

2024

# CAB230 Assignment 2 Client Side

Replace image with one  
with some relevance to  
your application here



CAB230

Volcano API – Client Side  
Application

Aaron Halangoda

N11285672

11/5/2024

## Contents

Introduction .....	2
Purpose & description.....	2
Completeness and Limitations.....	3
Use of End Points .....	4
/countries.....	4
/volcanoes.....	4
/volcano/{id} .....	5
/user/register .....	5
/user/login .....	6
Modules Used .....	7
Ag-grid-react .....	7
Axios.....	7
Material UI Core.....	7
Material UI X Charts .....	7
Pigeon Maps.....	7
JWT Decode.....	7
Application Design .....	7
Navigation and Layout .....	7
Usability and Quality of Design .....	9
Accessibility .....	9
Technical Description.....	10
Architecture .....	10
Test plan.....	10
Difficulties / Exclusions / unresolved & persistent errors.....	10
User guide .....	14
References .....	17

## Introduction

### Purpose & description

This is written in a high-level professional tone. Tell us in about a paragraph or so what the app is supposed to do. This should be in **your** words, though you should feel *absolutely free* to steal some of it from the assignment specification.

In a second (and maybe third) paragraph, go ahead and tell us what to look for in your app: ***What did you do that was different? Did you do something to provide the user with functionality beyond what was expected? Is there some special set of modules that you have used that make it look great? Is there some other module that you have used that makes it more efficient?*** At this point, this description is at a very high level still. You will list your modules below.

At this point you can show 1-2 basic screenshots of your application to illustrate the approach, but leave the more detailed screenshotting to the use cases below.

The purpose of the application is to educate people about and display volcanoes all over the world. The project aims to deliver information on each volcano, such as when it erupted. By doing so, the application serves as an educational one where the user can browse different countries and what volcanoes reside within each. The application can provide an experience in viewing volcanos and registering and logging in users using API calls. The application is supposed to provide other information regarding volcanos and their whereabouts, such as how many people live within a certain vicinity of the volcano.

When looking at the application, there are a few things aside from the standard features, such as a 404 page that has been included that goes beyond the scope of what was required. From different functionalities to modules being used to deliver an overall clean and professional look of the application, quite a few components go beyond. Such components range from UI to interactive features. In regards to interactive functionality, added features that went beyond what was expected include a search functionality that takes the user directly to the volcano, gets the user's location and displays volcanos that are near them, a history of what the user has looked at previously and a comparison of volcanos. Regarding UI features and interactiveness, components such as alert pop-ups have been included, image carousels, show password buttons and personalised messages for the user on the home screens with their name displayed. Modules have been utilised to help improve the overall design and look of the application. Most notably, Material UI was used to accomplish this. This module allowed more efficient code as it already has the functionality included, and it was just a matter of implementing it appropriately within the application.

## Completeness and Limitations

The application's purpose is to display volcanoes around the world interactively. Features include a login and registration page, a volcano display page and a list of volcanoes, which are all dynamically displayed, especially regarding the table and volcano map. Regarding what works and what doesn't, all necessary features work to the expectation. All login and registration functionalities have been adhered to and work as expected. Listing the volcanoes and displaying the volcano information is the same. Information is restricted unless the user logs in and the session expiry time is appropriately and correctly implemented. There are no faults with the overall functionality. Information is correctly implemented in the table using ag-grid and displays all relevant information, especially when introducing filters, sorting and radius parameters.

However, there is one key issue that does affect the overall UI and functionality. This is the background image. When the user gets the URL wrong, the image disappears and returns only if the page is refreshed. The solution is known but has yet to be implemented due to time constraints and routing implementation. Another functionality that needs to be fixed is the chart that displays the density of the population. But the information is still there when you hover over each bar. Another flaw is using local storage to hold the user's email; this was implemented so it could be used for the personalised greeting; however, using other implementations, such as useContext, would be much more appropriate in this setting. Finally, some design choices, such as a consistent layout and colour choice, could be better implemented to make the application more professional.

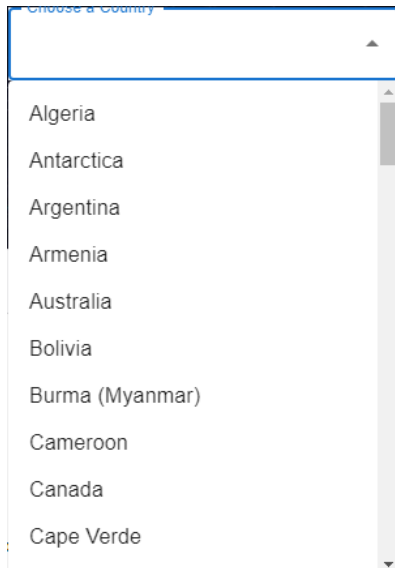
Based on the robustness and the overall functionality of the implemented features, the application closely aligns with the criteria for a grade between 6 and 7, with more leaning towards grade 7. This grade was chosen as the application as all the navigation is handled through React Router, especially regarding the navigation between the Volcano list page and the Volcano data page. In addition, forms have been utilised for inputs, especially on the Login and Register pages. All chosen functionalities display the correct and relevant information, ensuring the user will see the most up-to-date information. Regarding the website's design, apart from some colour choices and minor layout, the application's overall design is thorough, has appropriate colouring and is very easy to use. This includes highlighting to the user what page they are on by highlighting the navigation bar and directing them to press certain buttons or search items through prompts. The map component has been added successfully to the application, allowing it to change the location of the volcano chosen dynamically. The bar graph shows no delays in generating the map or the population data. All features are dynamic, with little to no wait or slow execution times. The calls to API are only generated when needed and are not called all at once. This will benefit when implementing a server-side as all the information will be loaded on demand, minimizing traffic and improving efficiency.

As seen above, combining the advanced functionalities and the application's overall design justifies the claim that it meets the grade of 7.

## Use of End Points

### [/countries](#)

This endpoint displayed all the countries with at least one volcano within their territory. This api endpoint was used when retrieving all countries to display it to the user. This is used when the user selects the volcano they want to see. Another instance is when the user views comparisons of volcanos.



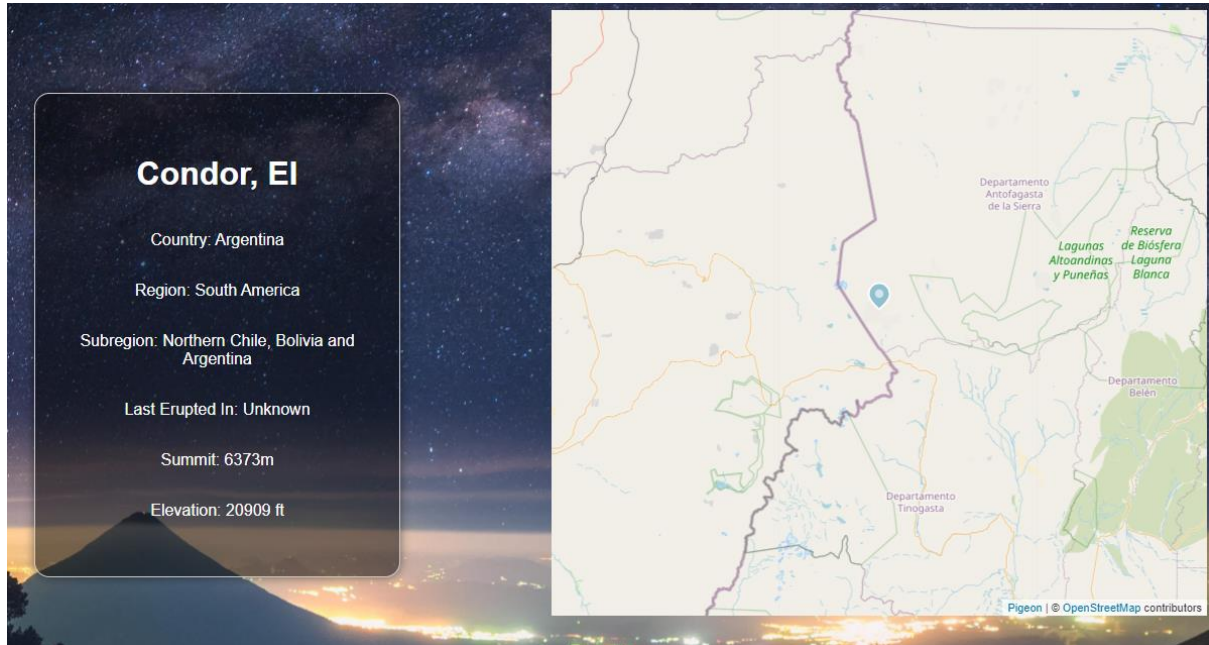
### [/volcanoes](#)

This endpoint was utilised when showing the user a list of associated volcanoes in the country that they have selected. The data is fetched from the endpoint and displays all the volcanoes related to the chosen country. It's used in this section after a user selects a country. It then populates the table to show the volcanoes in said country.

Volcano Name
Antofagasta Volcanic Field
Crater Basalt Volcanic Field
Aracar
Atuel, Caldera del
Condor, El
Blanca, Laguna
Blanco, Cerro
Infiernillo
Huanquihue Group

</volcano/{id}>

The volcano Id endpoint accesses detailed information about a specific volcano based on its ID number. This includes location, population, eruption history, regions, summits and elevations. This is used on individual volcano pages. When the user clicks on a Volcano from a provided list, it will redirect the user to a separate page to get the relevant information from this endpoint. Displaying all of the necessary information to the user

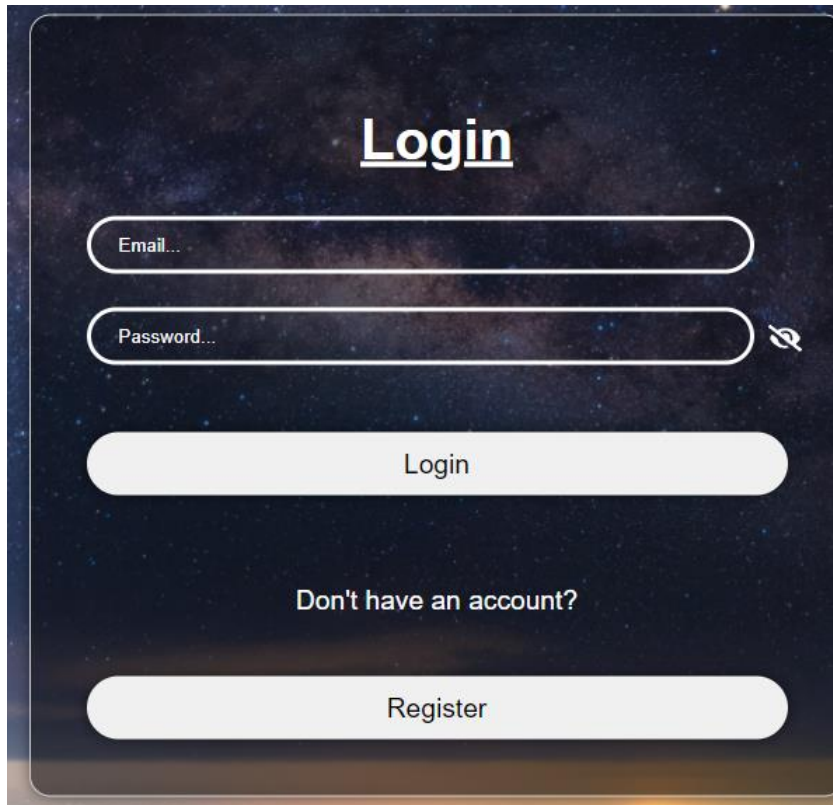


</user/register>

This endpoint handles new user registrations. It accepts the user's details and registers them, allowing for user authentication. It is used within the registration form, where it requests users to input the relevant information to create a new account. This step is as important as the login form, as it is crucial in giving the user a personalised experience and full access to the application. This form can be seen below.


</user/login>

This endpoint is one of the most necessary and vital endpoints as it allows users to access specific information and functionality that requires a user to log in. This endpoint facilitates users by verifying the credentials against stored data. This endpoint is used within the login form, where users who have just registered or are returning are able to submit their credentials to access their accounts. Upon success, the user is created with a personalised message on the home page.



**Login**

Email...

Password... 

Login

Don't have an account?

Register



## Modules Used

This is just a list of the external modules that you have used. You need not specify core React modules. In each case, ***we want the name, a brief description, and a link to the docs at npm or github or wherever.*** The example is `ag-grid-react` as most people will be using this. Just copy that style and add more as necessary.

The following modules were utilized to help the flow of the application and the UI elements. The use of these modules were used throughout the application to help the layout and the functionality. The following modules include:

### *Ag-grid-react*

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

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

### *Axios*

Axios is a module that supports promises. It is promise based HTTP client for the browser.

<https://www.npmjs.com/package/axios>

### *Material UI Core*

This module uses Googles Material Design and holds prebuilt components ready to use such as autocomplete, toggles and dropdown selections.

<https://mui.com/material-ui/getting-started/installation/>

### *Material UI X Charts*

This module provides a visualized graphs and charts such as bar graphs. Being used for things such as population data.

<https://mui.com/x/react-charts/getting-started/>

### *Pigeon Maps*

Module used to generate a map of the world and place a marker at a specified location

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

### *JWT Decode*

The JWT Decode module provides the ability that decodes JWT Tokens which are base64 encoded. Once decoded it reveals the decoded message/instructions

<https://www.npmjs.com/package/jwt-decode>

## Application Design

### Navigation and Layout

When designing the application for volcanoes, several considerations were taken, especially concerning how the user would navigate the website and how it would flow between the screens. The application aimed to create an intuitive and accessible application that met the users needs. During the design phase a critical process that was taken into consideration is how to make the application easy to use yet remain informative, that was the approach that was taken. With that mindset, the pages and layout of the application were done with minimalism to ensure that the user could see what



they wanted and when to see it. Although no mock designs or wireframes were created the application went through different iterations of layout designs and navigation choices.

Alternative design choices included different button styling and adapting a search bar in the middle of the home page. A sidebar navigation was considered initially; however, after testing, it was determined that a top navigation bar was more intuitive and more suited for quick access.

In regards to the choices made for the navigation the key choice was to create a navigation bar which will allow the user to interact and navigate them to the page of their choice. The navigation bar includes, Home, List of Volcanoes, Login and Register. This was chosen to mitigate the amount of information on the page and allow clear, visible instructions on where to go. In regards to the flow between screens, each of the buttons take the user directly to the desired page which ensures a logical transition between the screens. With each interactive button leading the user directly to their desired outcome with no falter or delay when switching screens. The menu items used, as mentioned, were mainly clickable links in the nav bar and buttons on the home screen that will take the user to the necessary pages. The overall layout that was chosen for the application was primarily a column layout to ensure that the information went from left to right with information towards the right handside. Another layout design choice was a flexbox layout. It allowed elements to be stacked in a single column and the information within containers, making it more visually appealing and providing a straightforward journey throughout the content.

## Usability and Quality of Design

### Accessibility

The application satisfactorily meets this criterion regarding text equivalents as all the images used within the application have appropriate alt texts to indicate and display what the image is and why it is used. Images are the only non-text elements used throughout the application. The closest non-text elements would be the map and graph elements that were used to show the specific volcano data. However, texts have been used within those elements to distinguish them.

When evaluating the application against information conveyed in color and readable documents without style sheets, the application partially meets this accessibility requirement. If the application is loaded without any CSS associated with it, then the text, buttons, search bars, and other elements will be rendered but have poor layout structure and design. Due to the application using HTML tags such as inputs and buttons, some of the elements would be rendered in; however, some would not render in as it relies on CSS properties to be operational.

Regarding to dynamic content updates the application does meet this accessibility requirement as it all elements such as the map, chart, and personalised homepage all get dynamically updated as soon as changes occur. This includes a change in volcanoes, users or population density. The same goes for the alerts that update the alert status based on the scenario. Such examples include if the user gets an error alert and then gets the forms filled correctly a success alert is shown straight away. Due to limited text equivalents, such as a description for the map this criterion, there is no way to determine if it is dynamically updated. As such, the design of the application not only helps but ticks off this accessibility criteria.

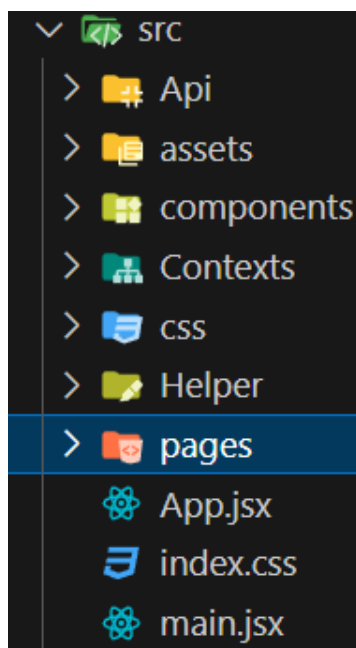
Regarding correctly identified and appropriate table use, the application does help in meeting this criterion as the table used for the volcano list is appropriately laid out. The ag-grid module correctly implements the elements and titles, ensuring that the row and column headers are distinguishable from one another. As such, the application, thanks to the use of the ag-grid module, satisfies this accessibility requirement.

## Technical Description

### Architecture

When looking into the src directory, each folder represents different functions and responsibilities for the application. The control flow is primarily managed through the App.jsx and Main.jsx, with the App being the root component and the Main being the entry point that initialises the react application. Services related to the API are handled within the `Api` directory, which encapsulates all API calls. This service setup separates the calling logic from user interaction to ensure maintainability and modularity. The `assets` folder has the images or videos used for the application. The `components` folder holds code for components like navigation bars or footers. The `CSS` folder holds all of the individual styling for the pages; the `helper` folder contains code that makes it easier to reuse, such as the navigation to pages. The `Contexts` folder contains the files relating to the creation of useContext components. Which is then able to be accessed from other files. Finally, the pages folder holds all the necessary code to operationalise the pages.

These choices made the code reusable and readable instead of having everything in one file. Separating the code into relevant folders helps people better understand what the code is doing in different application areas, improving modularity, maintainability, and scalability.



### Test plan

#### Difficulties / Exclusions / unresolved & persistent errors /

In developing the React application, several challenges pushed the technical skills needed. These challenges were roadblocks encountered throughout production and bugs in the final output. Roadblocks encountered throughout production include validating users, registering and

authenticating them, ensuring the user is logged in throughout the application, the session expiration time, and the background image occasionally disappearing. The first roadblock issues included that any email or password that the user inputs would get accepted and that they could not log in with the email once they registered. This was a significant roadblock as the users weren't able to sign in to view information which would result in the user having to re-register to log in again. This roadblock was resolved through Regex and a form validation that checks the input details before proceeding with the registration and login.

The second roadblock posed as an issue as there were multiple times in which the user's details would not be authenticated throughout the site. At first to solve this issue, local storage was used to store the details of the user each time and then when the information was needed it was grabbed from the local storage. This roadblock was then resolved by using React's `useContext`, which allows the data to be accessed within all components.

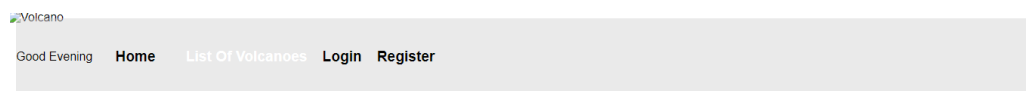
The third major roadblock was the session expiration time. This roadblock prevented the user from automatically being signed out as there were errors in implementing the countdown. Attempts were made to solve this roadblock by initiating an incremental method in which once the number finally got to 86400 after each second it would sign the user out. Although this method helped with the roadblock, it would keep failing. Ultimately the roadblock was resolved by using the JWT Decode module which helped determine the difference between the times and once it reached that specific time it would sign the user out. Although this was not a significant roadblock, it did push back the time required to spend on other components.

Finally, the last roadblock was only partially resolved, as this bug still affects one page. The roadblock that was faced was the design of the background, more specifically the background image. The roadblock that was found was that each time the user goes to the pages and accidentally misspells or gets the URL wrong, it would cause the background image to break and disappear causing all the text that was used invisible. The only way to fix this roadblock was to reinitiate the React application. This bug was later found and fixed as it was an issue with the router and how it would route to the pages. As such, appropriate action was taken to solve this roadblock. However, the bug persists within the page with the endpoint `/volcanoes/{id}`. The bug still resides in there as there was no way to fix the route to that URL without changing or redoing the code for the page. This was due to the ID being grabbed from the URL, and if the URL changed within the route component, then there was no way to grab the ID to display the specific volcano data.

Everything has been completed in terms of functionality except the functionality exceeding expectations. Due to time constraints, the volcano comparison and the volcanoes near the user could not be implemented in time. However, they can be implemented in future iterations. All other functionality, such as login, logout, and volcano lists, has been implemented successfully.

In regards to bugs, one major bug that still remains is the background image. The bug included that when the user gets the URL wrong on a specific page, the background image will disappear, and the

only way to get it back is to refresh the page when the user is on a different component. As seen in the image once the url is incorrect the image defaults into the alt text and displays the broken image symbol. Another bug still remains is the population density chart, this bug occurs when the population exceeds a specific threshold of 1 million people. When this threshold is met, the numbers on the side of the chart get cut off, making it unreadable and unable to know the population within a certain radius. The only way is to hover over each bar. Another bug is that in certain volcanoes the x-axis information is invisible and does not display the label of the population density, which can be seen below. Another bug is minor, but when the user is logged in, the previous volcano history is stored. If the user logs out then goes back in they can still see the history, however, if the page is refreshed the history is deleted. This is not what is meant to occur but rather when the user logs out and then back in there should be no history of previously viewed volcanoes.

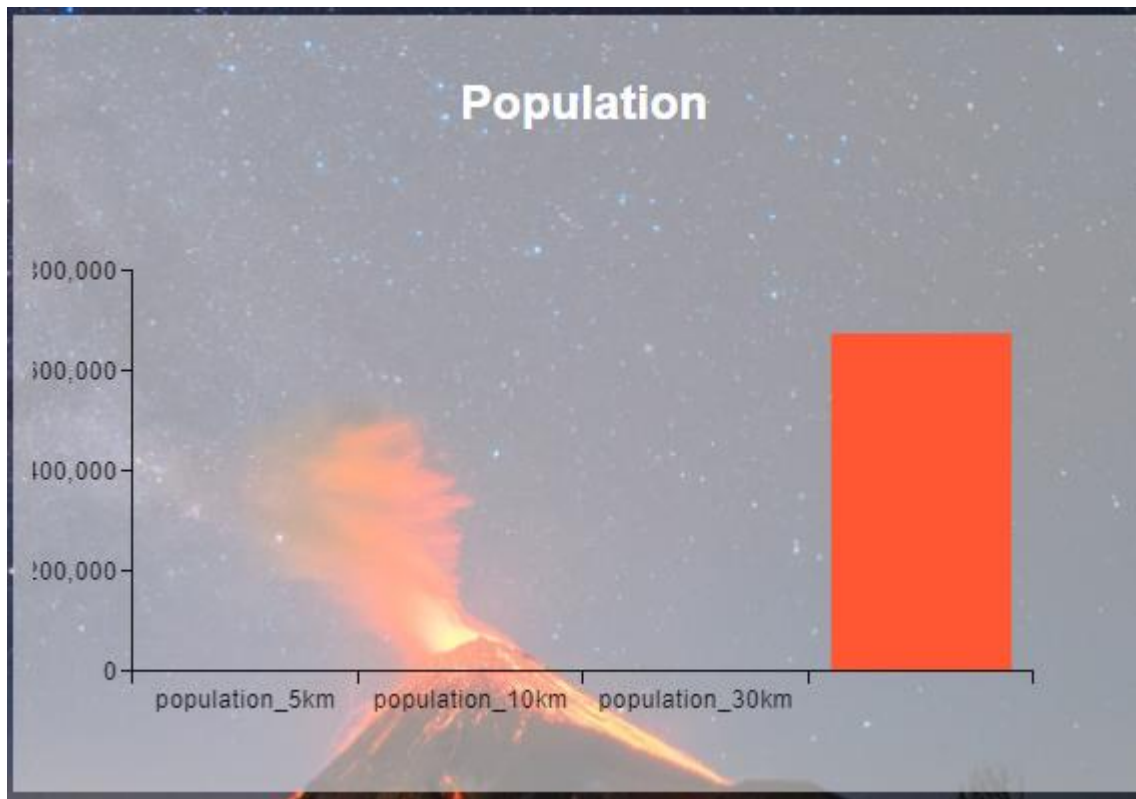


Background image after incorrect url inputted

```
<div className="image-container">
  <img src='src/assets/night-scenery-night-stars-volcano-mountain-landcape-uhdpaper.com-4K-
    className="background-image" />

  <div className="overlay">
    <Navbar />
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/volcanoes" element={<VolcanoList />} />
      <Route path="/login" element={<Login />} />
      <Route path="/register" element={<Register />} />
      <Route path="/volcanoes/:id" element={<VolcanoPageData />} />
      <Route path="/404" element={<ErrorPage />} />
      <Route path="*" exact={true} element={<Navigate replace to="/404" />} />
    </Routes>
  </div>
</div>
```

Code for the URL Bug



Population density bug

```
div className='volcano-pop'>
  <h2>Population</h2>
  {volcano && (
    <BarChart
      xAxis={([
        scaleType: 'band',
        data: ['population_5km', 'population_10km', 'population_30km', 'population_100km']
      ])} //set the x axis
      series={([
        data:
          [volcano.population_5km, volcano.population_10km, volcano.population_30km, volcano.population_100km]
      ])}
      width={550}
      height={300}
      colors={['#FF5733']}
    />
  )}
</div>
```

Code for the population density bug

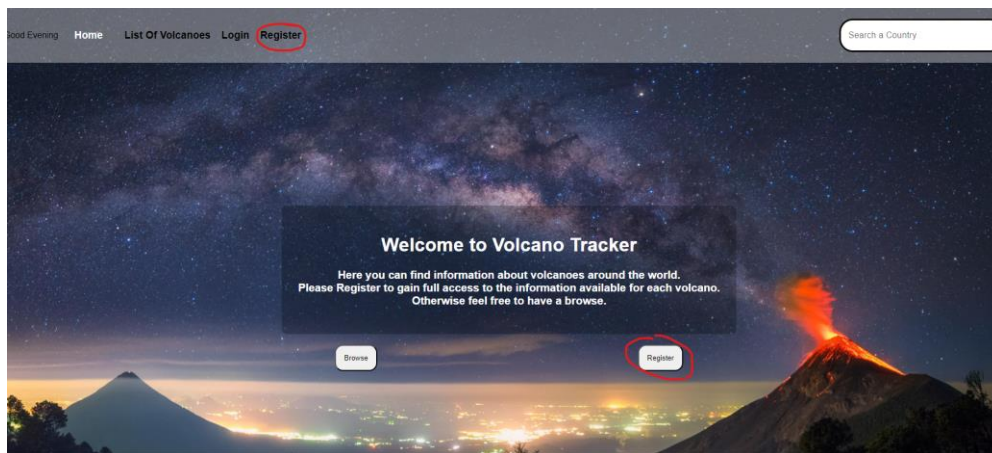
```
const logout = () => {
  clearTimeout()
  setLoginUser(null)
  setIsAuthenticated(false)
  localStorage.removeItem('loginUser') //Th
  localStorage.removeItem('email') //This w
  localStorage.removeItem('volcanoHistory')
```

Code for the volcano history bug

## User guide

Tell us how to use your application

Use screenshots liberally here. You may re-use screenshots from above.



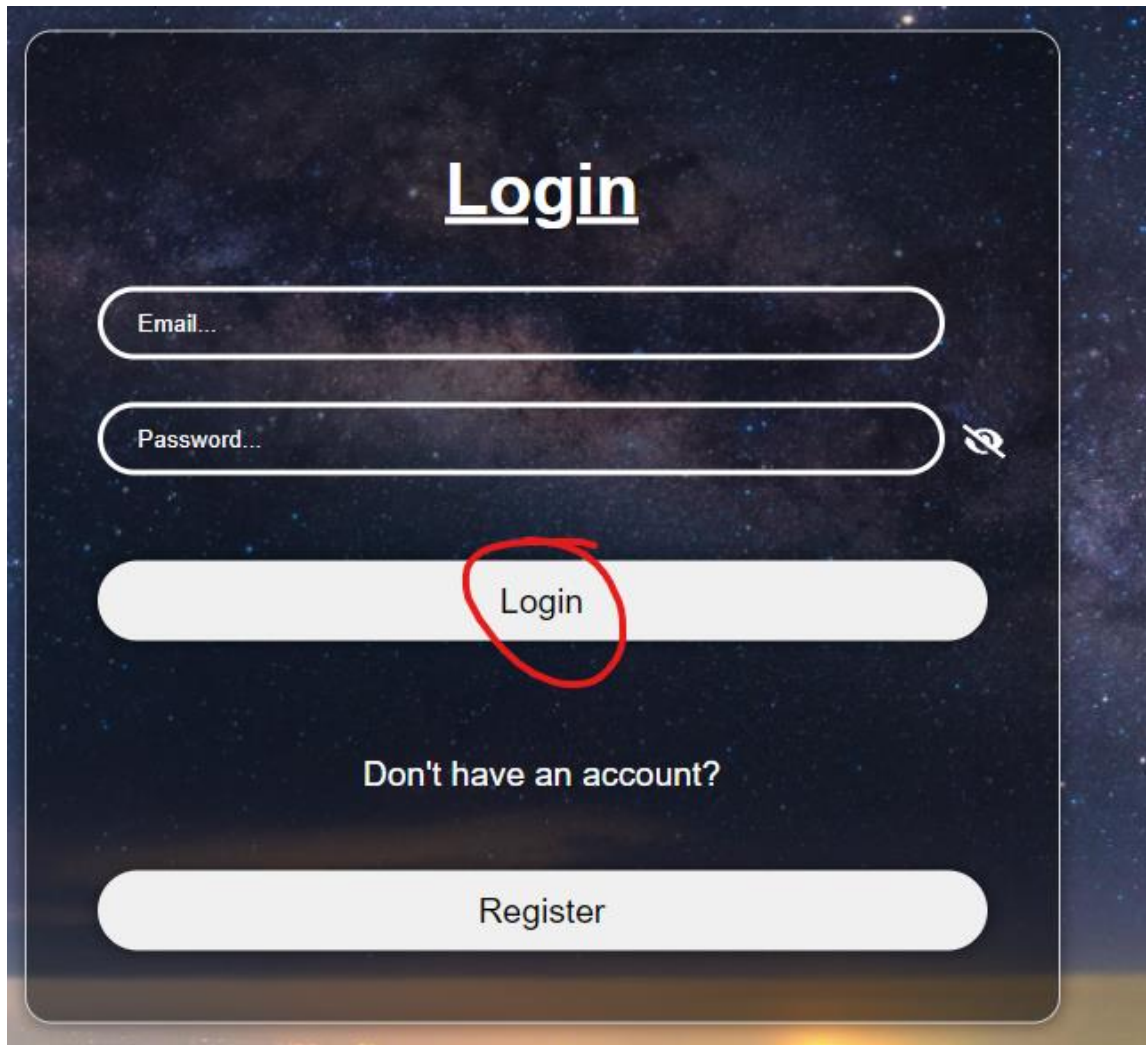
To use the application begin from the homepage. From there you can select whichever button, link or search bar to take you to the next page.

A screenshot of the Register form. The form is titled "Register" and has three input fields: "Email...", "Password..." (with an eye icon), and "Confirm Password..." (with an eye icon). Below these fields is a "Register" button, which is circled in red. Underneath the button is the text "Already have an account?". At the bottom of the form is a "Login" button, also circled in red. The background of the form is a dark, starry sky.

In this instance the user hits the register button, it will then take them to the registration form page which will require the user to sign up. if they already have an account they can press the login button at the nav bar or within the registration

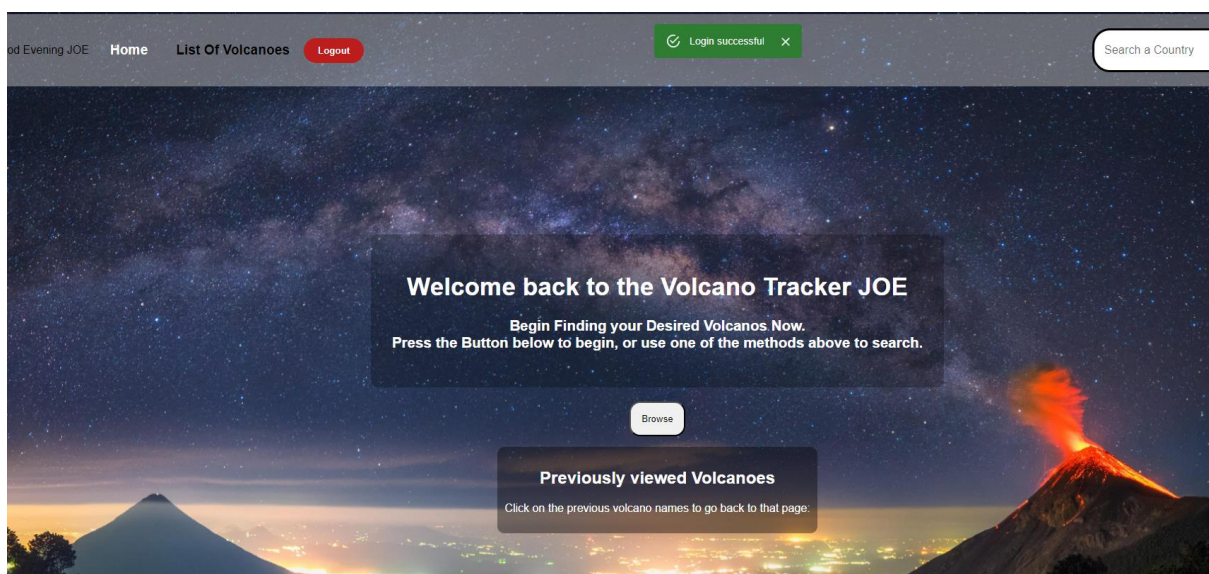


form.



The login form is centered on a dark background with a starry space theme. It features a large, bold, underlined title "Login" at the top. Below the title are two input fields: "Email..." and "Password...". The "Password..." field has a small eye icon to its right. Below these fields is a large, light gray button labeled "Login", which is circled in red. Underneath the button is the text "Don't have an account?". At the bottom of the form is another large, light gray button labeled "Register".

Once the user is registered, it takes them to the login page, where they are prompted to log in. Once done it then takes them back to the home page which is now changed for personalisation.



The user can then select either the explore button or the navigation button. Once selected, it will take the user to the list of volcanoes page.

Please search a country to choose a volcano from:

Population Density Radius:

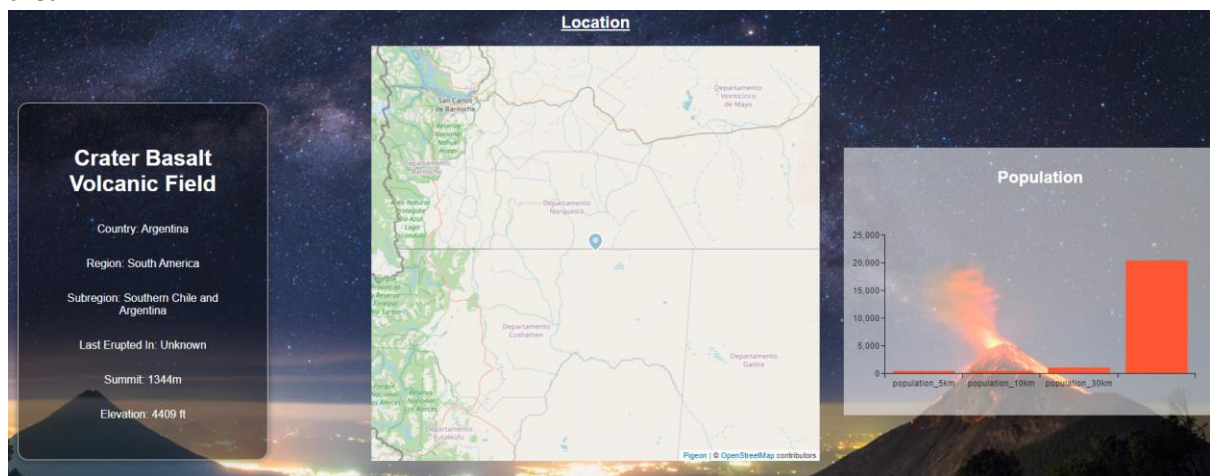
Choose a Country

None

Volcano Name	Region	Subregion
--------------	--------	-----------

No data is being displayed at the moment, have you selected a country? If you have and there is still no data, then there is no data to display. Maybe try a different country or radius.

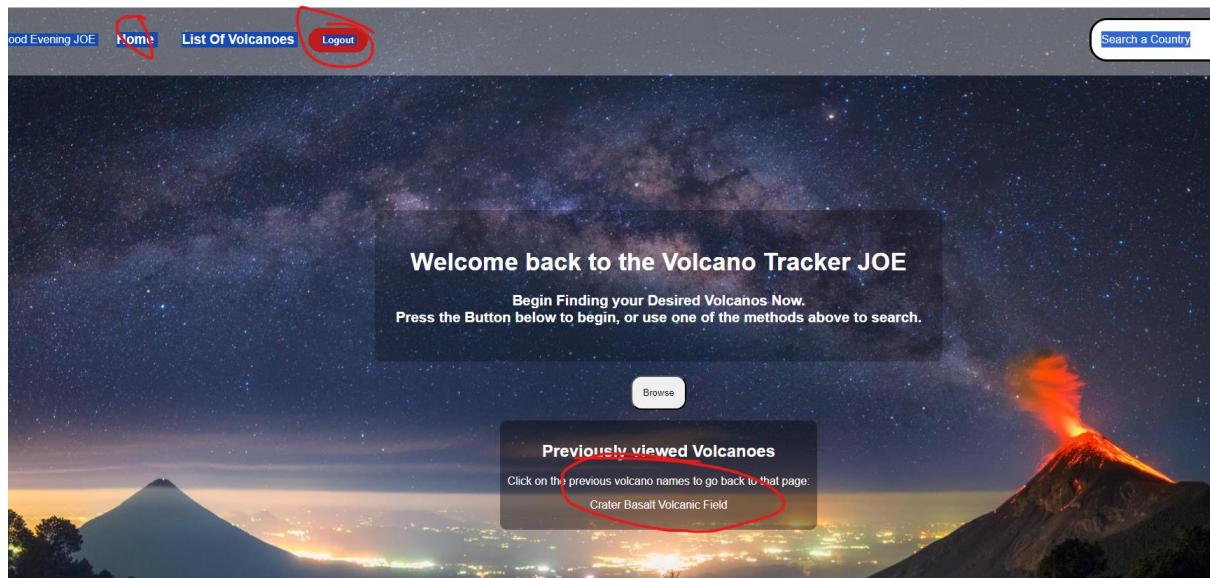
Once on the list of volcanoes page, the user can select a country and radius from the drop-down menu. The page will then display volcanoes in that area.



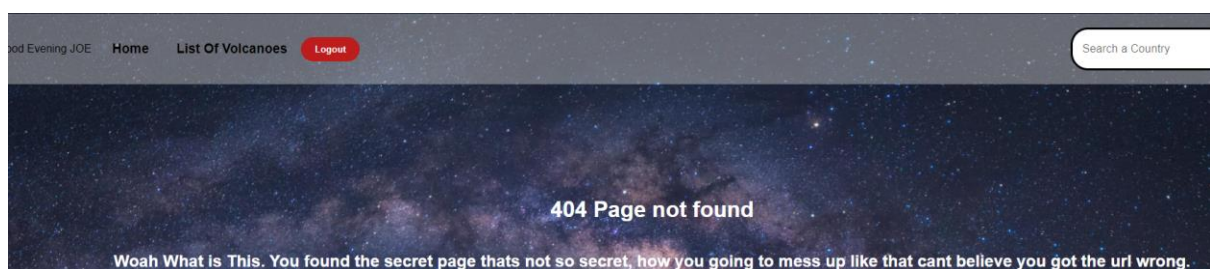
Once a user selects a volcano, they will be redirected to the volcano page, which shows its data. If they are logged in, they can see the population



density.



The user can then return home and see the volcano they just saw earlier. After logging out, the user will be taken back to the login page.



Any errors in the URL will take the user to the error page.

## References

AG grid: High-performance react grid, angular grid, JavaScript grid. AG Grid: High-Performance React Grid, Angular Grid, JavaScript Grid. (n.d.). <https://www.ag-grid.com/>

Axios. npm. (n.d.-a). <https://www.npmjs.com/package/axios>

Dros, A. (n.d.). How I Plan My Landscape Photos for the Highest Chances of Success. Retrieved from <https://petapixel.com/2017/08/28/plan-landscape-photos-highest-chances-success/>.

Ijay.igboagu. (2023, December 14). How to fetch API data in react. freeCodeCamp.org. <https://www.freecodecamp.org/news/how-to-fetch-api-data-in-react/>

Installation. Material UI. (n.d.). <https://mui.com/material-ui/getting-started/installation/>

JWT-decode. npm. (n.d.-b). <https://www.npmjs.com/package/jwt-decode>

Limbad, D. (2021, December 28). Password visibility toggle component in Reactjs/Nextjs using typescript. Medium. <https://dhrumilimbad.medium.com/password-visibility-toggle-component-in-reactjs-nextjs-using-typescript-ecec9249568d>

MUI . (n.d.). Charts - getting started. MUI X. <https://mui.com/x/react-charts/getting-started/>

Pendleton, H. (2021, September 15). USEAUTH. Medium.  
<https://hhpendleton.medium.com/useauth-265512bbde3c>

Pigeon Maps. (n.d.). Installation. Pigeon Maps. <https://pigeon-maps.js.org/docs/installation/>

West, K. (2023, August 14). User authentication made easy: Usecontext. Medium.  
<https://medium.com/@katherinewest285/user-authentication-made-easy-usecontext-cc79b5a3ce82>