

2022

CAB230 Assignment 1 Client Side



CAB230

Volcano API – Client Side Application

<Yeonsu Lee>

<N11155418>

5/20/2022

Contents

Introduction.....	2
Purpose & description.....	2
Completeness and Limitations.....	2
Use of End Points.....	3
/rankings.....	3
/countries.....	3
/factor/{year}.....	3
/user/register.....	3
/user/login.....	3
Modules Used.....	3
Ag-grid-react.....	3
React-Select.....	3
Pigeon maps.....	3
React-Charjs-2.....	3
Application Design.....	4
Navigation and Layout.....	4
Usability and Quality of Design.....	4
Accessibility.....	4
Technical Description.....	5
Architecture.....	5
Test plan.....	6
Difficulties / Exclusions / unresolved & persistent errors.....	6
Extensions (Optional).....	6
User guide.....	6

Introduction

Purpose & description

Our web application's purpose is to interact with customers' interests by using invisibly with Web API, which has volcanos information of 1,343 countries. When the customer enters our web, our app shows our web application purpose and functionality on the Carousel with volcanos pictures. Also, there is a navigation bar on which customers select login, join and volcano list. These menus enable our customers to select their purpose. Especially the menu volcanylist, our main functionality, shows the information of volcanos such as name, region, subregion, country, by filtering country and distance populated with. The detailed information can be found by clicking the row in which customers want more information. The detailed information has two main differences, the customer's login and none. The customers who log in to our web application can see more details, such as a map and population chart; otherwise can not. These differences attract customers to join our web and improve usability.

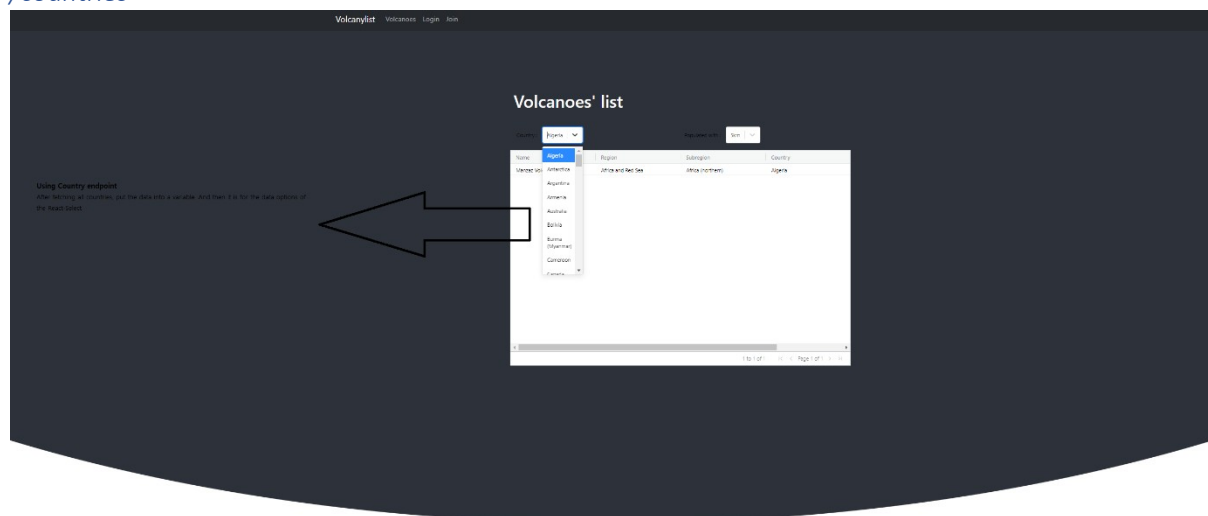
The differences from other websites, we used Reactstrap, React Select modules, Caroussel, AG-Grid, Pigeon maps and React-chartjs 2. Using the Reactstrap and Carousel make our web more illustrative and interactive. AG-Grid and React Select make filtering more convenient and show information about volcanos at a glance. The others, Pigeon maps and React-chartjs 2, made differences between those who log in and those who do not and more details such as map and pie chart.

Completeness and Limitations

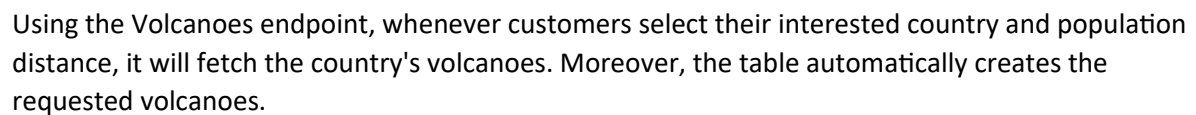
Every API endpoint is implemented correctly, such as data and authenticate query endpoint, data API, and modules such as AG-Grid (including filter or sorting), Pigeon maps, chartjs, Reactstrap and React-select. However, initially, we planned to use AG-Grid's filter system. So I wanted to fetch all the volcanos data using For statement in the fetched country API. However, the 429 error occurs because of too many Swagger API requests if we fetch all the data. So instead, I used the React-Select module like the AG-Grid filter system to make a more interactive component. It enables customers to filter more detailed; it can support dropdown and free text input.

Use of End Points

/countries



/volcanoes



/volcano/{id}

Volcanylst

Volcanoes

Logout

Manzaz Volcanic Field information

The volcano that you selected was Manzaz Volcanic Field

Manzaz Volcanic Field

Country : Algeria
Region : Africa and Red Sea
Subregion : Africa (northern)
Last Eruption : Unknown
Summit : 1672
Elevation : 5486

Manzaz Volcanic Field map

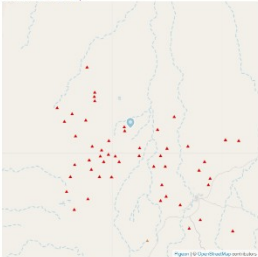
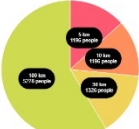


Figure 14-34 © WorldMap.com authors

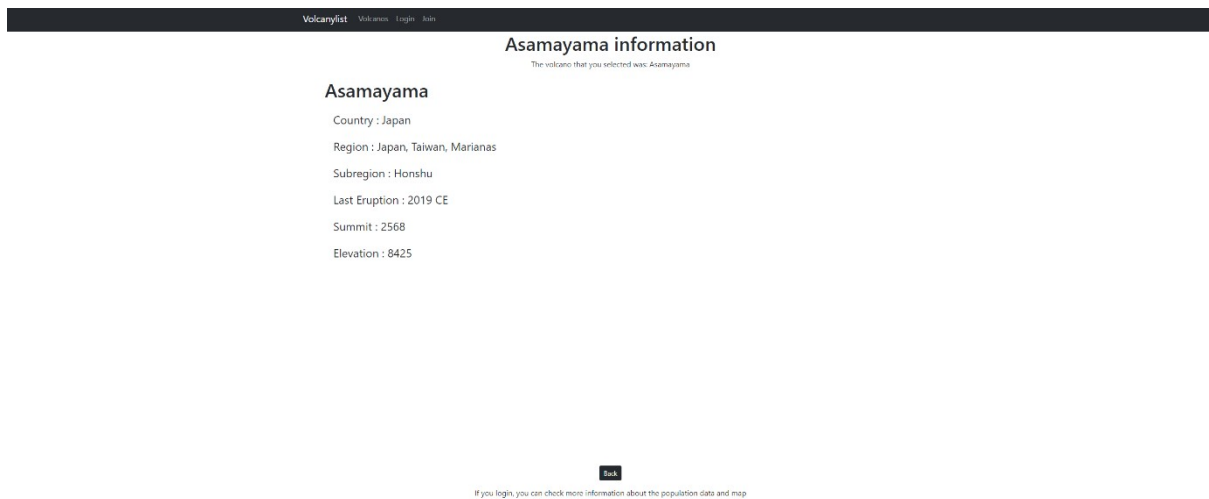
Latitude:23.92
Longitude:5.83

Info

Population chart

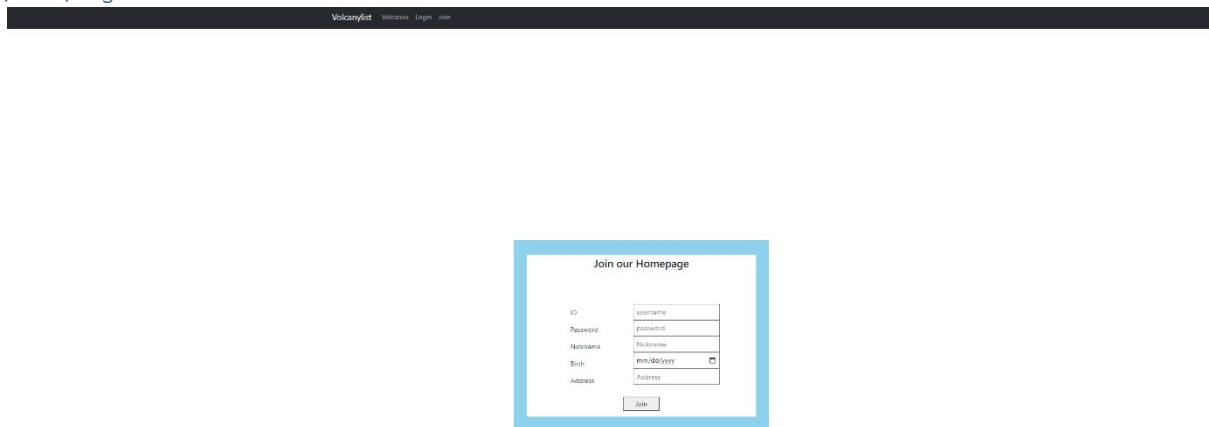


Customers who do not log in

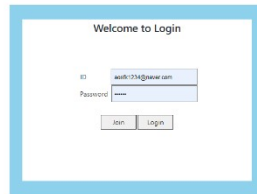


Using the `Volcanoes/id` endpoint, when customers select the interesting row, it makes the information from the `Volcanoes/id` endpoint according to whether they log in or not.

</user/register>



Using the register endpoint, customers post their information and join our web application.



Using the login endpoint, customers post their information and log in to our web application.

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 to provide some component library like Bootstrap CSS framework. Of the components, I used the navbar and carousel.

<https://reactstrap.github.io/>

React-Select

Module to provide dropdown and free text input.

<https://react-select.com/home>

Pigeon maps

Module to provide Maps display.

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

React-Chartjs-2

Module to provide various charts using the data. Of the charts, I used the Pie chart.

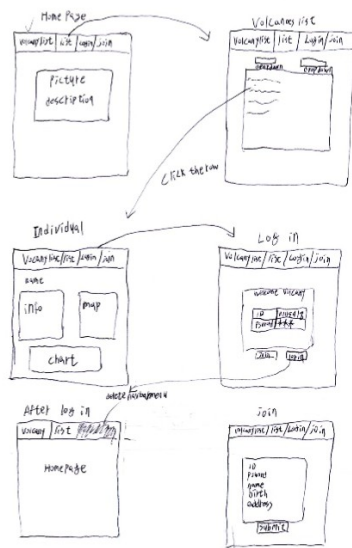
<https://www.npmjs.com/package/react-chartjs-2>

Application Design

Navigation and Layout

Since a neat web application was considered the basis of the project, I used a carousel in react-strap to show photos of volcanoes and a basic description of our homepage. Moreover, customers can smoothly use web functionality according to their interests through React-Strap Navbar. Using the React-Strap, make it interactive with our web.

Initial sketch for my project



Usability and Quality of Design

- Our web application display is clean as Instagram. Moreover, customers can see the functionalities at a glance. However, unlike other web applications, such as the Smithsonian Institution's Global Volcanism Program (<https://volcano.si.edu/>), our web application needs more functionality visually like today's volcano activity or volcanoes' news. It is so clean that there is so much empty space. So, as a side effect, customers may regard our web application as visually unattractive.
- The navigation bar is clear and intuitive. However, as written above, a lack of functionalities and empty space is a side effect to customers.
- Using the router module from React, the visual design is consistent with all pages. However, the individual volcanoes page and volcanoes page is not put in the homepage's background. When the background is in them, it visually can not attract functionality such as table, chart and map. However, customers can feel awkward, unlike on the homepage to the lack of colour.

Accessibility

Web content accessibility checklist	Yes	No
Provide a text equivalent for every non-text element – alternatives to images, symbols, scripts, graphical buttons, sounds, audio and video files and so on.	Check	

all the non-text elements are with text, and customers understand what the functionality will be supposed to do. For example, the button of login or join, map and population chart can easily enable customers to understand what it is.

Ensure that all information conveyed with color is also available without color, for	Check	
--	-------	--

example from context or markup.		
---------------------------------	--	--

All the elements can easily be noticeable at a glance. Especially, the volcanoes table section and the individual section with chart and map do not have background colour because it can hinder this functionality.

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.	Check	
--	-------	--

All the documents are implemented without style sheets. I used all the data inline style CSS or style sheet in the project. Therefore, the Style sheet simply is considered an additional functionality. So in the case of deleted style sheet, all the documents are implemented, and customers can read all data, although it can not be intuitive.

Ensure that text equivalents are updated when dynamic content changes.	Check	
--	-------	--

All the data are updated dynamically whenever customer select their interests. When customers select it, according to select value, it will fetch API of select value. And then, it will dynamically update it. It will make customers more interactive with our application.

Avoid causing the screen to flicker.	Check	
--------------------------------------	-------	--

Our application does not have blink context or contents using CSS or other modules. Therefore, it meets the criteria.

Use the clearest and simplest language appropriate for a site's content.	Check	
--	-------	--








In our application, all the contents like volcano information, Navbar menu or homepage description are comprised of simple and appropriate text.

For tables, identify row and column headers – clearly differentiated from the data.	Chcek	
---	-------	--

All the column headers of the table are assigned to appropriate text. So, customers can check what is for the data at a glance.

Technical Description








Architecture

Name	Date modified	Type	Size
 node_modules	5/18/2022 4:23 AM	File folder	
 public	5/17/2022 7:41 PM	File folder	
 src	5/19/2022 6:34 AM	File folder	
 .gitignore	5/11/2022 6:24 PM	Text Document	1 KB
 package	5/18/2022 4:23 AM	JSON File	2 KB
 package-lock	5/18/2022 4:23 AM	JSON File	1,164 KB
 README	5/11/2022 6:25 PM	Markdown Source...	4 KB

Basically, there are src, modules, and public folders in our project's source code. Pictures of volcanoes for the carousel on the homepage are saved in the public folder.








For making our web application, I needed to split the components and screens (web pages), which have a code for their purpose. So, main folder, src, is comprised of component folder and screens folder.




 component	5/21/2022 11:57 PM	File folder	
 screens	5/21/2022 9:02 PM	File folder	
 App	5/22/2022 12:18 AM	Cascading Style S...	1 KB
 App	5/22/2022 12:18 AM	JavaScript File	1 KB
 index	5/22/2022 12:18 AM	Cascading Style S...	1 KB
 index	5/18/2022 12:59 AM	JavaScript File	1 KB
 setupTests	5/22/2022 12:18 AM	JavaScript File	1 KB

In the screens folder, there are basic web page codes such as homepage, volcano list, individual, login, and join. The screens mainly is comprised of components in component folder and some code for the styles and container. It will show functionality.

In src/screens

 Homepage	5/22/2022 12:18 AM	JavaScript File	1 KB
 Individual	5/22/2022 12:18 AM	JavaScript File	7 KB
 Join	5/22/2022 12:18 AM	JavaScript File	5 KB
 Login	5/22/2022 12:18 AM	JavaScript File	4 KB
 Volcanos	5/22/2022 12:18 AM	JavaScript File	6 KB

In the component folder, it is for the components. There are the NavBar component, body folder, and styles folder. Firstly. NavBarElement component is used in the APP.js through router because it will consistently be shown whenever customers click the navbar menu. Because it is considered the most basic component, the component folder has it.

 body	5/15/2022 10:16 PM	File folder	
 Styles	5/9/2022 12:49 AM	File folder	
 NavBarElements	5/22/2022 12:18 AM	JavaScript File	3 KB

Secondly, the body folder has the carousel code component. It is for the homepage carousel. It is comprised of the code of volcano pictures' path and our project description. And homepage of the Screens folder using the carousel component code show customers the carousel.

In component/body



CSS styles are saved in the Styles folder to make it easier to apply header, container, and webpage design to other Screens. So, on the homepage, login, join of the Screens folder used the header and container for design. In component/styles/container



Container

5/22/2022 12:04 AM

JavaScript File

1 KB

In component/styles/header



Header.styled

5/19/2022 7:03 AM

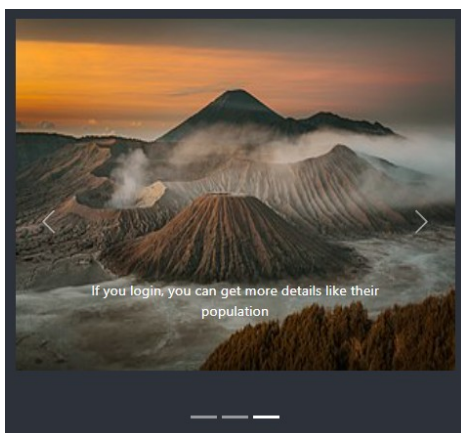
JavaScript File

1 KB

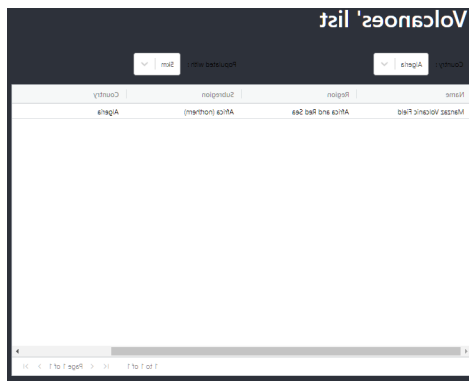
Test plan

Task	Expected Outcome	Result	Screenshot
Carousel image and text	Show the image and text	PASS	1
Data table	Show the data by clicking	PASS	2
Free input text on React-Select	Free input text of the country	PASS	3
Sort of the volcano name	Sorting from alphabet "A"	PASS	4
Check pagination of the table	Smoothly show next data	PASS	5
Check the map and chart when log in	Show the map and chart	PASS	6
Check the map and chart when not log in	Don't Show the map and chart	PASS	7
Check the deleted menu login/join on the NavBar	Don't show the login/joining menu	PASS	8
Invalid login, error message	Alert error message	PASS	9
When joining, ID already exists	Alert error message	PASS	10

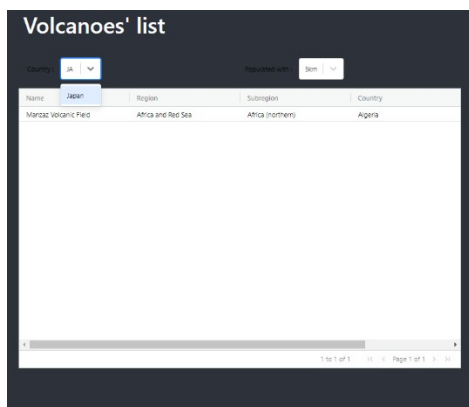
1. Carousel image and text



2. Data table



3. Free input text



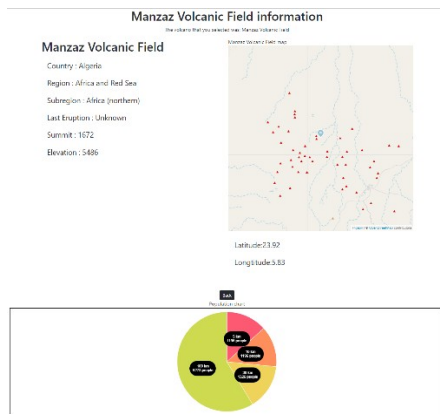
4. Sorting name of volcano

Name	Region	Subregion	Country
Abu	Japan, Taiwan, Marianas	Honshu	Japan
Agashima	Japan, Taiwan, Marianas	Izu, Volcano, and Mariana Islands	Japan
Adatsuyama	Japan, Taiwan, Marianas	Honshu	Japan
Adamiyama	Japan, Taiwan, Marianas	Honshu	Japan
Aira	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Akagisan	Japan, Taiwan, Marianas	Honshu	Japan
Akasan	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Akan	Japan, Taiwan, Marianas	Hokkaido	Japan
Ata	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Akita-Komagatake	Japan, Taiwan, Marianas	Honshu	Japan
Akita-Nakayama	Japan, Taiwan, Marianas	Honshu	Japan
Akusejiima	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Aburatsubo	Japan, Taiwan, Marianas	Honshu	Japan
Banda-san	Japan, Taiwan, Marianas	Honshu	Japan
Chachadake (Tate)	Kuri Islands	Kuri Islands	Japan

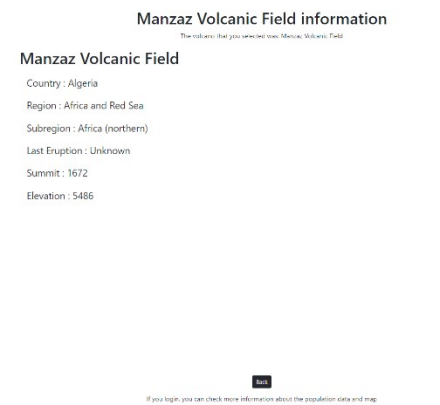
5. Check pagination of the table

Name	Region	Subregion	Country
Huichigatake	Japan, Taiwan, Marianas	Honshu	Japan
Hokkaido-Komagatake	Japan, Taiwan, Marianas	Hokkaido	Japan
Kikai	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Kirishimayama	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Kita-Ito	Japan, Taiwan, Marianas	Izu, Volcano, and Mariana Islands	Japan
Koburima	Japan, Taiwan, Marianas	Izu, Volcano, and Mariana Islands	Japan
Kuchinoshima	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Kuchinoshima	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Kyushu	Japan, Taiwan, Marianas	Honshu	Japan
Kyushu	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Ito	Japan, Taiwan, Marianas	Izu, Volcano, and Mariana Islands	Japan
Nakanoshima	Japan, Taiwan, Marianas	Ryukyu Islands and Kyushu	Japan
Namisan	Japan, Taiwan, Marianas	Honshu	Japan
Naruko	Japan, Taiwan, Marianas	Honshu	Japan
Kurikayama	Japan, Taiwan, Marianas	Honshu	Japan

6. Check the map and chart when log in

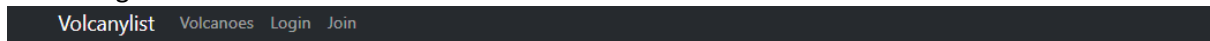


7. Check the map and chart when not log in



8. Check the deleted menu login/join on the NavBar

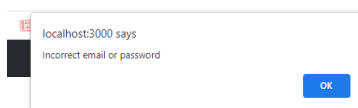
Before login



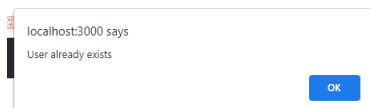
After login



9. Invalid login, error message



10. When joining, ID already exists



Difficulties / Exclusions / unresolved & persistent errors /

When using Pigeon-maps API, our web did not load the map. Although the data which has been fetched was parsed as an integer from string type, the map was not loaded. So, I checked the data type by using console.log. In the beginning, printed the NaN, and then printed integer type. It is because the useEffect used the NaN data, which has not yet been fetched from the API. Therefore, I used the If statement to fetch just in case of the integer type, not NaN data. As a result, it worked.

Moreover, I wanted to use the AG-Grid filter because it is more interactive with the customer. However, to use the AG-Grid filter, row data of AG-Grid need all country's volcanoes information. So, I tried to put the For statement with fetching data (Countries endpoint) into the For statement with fetching data (Volcanoes). When I check the data using console.log, all the data is printed correctly. However, the 429 error (too many requests) sometimes occurred. To solve the error, I needed to ask the swagger team. As a result, instead of the AG-Grid filter, to be more interactive, I used React-Select. It results that our application is made more convenient using free text input.

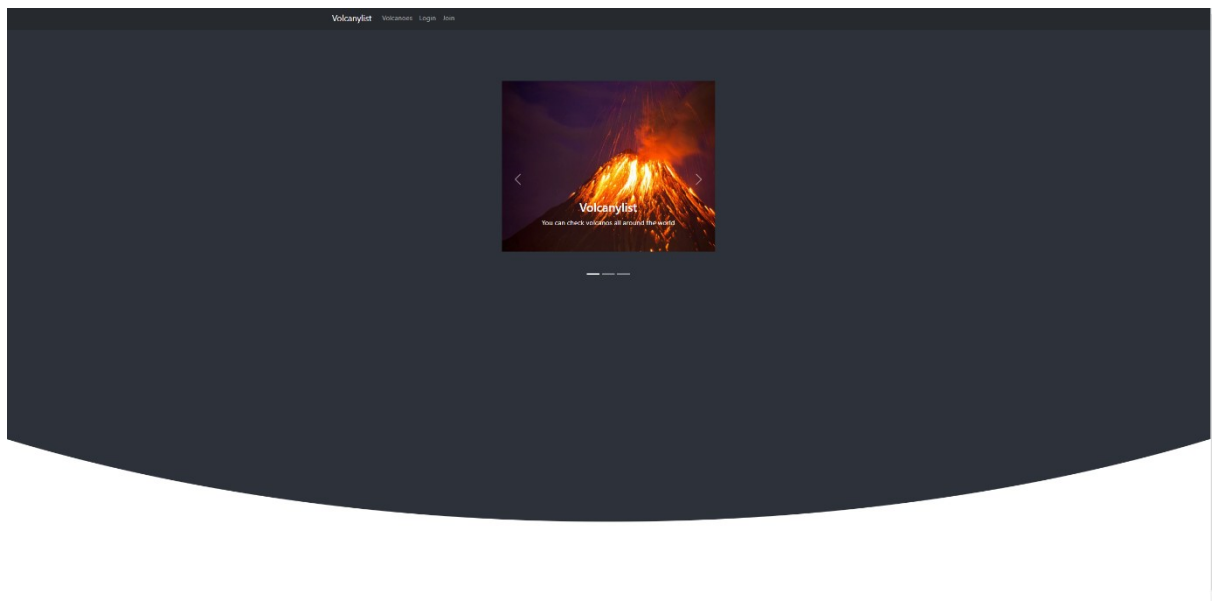
Extensions (Optional)

If there is today's volcano activity API, I want to add today's volcano activity widget. So, from today's volcanos activity widget, the functionality enables customers can sort recent volcano activity and see the pictures. This extension of functionality attracts more customers and improves our application.

User guide

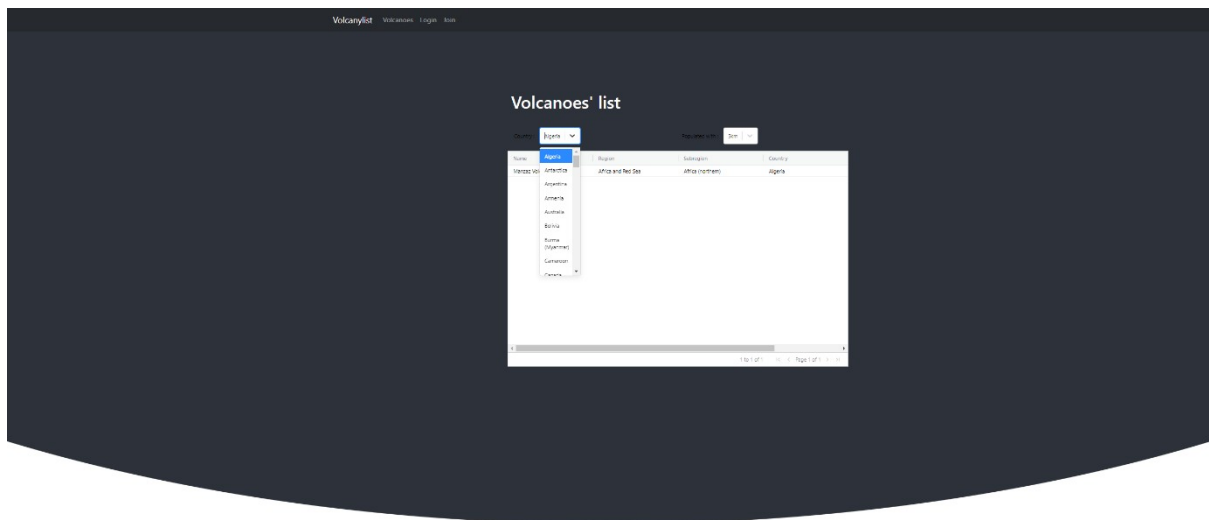
Firstly, when customers enter our web application, they can see the description of our application.

1. Homepage



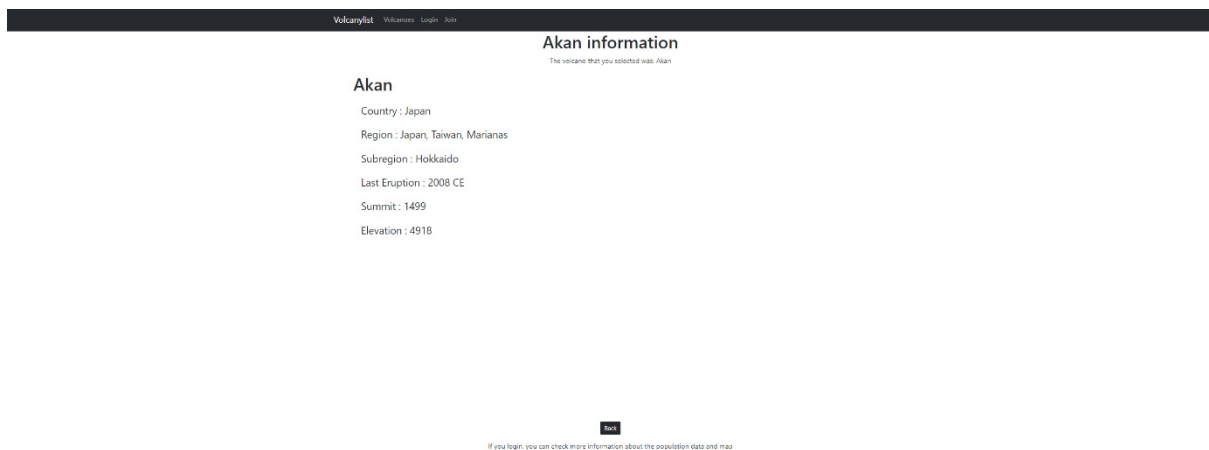
And then, they will use the volcanoes on the NavBar menu to check volcanoes' information. When clicking the volcanoes. They can see the table and dropdown, two features (countries and population within). Whenever selecting the dropdown, our application fetches the data from the API and will automatically update the data.

2. Volcanoes list



Moreover, customers can use free text input of the two dropdowns of the React-Select. Moreover, when the customer selects their interesting row, our application will update the information about the volcano.

3. Individual page (Not log in)



When clicking the row and not logging in, they can see the above screens. Moreover, the application states that if login, customers can check more details about the volcanoes.

Customers need to join or log in to see more details about the volcanoes. So, they will use the menu of NavBar. They can log in or join.

4. Log in page

Welcome to Login

ID	<input type="text" value="user1234@vokarylist.com"/>
Password	<input type="password" value="1234"/>
<input type="button" value="Join"/> <input type="button" value="Login"/>	

5. Join page

Join our Homepage

ID	<input type="text" value="user1234@vokarylist.com"/>
Password	<input type="password" value="1234"/>
Nickname	<input type="text" value="NickName"/>
Birth	<input type="text" value="year1987year"/>
Address	<input type="text" value="Address"/>
<input type="button" value="Join"/>	

6. Individual page (Log in)

