Rona Rosal

N1036038



Replace image with one with some relevance to your application here

CAB230

Volcano API – Client Side Application

Contents

[Introduction 2](#_Toc68899925)

[Purpose & description 2](#_Toc68899926)

[Completeness and Limitations 3](#_Toc68899927)

[Use of End Points 4](#_Toc68899928)

[/countries **Error! Bookmark not defined.**](#_Toc68899929)

[/volcanoes 4](#_Toc68899930)

[/volcano/{id} 5](#_Toc68899931)

[/user/register 5](#_Toc68899932)

[/user/login 7](#_Toc68899933)

[Modules Used 7](#_Toc68899934)

[Ag-grid-react 7](#_Toc68899935)

[Boot-Strap **Error! Bookmark not defined.**](#_Toc68899936)

[Pigeon Maps **Error! Bookmark not defined.**](#_Toc68899937)

[Application Design 8](#_Toc68899938)

[Navigation and Layout 8](#_Toc68899939)

[Usability and Quality of Design 9](#_Toc68899940)

[Accessibility 9](#_Toc68899941)

[Technical Description 10](#_Toc68899942)

[Architecture 10](#_Toc68899943)

[Test plan 12](#_Toc68899944)

[Difficulties / Exclusions / unresolved & persistent errors 13](#_Toc68899945)

[Extensions (Optional) **Error! Bookmark not defined.**](#_Toc68899946)

[User guide 13](#_Toc68899947)

[References 14](#_Toc68899948)

[Appendices as you require them 14](#_Toc68899949)

## Introduction

### Purpose & description

The volcano application is designed to allow clients to interact and analyze data from a REST API.

It allows users to navigate between pages and query for information about a Volcano that is publicly available, such as, the country ID, Volcano name, Country, Region, and Subregion, last eruption, summit, elevation, longitude, and latitude which will be display on a table format.

However, there are also data’s that are not available to un-authorized users. To access this data, a user must be registered in the system and must be logged in. Some of the data’s that cannot be accessed publicly and requires a log-in is the population density.

Moreover, the application also allows a user to register as a new client or log-in if he/she already has an account.

In addition, some of the successful implementations for the project was the successful fetches of all API end points, clean formatting and navigation of the pages, individual pages where the authenticated data is displayed and an attempt to implement the map feature. Moreover, user registration and log-in features where also implemented and allows a promise to be displayed on screen for user references.

Photo: Landing Page with buttons for clean navigation.

A picture containing text, sky, outdoor, sunset

Description automatically generated

### Completeness and Limitations

* Successful and clean implementation of the landing page, where users can navigate and choose to access pages such as the volcano list, registration page, or log-in page using a button component.
* Successful implementation/fetching of /countries endpoints, the fetch returned a list of countries that is listed in a drop-down menu for users to choose from. However, the drop-down menu fails to communicate with the grid table.
* Successful implementation/fetching of the /volcanoes endpoint, the fetched data is displayed on a Grid table in the volcano list page. In addition, it allows the user to sort and filter information on the table. However, the country query is hardcoded. In addition, the use of pagination is also implemented on the table.
* Successful implementation/fetching of volcano/{id} endpoint, the fetched data is displayed on the individual volcano page when a user clicks the “Fetch Data” button. The data uses parameters to get the “ID” information from the table to input to /volcano/{id} endpoint so the queried information is accurate.
* Successful implementation/fetching of /user/register and user/login endpoints, users are able register and login by providing an email and password in an input component,

the component will the send a request to the API and return a promise.

For the user/register endpoint If the user registers and already exists, a message will prompt that a user already exist, on the other hand if a user is not registered yet the system will create a new user upon registration.

For the /user/login endpoint, if the user exists in the system as a registered user, it will fetch the authenticated API and made it available for the user.

* There is also an attempt to implement the map feature for the individual volcano page. However, it fails to automatically use the longitude and latitude information of the selected volcano by the user. So, it is hardcoded.
* Errors and prompt messages are displayed for the registration page. It will display “User is created” when the user successfully registers as a new client. On the other hand, it will display “The user already exists” if the client is already registered.
* Some usage of CSS styling for the landing page.

## Use of End Points

*/countries*

Fetches the /countries end point which returns a list of countries and displays it in a Drop-down selection, where the client can select a specific country.

Graphical user interface, text, application

Description automatically generated

#### /volcanoes

Fetches the /volcanoes end point which returns a list of object of volcanoes associated with the country selected by the client.

A screenshot of a computer

Description automatically generated with medium confidence

#### 

#### /volcano/{id}

Fetches the /volcano/{id} end point and returns more information about the queried volcano such as its last eruption, summit, elevation, longitude, and latitude. Moreover, if the client is authenticated user the information also shows the population density of the area.

Map

Description automatically generated

#### /user/register

Fetches the /user/register end point and allows user to post information to the API to register a user using the input and button component.

Graphical user interface, application

Description automatically generated

The registration page returns a message which is display for the user. It lets the user know if the information the user provided in the email and password in the input component is already registered. Returning an error message and “user already exist” text.

Graphical user interface, application

Description automatically generated

The registration page returns a message which is display for the user. It lets the user know if the information the user provided in the input component successfully created a new user. Returning a message which states “User Created”.

Graphical user interface, text, application

Description automatically generated

#### /user/login

The log-in pages allows a user to input an email address and password. If the user trying to log-in is already registered, the user can access authenticated data in the individual volcano list page. On the other hand, the promise return would “user does not exist”.

Graphical user interface, application

Description automatically generated

## Modules Used

#### 

#### Ag-grid-react

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

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

#### Boot-strap

Module to provide styles and themes to components such as the button that is used in my application. Moreover, it also includes the “container” feature used in <div> to align or position components in a neat way.

<https://react-bootstrap.github.io/getting-started/introduction/>

#### react-router

Module to provide functions in the application such as navigation and the use of URL parameters. Moreover, can also be used to implement nested routes.

<https://reactrouter.com/docs/en/v6/getting-started/overview>

#### Pigeon Maps

Module used to provide the implementation of maps in the application, some features include a visual display of the location given the latitude and latitude. In addition, a marker is also provided.

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

## Application Design

### Navigation and Layout

The navigation and layout will initially begin in the Home/Landing page. This page has 3 buttons where users can navigate to a specific page such as “Volcano List,” “Register”, “Log-In”.

***Home page:*** Consists of the buttons to use for navigation, a background for some basic styling, and icons display at the bottom of the page to provide basic information of watch the user can expect from the application.

***Volcano List Page:*** Consist of the drop-down menu so that the user can choose a specific country. Moreover, this is also where the table is located where the user can see the list of volcanoes in a specific country.

In addition, this page also consists of a back button so that the user can navigate back to the home page. Through this page the client can access the individual volcano page where the authenticated data and maps can be found. When the use clicks on a specific volcano they are directed to the individual page.

**Register and Log-In Page:** These pages basically allow the user to input details to register or log-in.

*Mock-up design/ Sketches:*

Diagram

Description automatically generated

### Usability and Quality of Design

To start, the overall layout out and design of the application is clean and easy to work around with. Each page is easy to navigate, and components used are well suited for the data that they hold. Moreover, the pages are designed to accommodate even someone who does not work around with computer a lot. The design is simple, not too many colors were used and there are no fonts that is not easy on the eyes.

In terms of usability, the application works seamlessly in terms of navigation., fetching and display of data. It also allows clients to register as a new user or log-in as an existing user.

On the other hand, there are a few clumsy components that do not function as expected. One example for this is, when a user chooses a specific country, it does not automatically update the table information due to the drop-down component failed to communicate with the table component. This can be improved by taking the value of the dropdown component when the value changes and use that value to query the /countries endpoint.

Moreover, the map function does not work as expected as well, the longitude and latitude are hardcoded and does not automatically update when a new volcano is selected.

In addition, the log-in feature does not display a prompt for the user whether they have successfully logged-in.

### Accessibility

* Provide a text equivalent for every non-text element – alternatives to images, symbols, scripts, graphical buttons, sounds, audio and video files and so on.
* No, there can be no text equivalent to non-text elements such as images, symbols, scripts, graphical buttons etc.
* Ensure that all information conveyed with color is also available without color, for example from context or markup.
* Yes, information such as text can be presented as plain or can be edited into different fonts and colors.
* 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.
* Yes, even without the use of CSS, HTML can still be used to organized documents and be readable.
* Ensure that text equivalents are updated when dynamic content changes.
* Yes, this can be done through the use of useState. Where it captures the on change value of the component and takes the new value.
* Avoid causing the screen to flicker.
* No.
* Use the clearest and simplest language appropriate for a site's content.
* Yes, Reactjs can be one of the simplest language to use for a site content.
* For tables, identify row and column headers – clearly differentiated from the data. Technical Description
* When differentiating Headers from the data, the first letter for a header is initially written in an uppercase format. Whereas data stored in rows are written in lowercase format.

### Architecture

The main codes needed for the application is stored in “src” folder while the images used in the application is stored in the “public” folder.

Graphical user interface, application

Description automatically generated

The “components” folder holds the .jsx files customListDropdown, footer, header, and nav which is used to style and hold data for the other pages in the application. These .jsx are imported to the pages where they are used.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

customListDropDown is imported in “VolcanoListPage”, while Footer, Header, and Nav is imported in the “Home” page.

Graphical user interface, application, Word

Description automatically generated

The “pages” folder holds the codes of each pages the user will navigate into.

The App.js will initially call the “Home.jsx” to display the landing page of the application the VolcanoListPage.js, RegisterPage.js, and LogInPage.js is then imported in the Home.jsx so it allows users to use button components to navigate to the respective chosen pages from the three above.

Navigation routes are used in the index.js to support the transition from one page to another.

Text

Description automatically generated

### Test plan

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Expected Outcome | Result | Screenshots (Appendix) |
| Landing Page | Page to store buttons to other pages | PASS | 1 |
| Fetch /countries | Return list of countries | PASS | 2 |
| Fetch /Volcanoes | Return volcanoes w/ data | PASS | 3 |
| Fetch /volcano/{id} | Returns authenticated data | PASS | 4 |
| Register a new user | Returns message “user created” | PASS | 5 |
| Register existing user | Returns message “User already exists” | PASS | 6 |
| Sort volcanoes | Sort volcanoes in table | PASS | 7 |
| Filter volcanoes | Filter volcanoes in table | PASS | 8 |
| Click “Back button” | Navigate to previous pages | PASS | 9 |
| Login User prompt | Display if user is successfully logged-in | FAIL | 10 |
| Locate Volcano (map) | Automatically locate volcano in map when selected | FAIL | 11 |

Difficulties / Exclusions / unresolved & persistent errors /

Difficulties

* Handling state to record dropdown on change value and using it to query the /countries endpoint
* Getting the longitude and latitude of a volcano location and using that to update the map function every time a new volcano is selected (Query is Hardcoded)
* The query for the /volcanoes endpoint is hard coded
* The query for the /countries endpoint is hard coded
* No error handling or message display available to the client for the log-in page

Exclusions

* Population density chart

## 

## User guide

Tell us how to use your application

Step 1: Navigate to volcano list page and explore public data

Step 2: Navigate to Registration page and register as a new user

Step 4: Log-In as a registered user

Step 5: Navigate to volcano list page and explore public and authenticated data

Step 5: Select a specific volcano

Step 6: Click “Fetch Data” button to access authenticated data

## References

Weekly Lectures

Week 6: Working with data tables in React (Tutorial Codes)

Week 7: CAB230 JWT Client Worksheet2 (Tutorial Codes)

Week 7: React Forms (Tutorial Codes)

Week 7: React Search Bar (Tutorial Codes)

Digamber(n.d). *React Bootstrap Custom Dropdown List with Fetch API Tutorial.* Retrieved from <https://www.positronx.io/react-bootstrap-custom-dropdown-list-with-fetch-api-tutorial/?amp&fbclid=IwAR0hJQ4EqQsNS8LfrBtOTPtGoGEO9ev6kJ0FzKbWwMcXvi4_k2_NFKBUxeY>

Pigeon Maps(n.d). *Map.* Retrieved from <https://pigeon-maps.js.org/docs/>

## 

## Appendices as you require them

1A picture containing text, sky, outdoor, sunset

Description automatically generated

2

Graphical user interface, text, application

Description automatically generated

3

A screenshot of a computer

Description automatically generated with medium confidence

4Map

Description automatically generated

5

Graphical user interface, text, application

Description automatically generated

6

Graphical user interface, application

Description automatically generated

7

Graphical user interface, text

Description automatically generated with medium confidence

8

Graphical user interface, text

Description automatically generated

9



10 No prompt if log-in was successful or not.

Graphical user interface, application

Description automatically generated

11 Map does not function as expected (Longitude and latitude is hardcoded)

Map

Description automatically generated