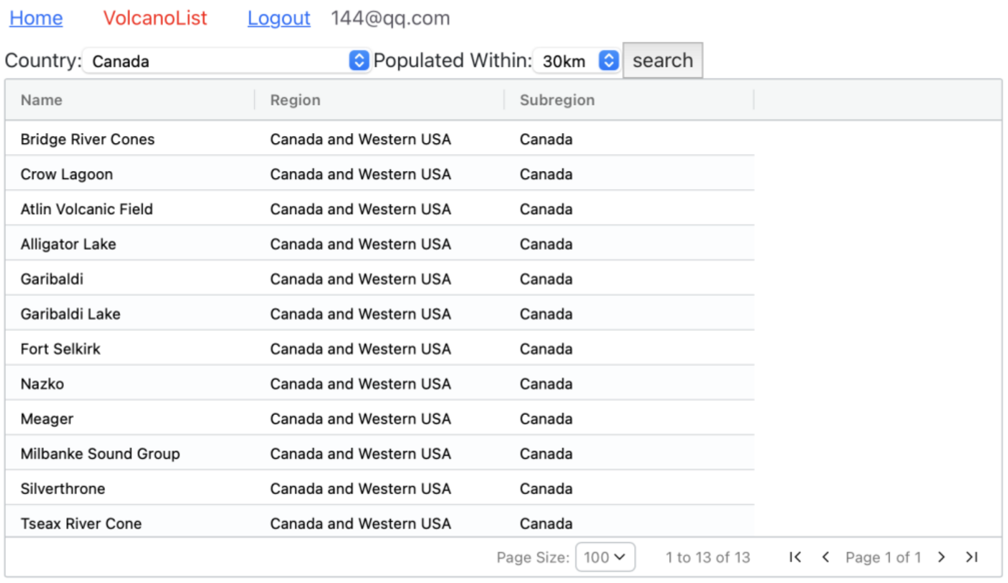
Error: Incorrect email or password

(pic 05)



Volcanos Of The World

(pic 06)

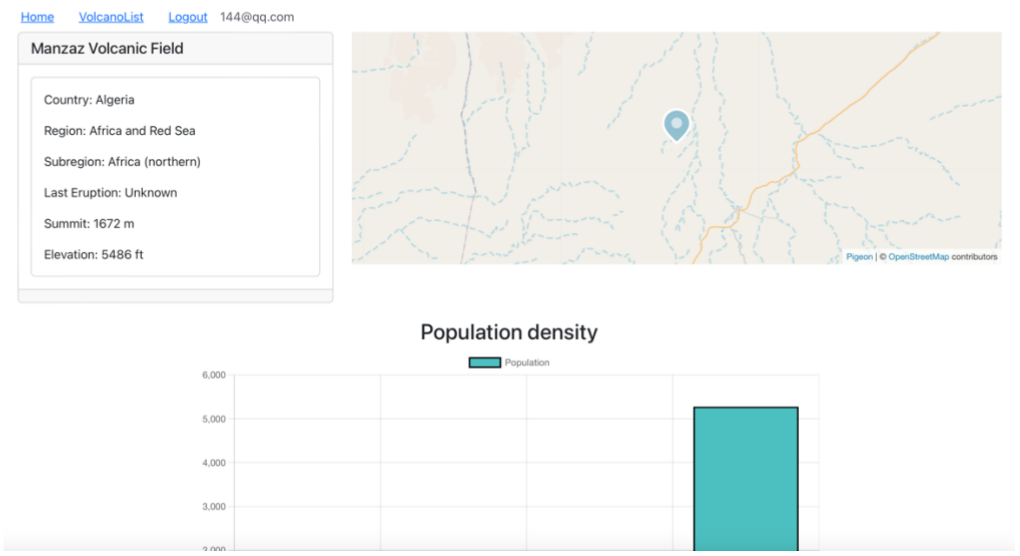


Country: Canada Populated Within: 30km search

| Name | Region | Subregion |
|---|---|---|
| Bridge River Cones | Canada and Western USA | Canada |
| Crow Lagoon | Canada and Western USA | Canada |
| Atlin Volcanic Field | Canada and Western USA | Canada |
| Alligator Lake | Canada and Western USA | Canada |
| Garibaldi | Canada and Western USA | Canada |
| Garibaldi Lake | Canada and Western USA | Canada |
| Fort Selkirk | Canada and Western USA | Canada |
| Nazko | Canada and Western USA | Canada |
| Meager | Canada and Western USA | Canada |
| Milbanke Sound Group | Canada and Western USA | Canada |
| Silverthrone | Canada and Western USA | Canada |
| Tseax River Cone | Canada and Western USA | Canada |

Page Size: 100 1 to 13 of 13 Page 1 of 1

(pic 07)



Manzaz Volcanic Field

Country: Algeria

Region: Africa and Red Sea

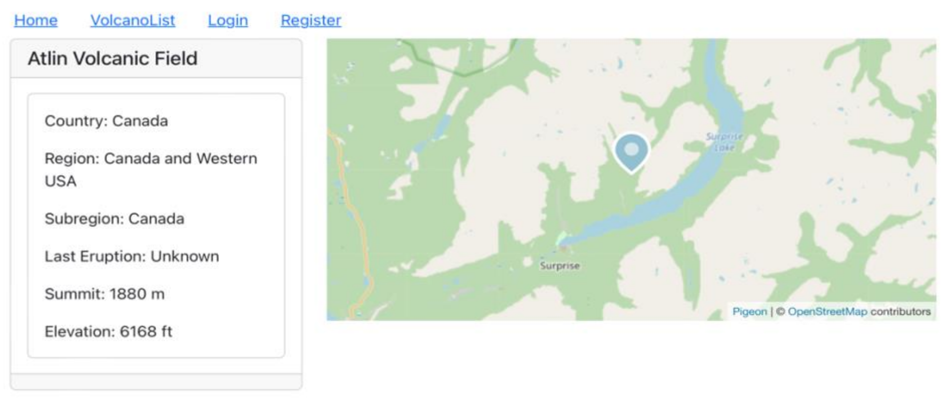
Subregion: Africa (northern)

Last Eruption: Unknown

Summit: 1672 m

Elevation: 5486 ft

(pic 08)



Home VolcanoList Login Register

Atlin Volcanic Field

Country: Canada

Region: Canada and Western USA

Subregion: Canada

Last Eruption: Unknown

Summit: 1880 m

Elevation: 6168 ft

(pic 09)