



Client Side Project

2024 05 08

Hajun Song

N9894403

CAB230

INTRODUCTION	3
PURPOSE & DESCRIPTION	3
COMPLETENESS AND LIMITATIONS	3
Use of End Points	4
/countries	4
/volcanoes	4
/volcanoes/{id}	5
/user/login	5
/user/register	6
Module Used	6
AG-Grid	6
Pigeon Map	6
Chart.js	6
ReactDOM	6
Application Design	7
Navigation and Layout	7
Usability and Quality of Design	7
Accessibility	8
Technical Description	9
Architecture	9
Difficulties / Exclusions / unresolved & persistent errors	10
User Guide	11
References	14

INTRODUCTION

PURPOSE & DESCRIPTION

The purpose of this project is to develop a React-based web application that enables client users to interact with a Volcano data API. The objective is to utilize a REST API to allow users to view and analyze data pertaining to volcanoes. By leveraging RESTful services, users can access comprehensive information about volcanoes and engage with the data in a user-friendly interface. Through the development of this client-side project, we aim to implement modern approaches to create a sophisticated application. This project provides valuable experience in querying REST APIs and presenting the results effectively on a web page.

This project aims to create a React-based web application that interfaces with a Volcano data API, enabling users to explore and analyze volcano information dynamically. Unique features include real-time data fetching, interactive visualizations using Chart.js, and a responsive UI design facilitated by React and CSS techniques. Efficiency is ensured through React Query for optimized data fetching and caching, while seamless navigation is enabled by React Router. Overall, these modules enhance user experience, functionality, and performance, making volcano data exploration intuitive and engaging.

COMPLETENESS AND LIMITATIONS

This React app successfully implements user registration and login endpoints, along with authenticated data retrieval. Table components utilize standard functionality provided by AG Grid, minimizing excessive server querying. The design is generally clean and uncluttered, although some areas may be slightly clumsy or confusing. While attempts have been made to implement map and chart components for displaying volcano location and population density data, there may be some shortcomings in their execution. Overall, the app meets the requirements for a grade 5 level, demonstrating basic functionality and navigation while also incorporating some advanced features.

Use of End Points

/countries

Home VolcanoList Login Register

Volcano List

Select a country ▼

From the landing page, when navigating to the Volcano list, there is a country section. Once a country is selected, drop-down menus appear, allowing users to further refine their selection by choosing specific countries.

/volcanoes

Home VolcanoList Login Register

Volcano List

Japan ▼

ID	Name	Country	Region	Subregion
1	Abu	Japan	Japan, Taiwan, Marianas	Honshu
16	Aogashima	Japan	Japan, Taiwan, Marianas	Izu, Volcano, and M
30	Adatarayama	Japan	Japan, Taiwan, Marianas	Honshu
65	Asamayama	Japan	Japan, Taiwan, Marianas	Honshu
68	Aira	Japan	Japan, Taiwan, Marianas	Ryukyu Islands and
75	Akagisan	Japan	Japan, Taiwan, Marianas	Honshu
76	Asosan	Japan	Japan, Taiwan, Marianas	Ryukyu Islands and
78	Akan	Japan	Japan, Taiwan, Marianas	Hokkaido
81	Akandanayama	Japan	Japan, Taiwan, Marianas	Honshu

Page Size: 100 ▼ 1 to 100 of 122 K < Page 1 of 2 > >1

Once a country has been selected from the dropdown menu, a list of volcano information corresponding to the specific country will be displayed. Users have the option to sort the list in ascending or alphabetical order.

/volcanoes/{id}

[Home](#)
[VolcanoList](#)
[Login](#)
[Register](#)

Abu

Country: Japan

Region: Japan, Taiwan, Marianas

Subregion: Honshu

Last Eruption: 6850 BCE


Summit: 641

Elevation: 2103

Latitude: 34.5000

Longitude: 131.6000

[View Population Chart](#)



Pigeon | © OpenStreetMap contributors

When a volcano is selected from the volcano list, users will be navigated to the volcano information page. This page will display the location of the volcano on a map along with other volcano details.

/user/login

[Home](#)
[VolcanoList](#)
[Login](#)
[Register](#)

Login

Email:

Password:

[Login](#)

When the login link is clicked, users will be redirected to the login page. To access detailed volcano information, users must authenticate by logging in. If a user attempts to view volcano population chart without authentication, they will be notified that user needs to be logged in. Upon successful authentication, users will be redirected to the landing page. If an incorrect email address or password is entered during login, an error message will be logged stating "incorrect email or password".

/user/register

Home VolcanoList Login Register

Register

Email:

Password:

Register

Users can register their account by entering their email and password. The email must be in the correct format for registration to proceed.

Module Used

AG-Grid

<https://www.ag-grid.com/> - Module to provide fully-featured table components, including sorting and filtering

Pigeon Map

<https://pigeon-maps.js.org/> - Module to provide map components, including marking.

Chart.js

<https://www.chartjs.org/> - Module to provide chart components, including bar charts.

ReactDOM

<https://legacy.reactjs.org/docs/react-dom.html> - ReactDOM is a module that provides a specific method for rendering React components into the DOM (Document Object Model)

Application Design

Navigation and Layout

In designing the react-app, this project aimed for simplicity and user-friendliness. Clean and intuitive navigation system was desired to be implemented, with menu items like "Volcano List" and "Register" readily accessible. The flow between screens was planned to ensure a seamless user experience, with logical progression from landing page to volcano list and individual volcano information. The layout prioritizes clarity and readability, with important information presented prominently and a consistent design language maintained throughout the site. We considered alternative navigation structures and layouts but ultimately settled on one that prioritizes ease of use and accessibility for users interacting with volcano data.

Usability and Quality of Design

Improvements can be made by enhancing the visual hierarchy to prioritize key information and actions. This might involve using different font weights or colors for headings, subheadings, and body text. Ensuring all interactive elements like buttons and links are easily identifiable and consistent can further enhance usability. Gathering user feedback through usability testing is also essential for refining the design for optimal usability.

1. Organization and Layout: The app's layout is clean and organized, facilitating easy navigation. However, refining spacing and alignment could enhance visual clarity.
2. Navigation: Navigation is intuitive, allowing users to access different pages seamlessly. Maintaining consistency across all screens is vital to prevent confusion.
3. Consistency: The app aligns with user expectations regarding navigation and layout conventions. Maintaining consistency in design elements fosters a cohesive user experience.
4. Visual Design: The design maintains coherence with balanced font usage and a consistent color scheme. However, improving visual hierarchy and ensuring consistency in interactive elements like buttons can enhance usability.

Accessibility

Here's an analysis of the app's accessibility based on the provided checklist:

1. Text Equivalents for Non-text Elements: The app should ensure that all non-text elements, such as images and buttons, have appropriate text alternatives. This helps users with screen readers or those who have disabled images to understand the content.
2. Color Accessibility: Information conveyed through color should also be available without relying solely on color. For example, if certain elements are highlighted in blue, there should be additional indicators like labels or symbols to convey the same information.
3. Readable Documents Without Style Sheets : Users should be able to read and understand the content even if style sheets are disabled or not supported.
4. Update Text Equivalents for Dynamic Content: Text equivalents should be updated dynamically when content changes. This ensures that users relying on screen readers or other assistive technologies receive timely and accurate information.
5. Avoid Flickering: Flickering content can be distracting and disorienting, especially for users with visual sensitivities or certain neurological conditions. The app should minimize or eliminate any unnecessary screen flickering to enhance usability.
6. Clear and Simple Language: Using clear and simple language improves accessibility for users with cognitive disabilities or those with limited language proficiency. Complex terms and jargon should be avoided where possible to make content more understandable to a wider audience.
7. Identify Row and Column Headers in Tables: Tables should include clear identification of row and column headers to facilitate navigation and comprehension, particularly for users accessing the content with assistive technologies like screen readers.

Overall, while the app may meet some of these accessibility guidelines, there may be areas for improvement, such as ensuring all non-text elements have appropriate text alternatives and enhancing readability without relying heavily on color or style sheets. Regular accessibility testing and user feedback can help identify and address any accessibility barriers effectively.

Technical Description

Architecture

build	13/05/2024 10:39 PM	File folder	
config	13/05/2024 10:39 PM	File folder	
node_modules	13/05/2024 11:01 PM	File folder	
public	13/05/2024 10:44 PM	File folder	
src	13/05/2024 10:44 PM	File folder	
.gitignore	13/05/2024 10:39 PM	Git Ignore Source ...	1 KB
babel.config	13/05/2024 10:39 PM	JavaScript File	1 KB
package	13/05/2024 10:39 PM	JSON Source File	2 KB
package-lock	13/05/2024 10:39 PM	JSON Source File	704 KB

The application's source code structure follows a standard convention, "config" housing configuration files, and "node_modules" storing dependencies. The "public" directory holds static assets, while the "src" directory contains the main source code files, including JavaScript, JSX, and CSS. Configuration files like ".gitignore" and "babel.config" manage project settings, while "package.json" and "package-lock.json" define project metadata and dependencies. This organization ensures a clear separation of concerns and facilitates efficient development, build, and deployment processes.

Name	Date modified	Type	Size
components	13/05/2024 10:44 PM	File folder	
App	13/05/2024 10:44 PM	JavaScript File	1 KB
App.test	13/05/2024 10:44 PM	JavaScript File	3 KB
index	13/05/2024 10:44 PM	Cascading Style S...	1 KB
index	13/05/2024 10:44 PM	JavaScript File	1 KB
navigation	13/05/2024 10:44 PM	Cascading Style S...	1 KB
reportWebVitals	13/05/2024 10:44 PM	JavaScript File	1 KB
setupTests	13/05/2024 10:44 PM	JavaScript File	1 KB

- Components:** This folder contains the main components of the application.
- App.js:** This component serves as the main entry point of the application, handling routing and rendering different components based on the URL.
- app.test.js:** This file contains unit tests for the App component.
- Other components: Additional components for different parts of the application, such as navigation, could be present here.



index.css: This file likely contains global CSS styles for the application.

index.js: This file is the entry point for rendering the React application. It likely imports the App component and mounts it to the DOM.

navigation.css: This file might contain CSS styles specifically related to navigation components if they exist.

setupTests.js: This file contains setup configurations for running tests, such as configuring testing libraries or mocking dependencies.

In this structure, the components folder houses the main components responsible for different parts of the UI. The index.js file serves as the entry point for rendering the React application, while setupTests.js provides configurations for testing.

Difficulties / Exclusions / unresolved & persistent errors

I encountered difficulties while integrating Pigeon Map and Chart.js into the application. Initially, errors persisted, preventing the proper display of these components. After downgrading the React version, I managed to implement the Pigeon Map successfully. However, Chart.js continued to pose challenges. Upon investigation, I discovered that the user authentication was not functioning correctly, leading to issues in retrieving population information even when users were logged in. Once I resolved this authentication issue, I was able to successfully implement Chart.js.

Additionally, I faced challenges with running unit tests for components written in Java. It appears that there may be misconfigurations in the setup of the Babel test, causing consistent errors during testing. This issue remains unresolved, and further investigation is required to identify and rectify the configuration errors.

Extensions

Potential future extensions and improvements for the app include resolving issues with Java unit tests to ensure code reliability, implementing advanced authentication methods like two-factor authentication and OAuth integration, integrating sophisticated chart libraries for enhanced data visualization, conducting an audit and implementing features for compliance with accessibility guidelines, optimizing performance through code splitting and lazy loading, supporting multiple languages and regions for internationalization and localization, and collecting user feedback and analyzing usage patterns for continuous improvement.

User Guide

Welcome to Volcano DB

[Home](#)
[VolcanoList](#)
[Register](#)
[Login](#)



First this is the Landing Page. If user selects “Home”, this will redirect to the the landing page.

[Home](#)
[VolcanoList](#)
[Login](#)
[Register](#)

Volcano List

Select a country ▼

[Home](#)
[VolcanoList](#)
[Login](#)
[Register](#)

Volcano List

Japan ▼


ID	Name	Country	Region	Subregion
1	Abu	Japan	Japan, Taiwan, Marianas	Honshu
16	Aogashima	Japan	Japan, Taiwan, Marianas	Izu, Volcano, and M
30	Adatarayama	Japan	Japan, Taiwan, Marianas	Honshu
65	Asamayama	Japan	Japan, Taiwan, Marianas	Honshu
68	Aira	Japan	Japan, Taiwan, Marianas	Ryukyu Islands and
75	Akagisan	Japan	Japan, Taiwan, Marianas	Honshu
76	Asosan	Japan	Japan, Taiwan, Marianas	Ryukyu Islands and
78	Akan	Japan	Japan, Taiwan, Marianas	Hokkaido
81	Akandanayama	Japan	Japan, Taiwan, Marianas	Honshu

Page Size: 100 ▼ 1 to 100 of 122 1K < Page 1 of 2 > >1

If the user is directed to the volcano list page, the user will be able to select the country and corresponding volcanoes will be displayed on the grid. Users can easily sort by ID number or by name.

Home VolcanoList Login Register

Abu
Country: Japan
Region: Japan, Taiwan, Marianas
Subregion: Honshu
Last Eruption: 6850 BCE
Summit: 641
Elevation: 2103
Latitude: 34.5000
Longitude: 131.6000
[View Population Chart](#)



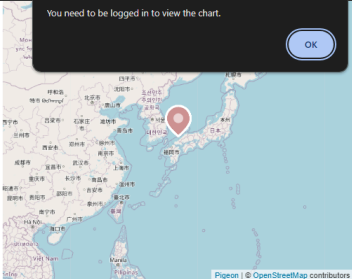
Pigeon | © OpenStreetMap contributors

User can see the volcano details once the volcano is selected however, when user wants to see the population chart this will occur.

← → ↻ localhost:3000/volcano/1

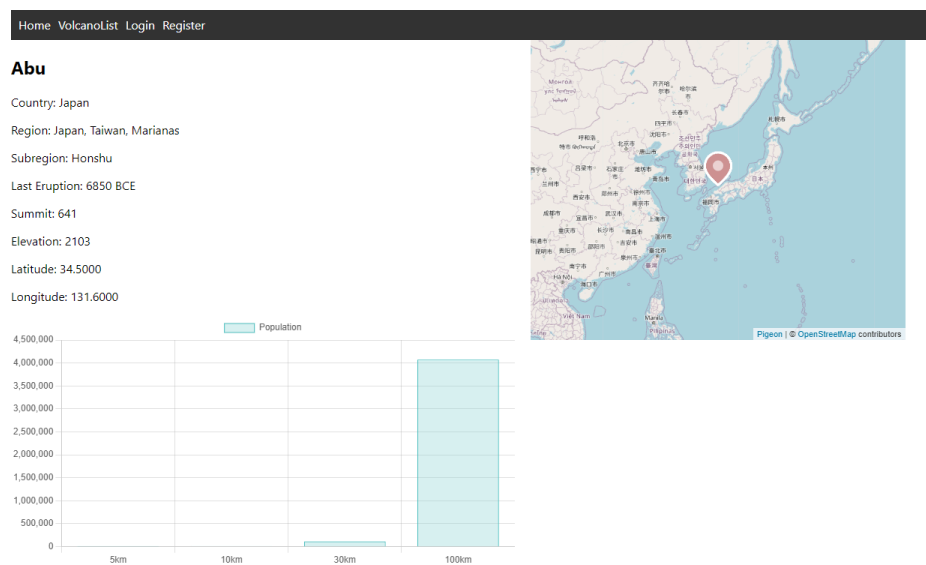
Home VolcanoList Login Register

Abu
Country: Japan
Region: Japan, Taiwan, Marianas
Subregion: Honshu
Last Eruption: 6850 BCE
Summit: 641
Elevation: 2103
Latitude: 34.5000
Longitude: 131.6000
[View Population Chart](#)



Pigeon | © OpenStreetMap contributors

localhost:3000 says
You need to be logged in to view the chart.
[OK](#)



If user log in, population data should be able to be seen.

References

1. *Chart.js*. (2023.). Open Source HTML5 Charts for Your Website.
<https://www.chartjs.org/>
2. *Demo | Pigeon Maps*. (2022.). <https://pigeon-maps.js.org/>
3. *REACT Grid: AG Grid: Reference | AG Grid*. (2023.).
<https://www.ag-grid.com/react-data-grid/reference/>
4. *react-dom*. (2022). Npm. <https://www.npmjs.com/package/react-dom>