

CAB230

[ASSIGNMENT 2: CLIENT SIDE]
['Volcano DB' React Web Application]



[Zachary Womal | n9669396]

[12/05/2024]

Report Contents

Report Contents.....	2
Introduction.....	3
Purpose & description.....	3
Completeness & Limitations.....	3
Use of Endpoints.....	4
/countries, /volcano.....	4
/volcano/{id}.....	5
/user/register.....	6
/user/login.....	8
Modules Used.....	9
Application Design.....	11
Navigation & Layout.....	11
Usability and Quality of Design.....	12
Accessibility.....	13
Technical Description.....	14
Architecture.....	14
Test plan.....	15
Difficulties, Exclusions, Unresolved Errors.....	17
User Guide.....	20
Use Case A.....	20
Use Case B.....	21
References.....	23
Screenshot Appendices.....	24

Introduction

Purpose & description

'Volcano DB' is a web application written in React and Typescript, with the intended purpose of allowing users to view and analyze data on a range of volcanoes with data taken from The Smithsonian Institution's Global Volcanism Program. By using a Javascript framework like React, the web application provides the user with a comparatively more modern and intuitive front-end interface to a small subset of the data available in the Global Volcanism Program database.

The application makes use of Tailwind CSS to help build a bespoke set of functions and React components to glue the style and functionality of the site together, as a way to make the application feel less 'generic' while remaining visually interesting, interactive and familiar to the application's users.

Completeness & Limitations

The application as a whole uses Tailwind CSS, a "utility first" CSS framework (Tailwind Labs, n.d.), to create a more purpose-built application in comparison to CSS/component libraries like Bootstrap, while still streamlining the development process in comparison to "vanilla" CSS stylesheets. Navigation options are handled in a menu bar running along the top of the website, with all routing handled by React Router - this means the application performs its page updates in a "single page application" style, where page elements are only re-rendered as their state changes, rather than performing a page-wide re-render when a new page is requested.

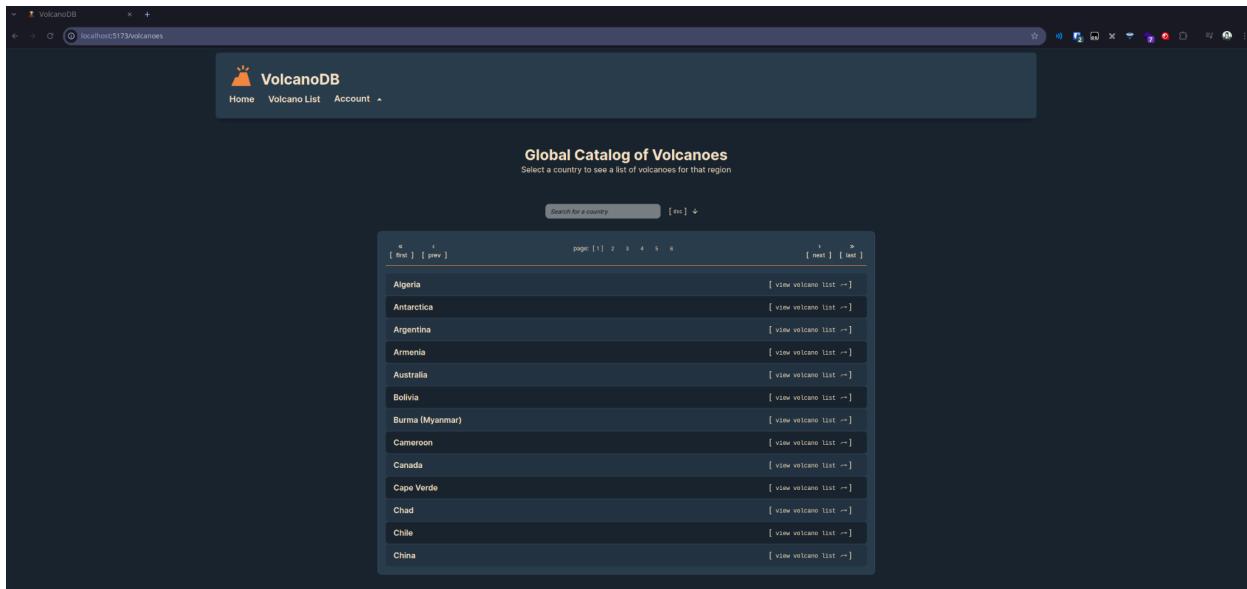
The display of thorough and intuitive data displays and filtering options are presented via a mix of bespoke solutions (such as the countries list, which implements a custom fuzzy-matching search function, list reordering, and pagination), with complex functionality handled by Node modules like Chart.js (which handles the rendering of a bar graph for population information), and Ag Grid React (to display the more complex tables of volcanoes and handle volcano table sorting).

User accounts are (for the most part) managed on the server side (e.g, passwords & hashing, secrets for token generation, user information, responses, and so on), so the implementation here is simply handling the provided tokens and related session status - if user signs in, the token is cached in the client's local storage, and if a user signs out, the token is cleared. The main reason for a user session is to facilitate access to gated information, so if a token is available to a user, it is sent with a request to fetch the authenticated dataset instead of the "free" dataset. Client data is appropriately cleared on logout so that authenticated information cannot be accessed by an unauthenticated user.

Use of Endpoints

/countries, /volcano

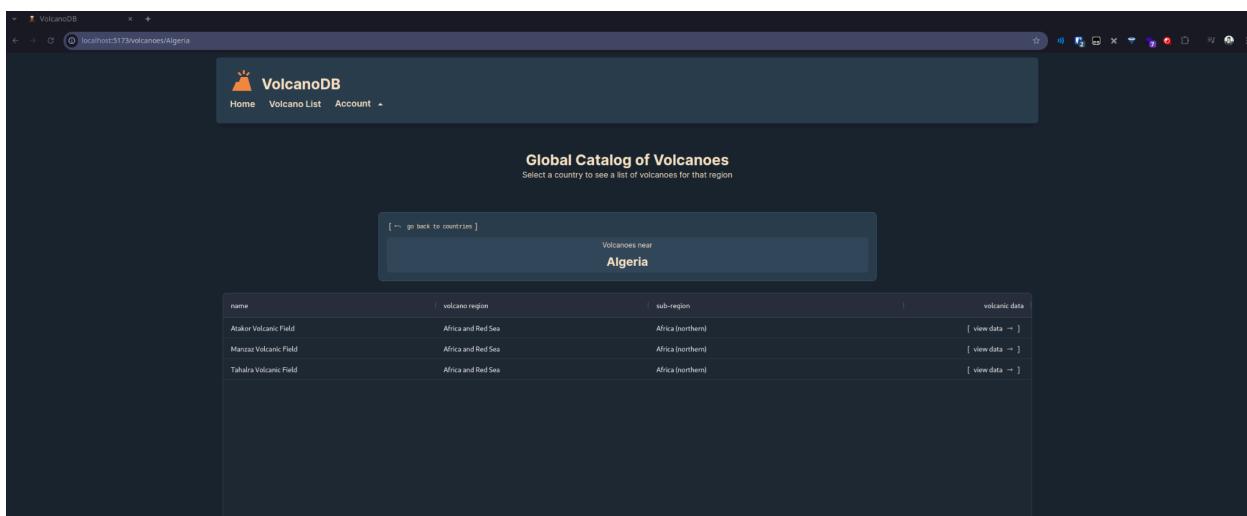
The `/countries` endpoint is coupled with the `/volcanoes` endpoint on the client side, as the two have coupled functionality given a user is intended to view volcanoes dependent on a selected country. It should be noted that the client side endpoint naming convention is the reverse of that on the server side to reflect the intended control flow (i.e, viewing a list of volcanoes by specifying a country).



A screenshot of a web browser displaying the 'VolcanoDB' application. The URL in the address bar is `localhost:5173/volcanoes`. The page title is 'VolcanoDB'. The main content area is titled 'Global Catalog of Volcanoes' with the sub-instruction 'Select a country to see a list of volcanoes for that region'. Below this is a search bar labeled 'Search for a country' with a placeholder '(e.g.)' and a dropdown menu showing the letter 'A'. A table lists countries with their names and a 'view volcano list' button. The countries listed are Algeria, Antarctica, Argentina, Armenia, Australia, Bolivia, Burma (Myanmar), Cameroon, Canada, Cape Verde, Chad, Chile, and China. The table has navigation buttons at the top and bottom.

name	volcano region	sub-region	volcanic data
Atakor Volcanic Field	Africa and Red Sea	Africa (northern)	[view data →]
Mancor Volcanic Field	Africa and Red Sea	Africa (northern)	[view data →]
Tahara Volcanic Field	Africa and Red Sea	Africa (northern)	[view data →]

The list of countries presented to a user at the `/volcanoes` route



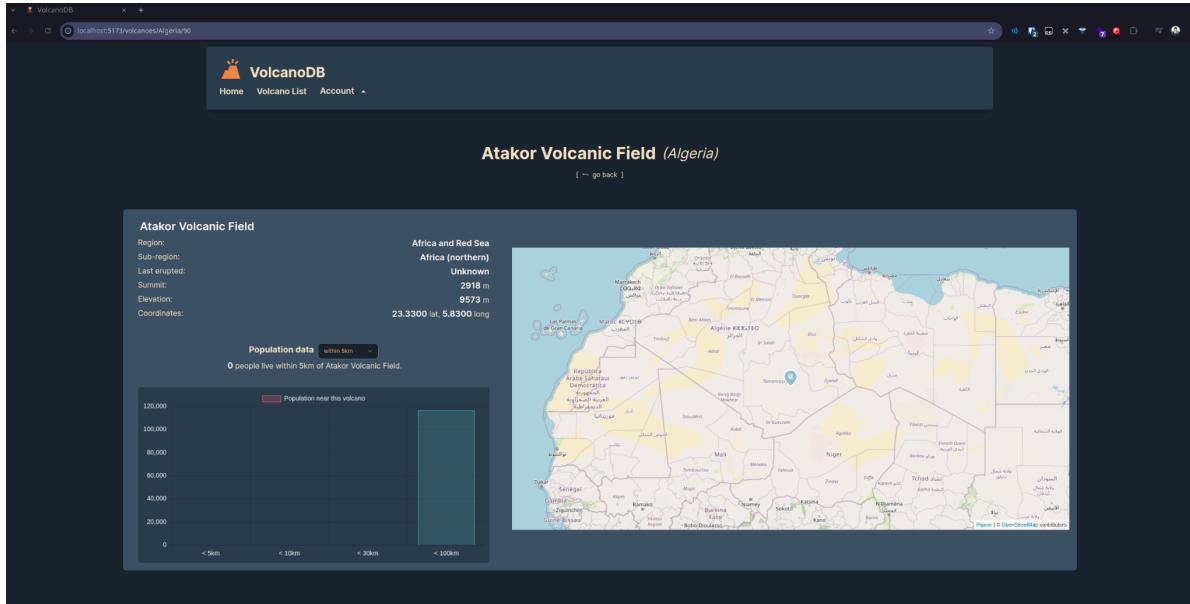
A screenshot of a web browser displaying the 'VolcanoDB' application. The URL in the address bar is `localhost:5173/volcanoes/Algeria`. The page title is 'VolcanoDB'. The main content area is titled 'Global Catalog of Volcanoes' with the sub-instruction 'Select a country to see a list of volcanoes for that region'. At the top left is a link '[← go back to countries]'. Below this is a section titled 'Volcanoes near Algeria'. A table lists volcanoes with their names, regions, and sub-regions, along with a 'view data' button. The volcanoes listed are Atakor Volcanic Field, Mancor Volcanic Field, and Tahara Volcanic Field, all located in Africa and Red Sea, Africa (northern).

name	volcano region	sub-region	volcanic data
Atakor Volcanic Field	Africa and Red Sea	Africa (northern)	[view data →]
Mancor Volcanic Field	Africa and Red Sea	Africa (northern)	[view data →]
Tahara Volcanic Field	Africa and Red Sea	Africa (northern)	[view data →]

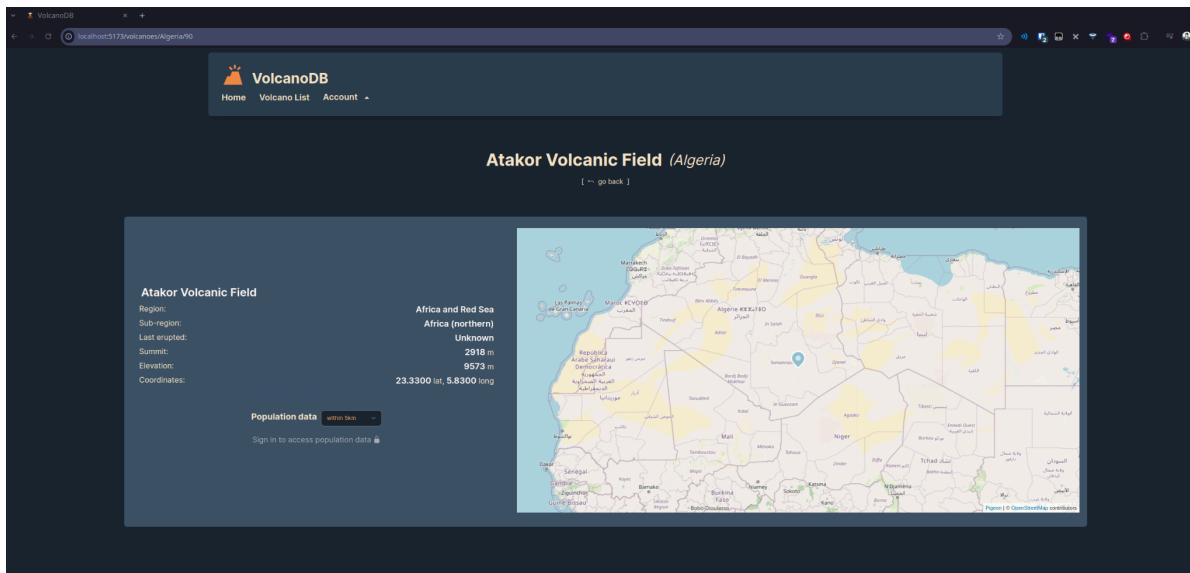
The volcanoes grid at `/volcanoes/{country}` - the above example uses `/volcanoes/Algeria`.

/volcano/{id}

The client handles data from this endpoint at `/volcanoes/{country}/{id}` (where `{id}` represents a specific volcano ID). The exact presentation changes depending on a user's authorization status:



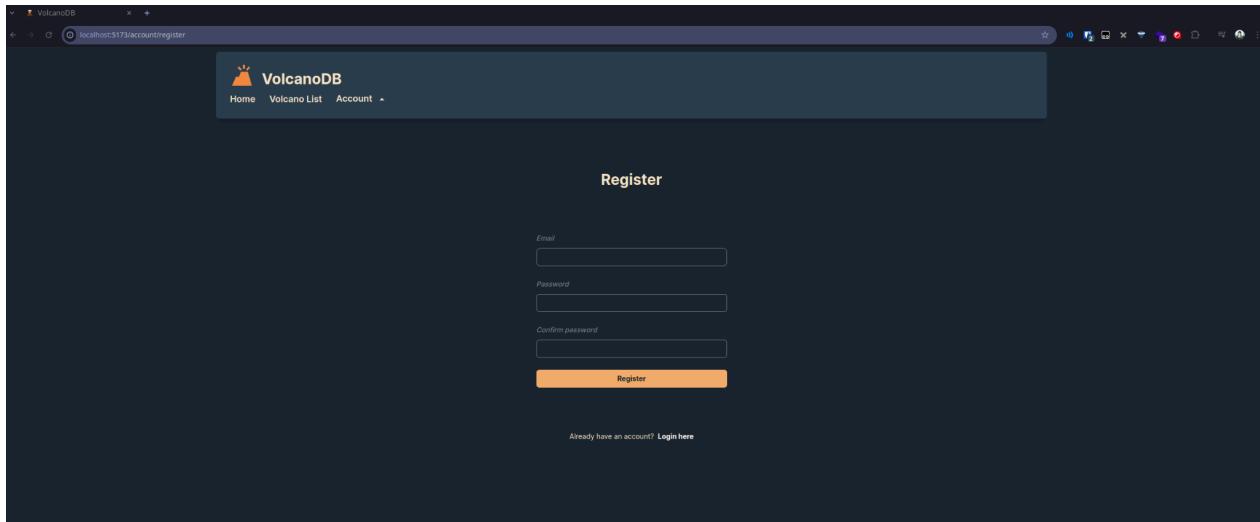
The authenticated view at `/volcanoes/{country}/{id}` - the example above features `/volcanoes/Algeria/90`



The unauthenticated view of the same page, `/volcanoes/Algeria/90`, where the text and graph of population data is replaced with a sign-in prompt and link.

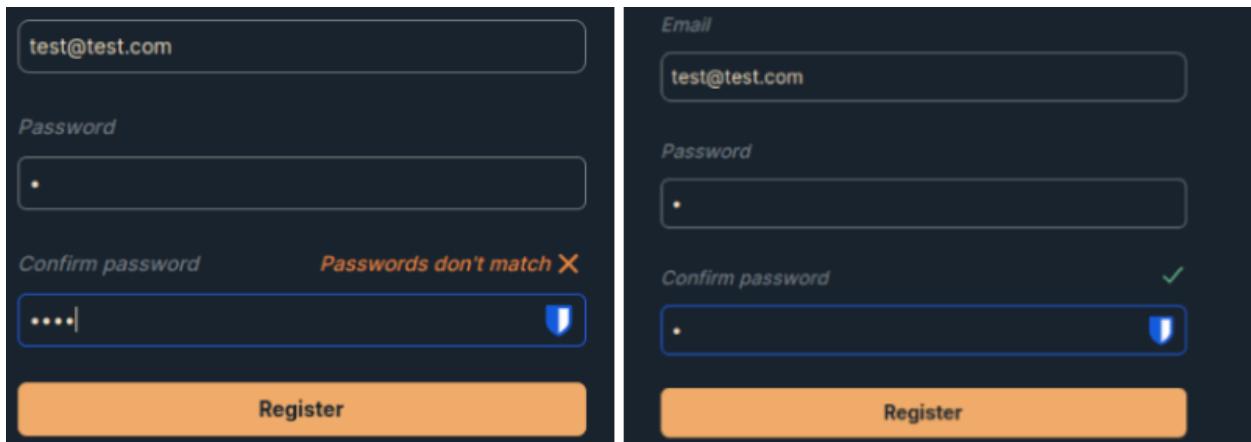
/user/register

A controlled form at `/account/register` corresponds with this endpoint form that accepts user input for an email and password, with the value of each input binding to a React state, which is updated on a change of value. The password confirmation field that attempts to mitigate the creation of an account that includes a typo in the password field.



The screenshot shows a web browser window for 'VolcanoDB' with the URL 'localhost:5173/account/register'. The page has a dark blue header with the 'VolcanoDB' logo and navigation links for 'Home', 'Volcano List', and 'Account'. Below the header is a 'Register' form. It contains three input fields: 'Email' (with placeholder 'test@test.com'), 'Password' (with placeholder '••••'), and 'Confirm password' (with placeholder '••••'). A large orange 'Register' button is at the bottom. At the very bottom of the page, there is a small link 'Already have an account? [Login here](#)'.

The base registration form

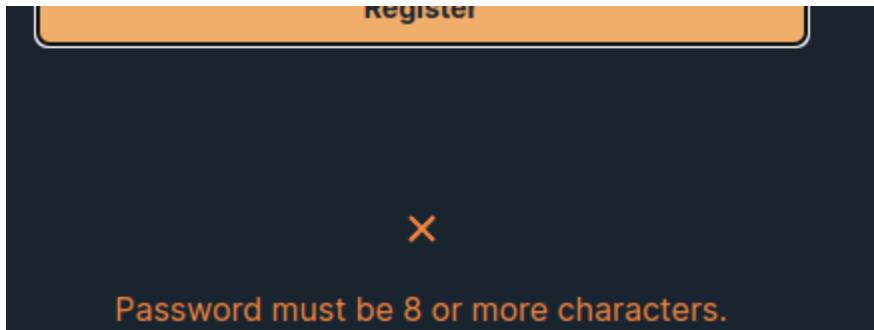


The image displays two side-by-side screenshots of the registration form. Both screenshots show the same inputs: 'Email' (test@test.com), 'Password' (a series of dots), and 'Confirm password' (a series of dots). In the left screenshot, a red error message 'Passwords don't match X' is displayed above the 'Confirm password' field, indicating a validation error. In the right screenshot, a green checkmark is displayed next to the 'Confirm password' field, indicating that the two entries match. Both screenshots feature an orange 'Register' button at the bottom.

Password intent check - non-matching vs. matching confirmation entries

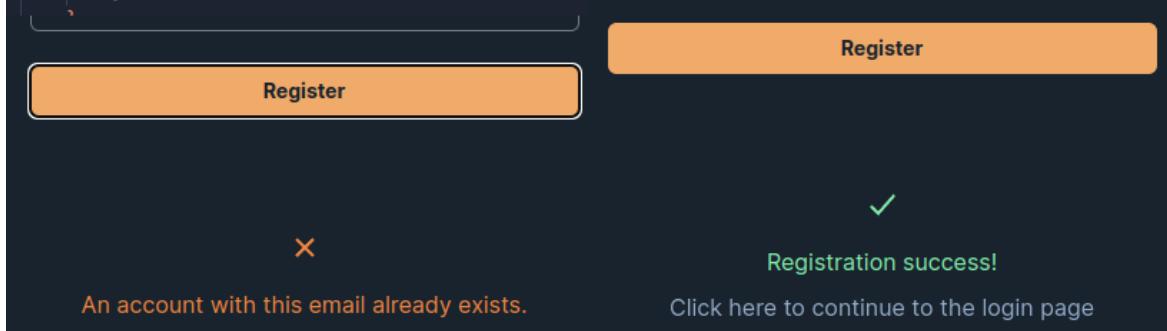
/user/register (cont.)

The form checks for a minimum password length of 8, and returns user-friendly status information to the user depending on the server's response.



A password entry with an invalid length; note that this can be bypassed by making a direct POST request to the API (e.g with a tool like `cURL`), so in an ideal scenario, password restrictions like this would also be accounted for on the server.

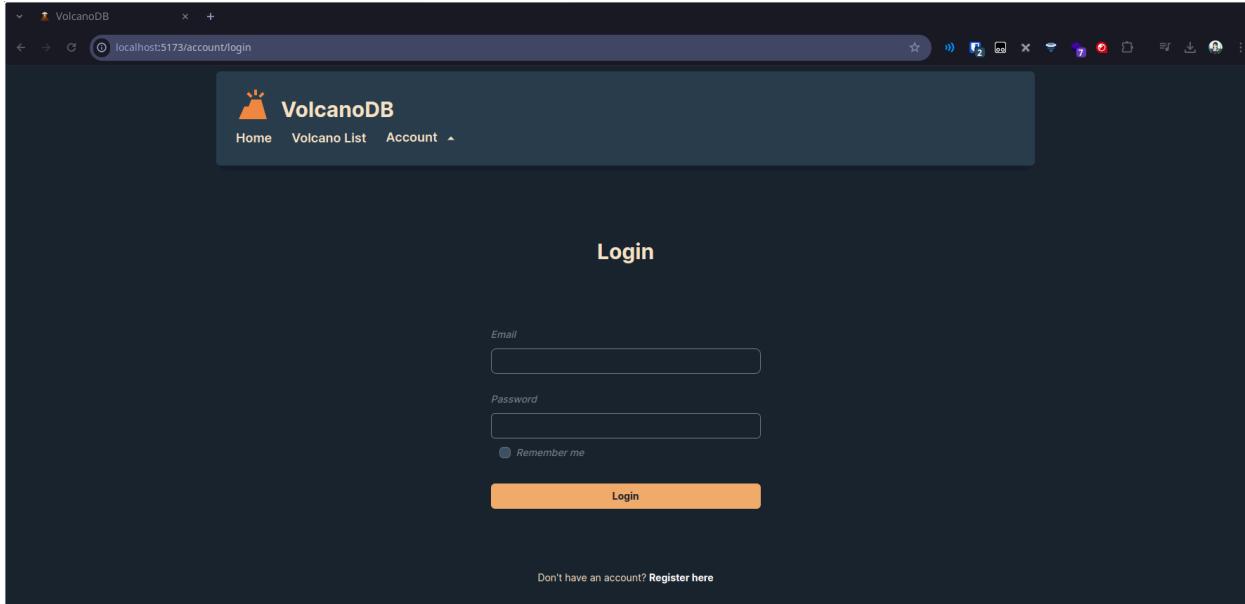
```
try {
  const result: void | Response = await postToApi(endpoint, data);
  if (result) {
    switch (result.status) {
      case 409:
        setRegisterResult('An account with this email already exists.');
        break;
      case 400:
        setRegisterResult('Registration requires both an email and password.');
        break;
      case 201:
        setRegisterResult('Registration success!');
        break;
      default:
        setRegisterResult(
          'An unknown issue occurred during registration. Please try again shortly.'
        );
        break;
    }
  }
}
```



The server may respond to invalid POST requests with a range of vague and/or jargon-like messages, so these are reinterpreted via the responses status code so the user can better understand why their request has returned an error, and adjust accordingly.

/user/login

The client login endpoint (at `/account/login`) functions similarly to the register endpoint, using a controlled form to post values to the corresponding `/user/login` API endpoint, handling a small range of server responses.



The base login page

/user/login (cont.)

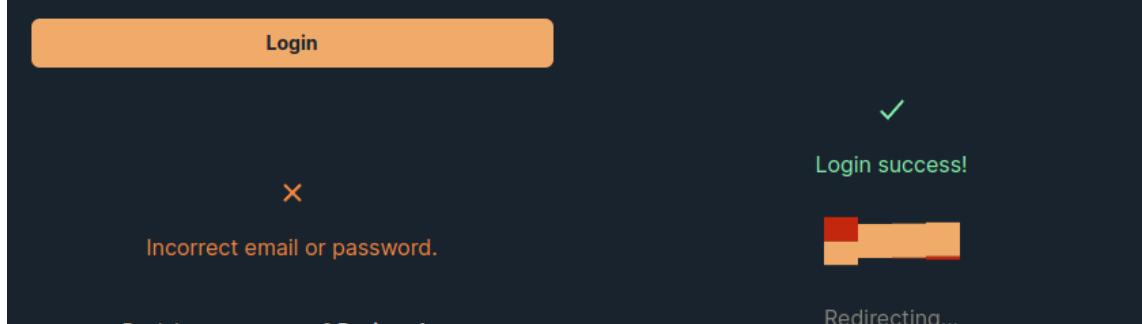
On a successful login, the server provides a JWT in its response; this token is stored in the browser's local storage to facilitate session persistence between browser sessions - note that, for security purposes, JWTs are not stored like this in production environments and should instead be stored as an `HttpOnly` cookie (where the Javascript runtime cannot read or write data); for the purposes of this example application, however, real user data is not handled. After token storage, unauthenticated data pulled from the API is cleared, and the user is redirected to the `/account/me` page.

```
try {
  const result: void | Response = await postToApi(endpoint, data);
  if (result) {
    switch (result.status) {
      case 200:
        try {
          await storeToken(result); // pass successful login response to JWT storage function
          setLoginResult('Login success!');
          reset(data);
          await sleep(500); // pause execution to allow user to read feedback
          navigate('/account/me');
        } catch (err) {
          console.error('error while parsing auth token => ', err);
        }
        break;

      case 401:
        setLoginResult('Incorrect email or password.');
        break;

      case 400:
        setLoginResult('Logging in requires both an email and password.');
        break;

      default:
        setLoginResult(
          'An unknown issue occurred during login. Please try again later.'
        );
        break;
    }
  }
} catch (err) {
```



Example response handling based on the code returned from the server - functionality is similar to the register page.

Modules Used

Using [Vite's react-ts template](#) (i.e. Vite is used instead of Webpack to bundle React code, which is built with Typescript, a Javascript superset that implements strong types) as a base for the application. The following [npm](#) modules were then installed on top of this for additional functionality:

Vite	Javascript bundler	https://github.com/vitejs/vite
React Router	Client side routing - re-renders just the changed parts of a web application when the user moves to a new page	https://github.com/remix-run/react-router
Tailwind CSS	CSS framework that implements a set of pre-defined "utility" classes that are implemented directly inside HTML markup	https://github.com/tailwindlabs/tailwindcss
Tailwind Forms	A Tailwind plugin which overrides default form styling for Tailwind integration	https://github.com/tailwindlabs/tailwindcss-forms
React icons	A large range of icons that can be imported into React components as React components	https://github.com/react-icon/react-icons
Ag-grid-react	Exposes a simple API to implement an interactive data grid	https://github.com/ag-grid/ag-grid
Pigeon Maps	Exposes a simple API to implement an interactive map	https://github.com/mariusandra/pigeon-maps

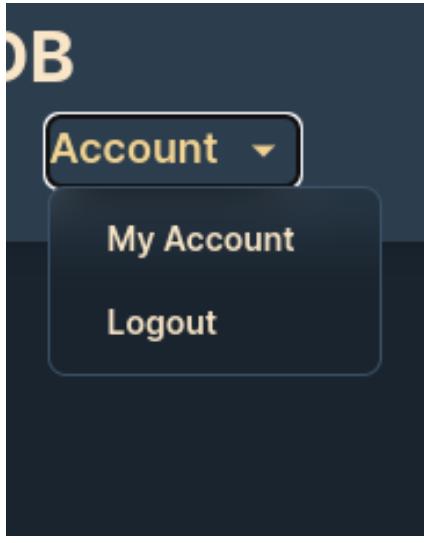
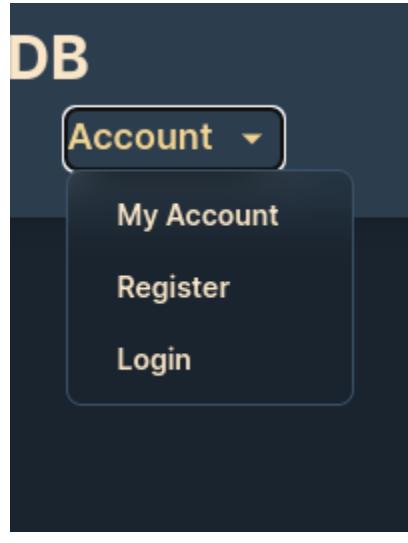
Application Design

Navigation & Layout

Generally, the application is designed to be simple, intuitive, and familiar:

- Tailwind CSS helps to maintain a consistent size and color palette throughout the application
- UI elements behave according to user muscle memory; for example, the dropdown in the navigation bar can be dismissed by clicking elsewhere in the application
- Buttons are positioned according to their intended direction; buttons that will move a user deeper into the application are positioned on the right, whereas buttons on the left direct a user back towards a starting point (with the same progression as if a user were to click the `back` arrow in their browser).
 - Interactive elements are highlighted by a color, opacity, or size change when hovered by a user.

A user's account authentication state is visible when opening the `Account` dropdown.

Logged in/authenticated	Logged out/unauthenticated
	

To reiterate earlier sentiments, the website is meant to be used in a way that allows a user to navigate with increasingly specific requests - it is intended to be used in the following order:

1. Search/browse countries with available volcano data,
2. Choose a country, displaying a list of volcanoes available for that country, with a small range of extra information on each volcano,
3. Choose a volcano to view specific details for that volcano,
4. Filter specific population data for that country (if the user is authenticated), either by hovering over the bars on the graph or making a selection from the select box menu.

Usability and Quality of Design

Given the above criteria, the application's design is mostly in-line with what a user would be familiar with given current web application design trends. For example, the difference between buttons from UI component libraries (the form submission buttons at the `/account/login` and `/account/register` endpoints) in comparison to the custom navigation menu items and other buttons may be an issue with some users as they might appear too much like plain text. Similarly, the spacing of various elements can leave the page feeling bare, which might make pages appear as if they are still loading content when they are not.

The Nav bar along the top is constrained to a portion of the screen to make links more obvious; originally, this ran along the entire length of the screen, but the addition of content on the landing page drew attention away from this aspect (which may still somewhat be the case).

Aside from this, the general page-to-page navigational flow is intended to gradually dial into a user's desired dataset, and the difference between interactive elements vs static elements is given using a combination of CSS color-opacity changes alongside cursor changes to provide contextual hints to a user. For example, users can hover their cursor over an element to determine whether it can be clicked, with the 'direction' or resulting action being described through the button's text. Each page should contain a link that either moves the user

- Further into each site area, increasing the specificity of the displayed data, to the end goal of a volcano's data,
- Backward into previously-visited areas; users are taken back to *their previously visited* page, rather than the previous page in the site's page flow (i.e, a link to a volcano from the landing page will take a user to that volcano's dataset, and the 'return' button on that volcano dataset page will move that user back to the landing page, instead of throwing them into the volcano list, which may be confusing or disorienting for a new user).

Accessibility

In terms of the accessibility of the content of this site, most custom elements require significant work. While Tailwind can be relied upon to provide a small range of accessibility utilities for screen readers and color content, as well as conditional ARIA (Accessible Rich Internet Applications) styling, these - along with most of the items on the W3C accessibility checklist - need to be implemented by hand based on the application's structure.

For example, as SPA routing means that entire page reloads won't occur, and while care has been taken to avoid jarring screen layout shifts or rapid and drastic color changes, some content in *Volcano DB* will still flicker when navigating between routes while content is fetched and DOM hydration occurs. In this case, SvelteKit, a relatively new Javascript framework, features built-in animations and hooks to fade content in and out as the page changes. As React is comparatively not so much a Javascript framework, similar functionality must be handled by a separate package (or hand-rolled by a developer), which is extra time spent programming or learning the API for a package.

Furthermore, *Volcano DB*'s HTML markup is constructed mostly via the use of `<div>` HTML elements, which, in the absence of CSS and scripting, means it is difficult to determine the structure of the page. Similarly, many of the buttons rely on Javascript and color changes to indicate to a user that they are a button, so users with Javascript disabled or a sight disability may find it difficult to determine the purpose of a range of UI elements.

Technical Description

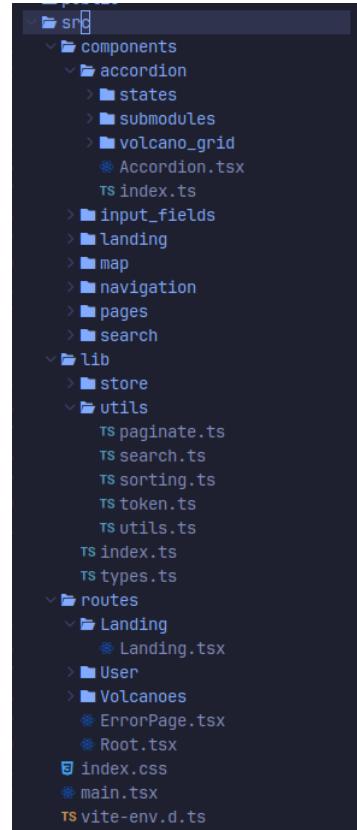
Architecture

The source code for *Volcano DB* is organized into three main directories:

- components/
 - lib/
 - routes/

The `components` and `routes` directories should be reasonably self-explanatory; portions of the UI are broken down into components and component sub-modules for improved code maintenance and reusability. Components may not necessarily be constrained to a particular route, so they are decoupled and stored separately.

Components are constructed into a coherent UI on each page, which are in the `routes` directory, where a route reflects a URL. The NavBar and Footer components don't change from route to route, so they are rendered in the `Root.tsx` route, with "main" content rendered handled by React Router from `main.tsx`.



`lib` houses reusable function content - things like `fetch` calls, cache `context` providers, and Typescript types, which would crowd a component with logic that benefit from reuse in other components.

To compliment the application's modularity, Vite, ESLint and Typescript are all configured in their respective config files (see `/vite.config.ts`, `/tsconfig.json` and `/.eslint.cjs`) such that an alias is set to use an '@' specifier to import the `src/` root directory (Vite, n.d.).

Each module is then re-exported in its own `index.ts` file. This allows a given module to be imported from, for example, '`@/lib`', or '`@/components/{component}`', which sacrifices some extra boilerplate code for simplified module management:

```
... > 8 ts:types > 9 * Login > 10 * ErrorPage > 11 * VolcanoRoot > :: 12 * Accordion > 13 * AccordionPanel
8   >■ navigation           src > routes > Volcanoes > * VolcanoRoot.tsx
7   >■ pages
6   >■ search
5 >■ lib
4  >■ routes
3   >■ Landing
2   >■ User
1   >■ Volcanoes
31  * VolcanoRoot.tsx
1  * VolcanoPage.tsx
4 import { SpiralSpinner } from 'react-spinners-kit';
5 import { useState, useEffect } from 'react';
6 import { useStore, fetchFromApi } from '@lib';
7 import { VolcanoRoot, Accordion } from '@components/accordion';
8 import { useParams } from 'react-router';
9 // import Accordion from '@components/accordion/Accordion';
10
11 const VolcanoRoot: React.FC = (): React.ReactElement => {
12   const [ data, isLoading, setisLoading ] = useState();
13   const [ params, setParams ] = useParams();
14   const [ accordion, setAccordion ] = useState();
15
16   const handleAccordionChange = (id: string) => {
17     setAccordion(id);
18   }
19
20   const handleVolcanoClick = (id: string) => {
21     setParams({ id });
22   }
23
24   const handleVolcanoLoad = () => {
25     setisLoading(true);
26     fetchFromApi('volcanoes')
27       .then(data => {
28         setisLoading(false);
29         setParams({ id: data[0].id });
30       })
31       .catch(error => {
32         setisLoading(false);
33       });
34   }
35
36   return (
37     <VolcanoRoot>
38       <VolcanoPage
39         <VolcanoList
40           <VolcanoCard
41             <VolcanoImage
42             <VolcanoTitle
43             <VolcanoDescription
44             <VolcanoActions
45           >/VolcanoCard>
46         >/VolcanoList>
47       >/VolcanoPage>
48     </VolcanoRoot>
49   );
50 }
```

Test plan

(screenshot appendices from page 24)

Application Area	Task	Expected Outcome	Result	Screenshot appendix
Landing page	View a random volcano (logged out)	Taken to the corresponding volcano page <i>without</i> population data	PASS	01
Landing page	View a random volcano (logged in)	Taken to the corresponding volcano page <i>with</i> population data	PASS	02
Account - login	Use valid details to log into an account using test credentials: email: ' mike@gmail.com ' , password: 'password'	The account is logged into, a token is stored in the browser's local storage, and the user is redirected to the '/account/me' client endpoint	PASS	03
Account - logout	Logout using the 'logout' button in the navbar drop down	User is logged out, the JWT in the browser's local storage is cleared, cached volcano data requiring authorization is reset, and the user is redirected to /account/login	PASS	04, 05
Account - login	Login with invalid email	Request sent to API, which will return a 401 status. 'Incorrect email or password' message is displayed to the user	PASS	06
Account - login	Login with valid email but invalid password	Request sent to API, which will return a 401 status. 'Incorrect email or password' message is displayed to the user	PASS	07
Account - login	Send a login request with no email/password	HTML form inputs carry 'required' attribute. Requests with Chrome/Firefox will highlight missing fields (no form submission).	PASS	08
Account - register	Send a valid registration request: email: ' askdjhflaksdjhfklkasjdhfkasdhf@a.com ', password: 'password'	Request is sent to API, 201 response returned. 'Registration success' message displayed, user is requested to log in.	PASS	09
Account - login	Login to account with newly-created credentials email: ' askdjhflaksdjhfklkasjdhfkasdhf@a.com ',	The account is logged into, a token is stored in the browser's local storage, and the user is redirected to the '/account/me' client endpoint	PASS	10

	password: 'password'			
Account - register	Attempt to register with an existing email	Request is sent to API, 409 response returned. 'An account with this email already exists' message displayed to user.	PASS	11
Account - register	Attempt to register with a password shorter than 8 characters	Request is filtered on the client. 'Password must be 8 or more characters' displayed to user.	PASS	12
Account - register	Attempt to register with invalid confirmation password	Request is filtered on the client. 'Password doesn't match the confirmation' is displayed to user.	PASS	13
Volcanoes - countries	Search for a country	List of countries is filtered to contain only those approximately matching the search query	PASS	14
Volcanoes - countries	Delete search term	Full list of countries is restored	PASS	15
Volcanoes - countries	Change country ordering	List of countries are sorted in ascending [Z-A] order	PASS	16
Volcanoes - countries	Change country page	Countries from the desired page are returned	PASS	17
Volcanoes - countries	Attempt to change to an invalid page (e.g clicking 'next' on the final page)	No state change	PASS	18
Volcanoes - Volcano list	Navigate to a country and view its list of volcanoes	Shows a small overview of each volcano in a table, with a button to view detailed data in the far right column	PASS	19
Volcanoes - details	View details on an arbitrary volcano while logged out	Shows all details other than population data and chart	PASS	20
Volcanoes - details	View details on an arbitrary volcano while logged in	Shows all details including population data and chart	PASS	21

Difficulties, Exclusions, Unresolved Errors

Where all 'critically important' functions were implemented, a small number of non-critical features were not finalized or able to be implemented on time:

- A 'Remember me' checkbox on the login form,
- An indicator in the NavBar to display whether a user is logged in or not,
- Searching and pagination features for the volcanoes grid (similar to that of the countries list),
- A way to search for volcanoes without first knowing the country (though this may not have been feasible due to API limitations).

This is mostly a case of time management, where more time was allocated to perfecting non-critical aspects of critical functions (e.g, using a basic fuzzy-matching search over a direct equality check, or registration password confirmation fields) rather than implement the similarly non-critical aspects above.

Relating to time restrictions, the application has a tendency for an over-reliance on `useEffect()` hooks where they may not have been necessary - this indicates that some aspects of the application's functionality are less efficient than the optimal case. Greater scrutiny into which functions truly require an effect and refactoring and retesting these components and functions would be ideal, though the smaller scale of the application means it was able to run responsively and without noticeable latency.

The codebase also uses naming conventions that may not have been named optimally - for example, the `Accordion` component and its sub-modules was initially implemented as an accordion that could be opened and closed to reveal or hide its contents. Ultimately, later changes to `/volcanoes` implementation to handle the user's selected country inside the URL means that this accordion-like functionality was dropped for simplicity. The remaining artifact - a slightly 'deceptive' naming scheme - means this might be confusing for code maintainability reasons, and would be ideal as a candidate for a simple fix in future iterations.

Similarly, the functionality for the NavBar's 'click of to dismiss' functionality still requires a vanilla `document.addEventListener()` function to attach an `onClick` attribute to the DOM as a whole; this was initially developed using a `composedPath()` to check a tree of elements after a click event was heard and find whether the click target was part of the drop down. This was changed to a React `useRef()` hook (code examples on the following page).

The component's initial design:

```
14 - const NavigationD: FC = (): ReactElement => {
15 -   const [dropdownOpen, setDropdownOpen] = useState('');
16 -
17 -   // dismiss open dropdown by clicking other section of the screen
18 -   const handleDismissClickout = (event: MouseEvent) => {
19 -     const path = event.composedPath();
20 -     const isClickInside = path.some((el: EventTarget) =>
21 -       (el as HTMLElement).classList?.contains('dropdown')
22 -     );
23 -     if (dropdownOpen === '' && !isClickInside) {
24 -       setDropdownOpen('');
25 -     }
26 -   };
27 -
28 -   // add/remove onclick listener to the dropdown on component mount/destroy
29 -   useEffect(() => {
30 -     document.addEventListener('click', handleDismissClickout);
31 -     return () => {
32 -       document.removeEventListener('click', handleDismissClickout);
33 -     };
34 -   }, []);
35 -
36 -   // dropdown toggle handler function
37 -   const handleClickDropdown = (parent: string) => {
38 -     dropdownOpen === parent ? setDropdownOpen('') : setDropdownOpen(parent);
39 -   };

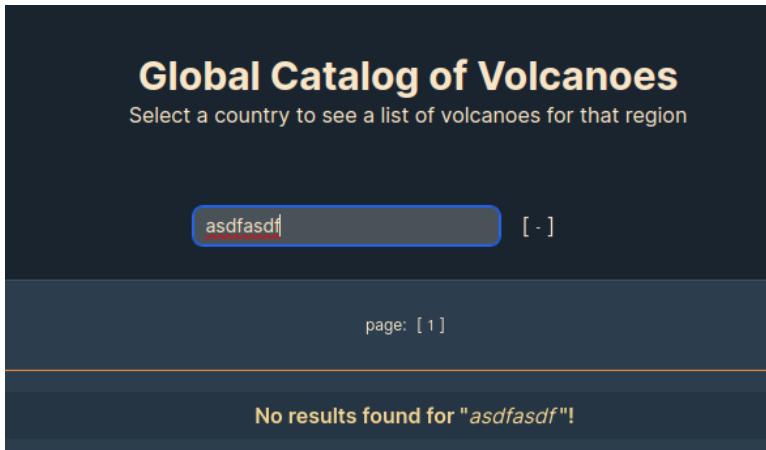
```

Refactored useRef() approach:

```
23 + const Navigation: React.FC = (): React.ReactElement => {
24 +   const [open, setOpen] = useState<string>('');
25 +   const deepRef = useRef<HTMLLIElement>(null);
26 +
27 +   // useRef to close the dropdown if the user clicks on a non-dropdown portion of the DOM
28 +   const clickOutside = (event: MouseEvent) => {
29 +     console.log(deepRef, event.target);
30 +     if (deepRef.current && !deepRef.current.contains(event.target as Node)) {
31 +       setOpen('');
32 +     }
33 +   };
34 +
35 +   // lifetime hook - attach/cleanup an event listener to the document to receive clicks from
36 +   // any element
37 +   useEffect(() => {
38 +     document.addEventListener('click', clickOutside);
39 +     return () => {
40 +       document.removeEventListener('click', clickOutside);
41 +     };
42 +   }, []);
43 +
44 +   const toggleDropdown = (identifier: string) => {
45 +     setOpen(open === identifier ? '' : identifier);
46 +   };

```

Aside from this, manual testing only turned up one remaining bug (though there are likely other edge cases that are yet to be found) - searching for a country and then clicking the reorder button causes the states of the two components to de-sync, where they would ideally be aware of each others' current states.



Here, the two components need to correctly synchronize as children of a common parent component, sharing state so that each function call and variable check can be done with consideration for the state of the sibling component.

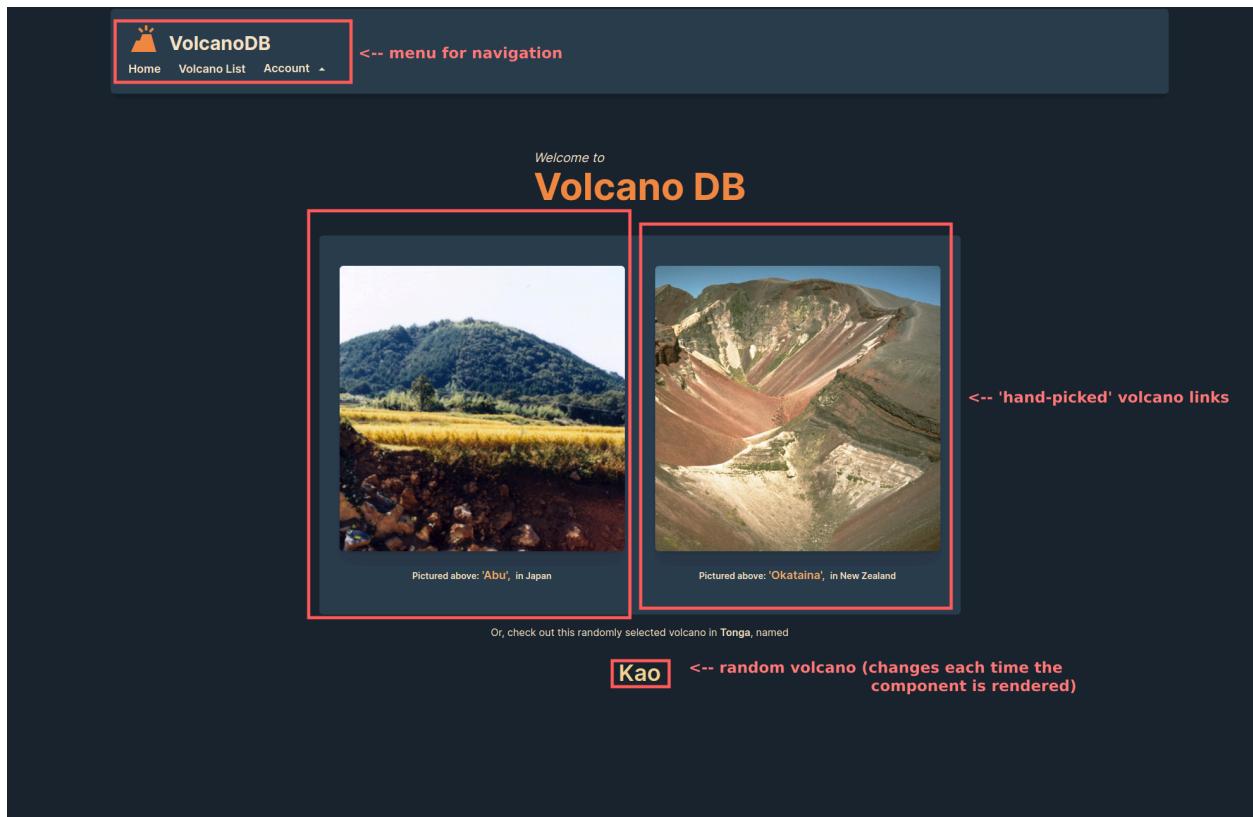
User Guide

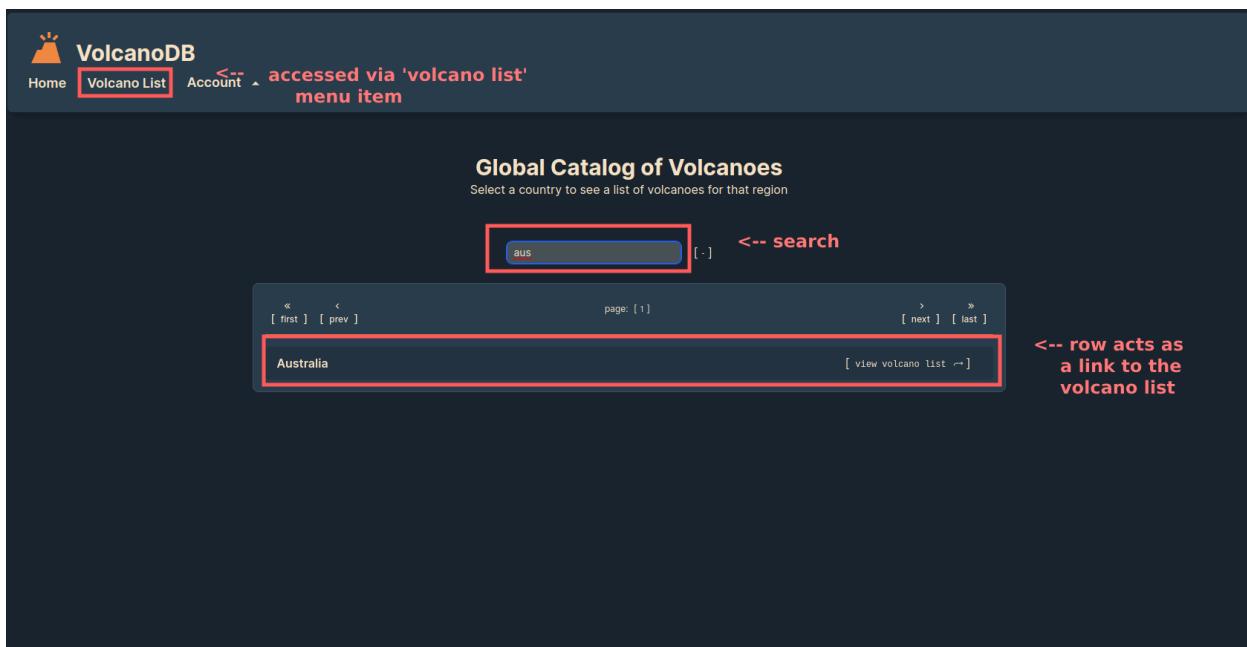
The two general audiences that *Volcano DB* is intended for may be one of two types of users:

- a) 'Just browsing' (e.g to browse the database at a whim - there is no desire to create an account and no specific search terms in mind),
- b) 'Data hunters' (e.g a user looking for specific information on volcanoes - this could be data about a specific volcano, or about the volcanic activity of the region, for example).

Use Case A

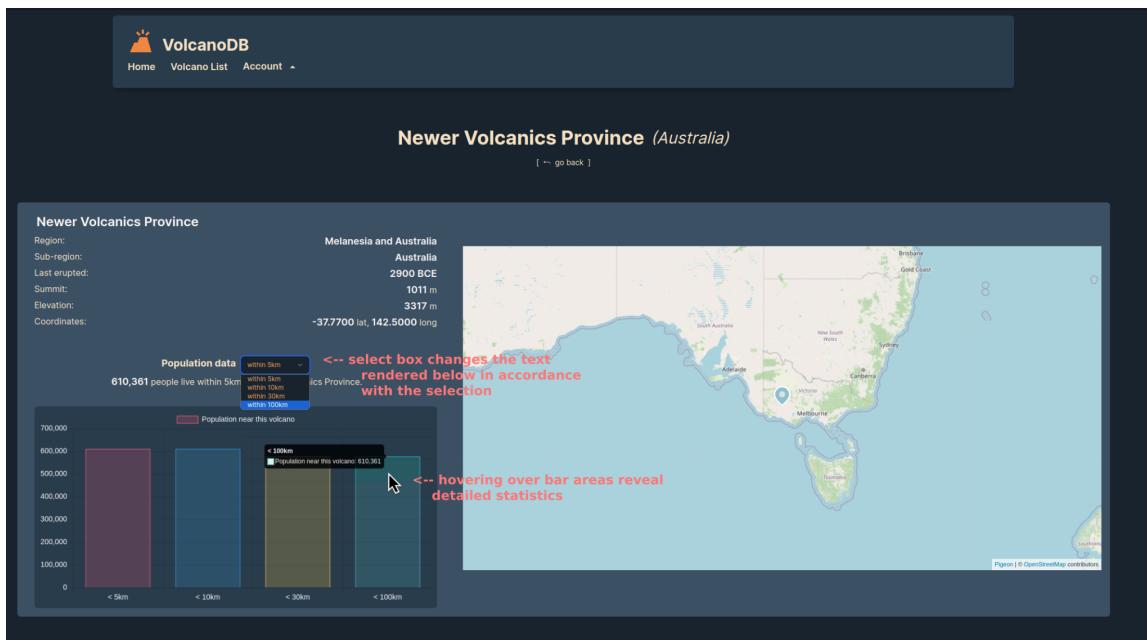
In *use case a*, the user is immediately presented with a way to access some random data - either via one of two images linking to hand-picked volcanoes, or a link to randomly chosen volcano. From there, this user might casually browse the catalog by country. This user represents a more relaxed approach to content consumption, with no active 'goal' in mind; here, *Volcano DB* immediately attempts to provide that user with interesting data from the landing page.



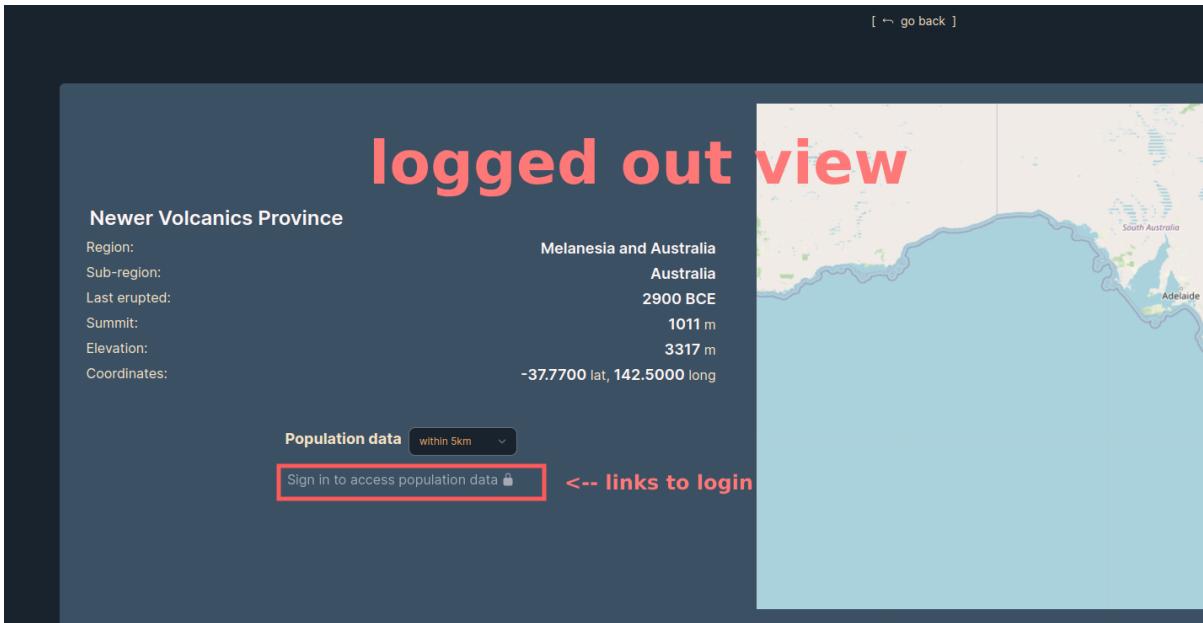


Use Case B

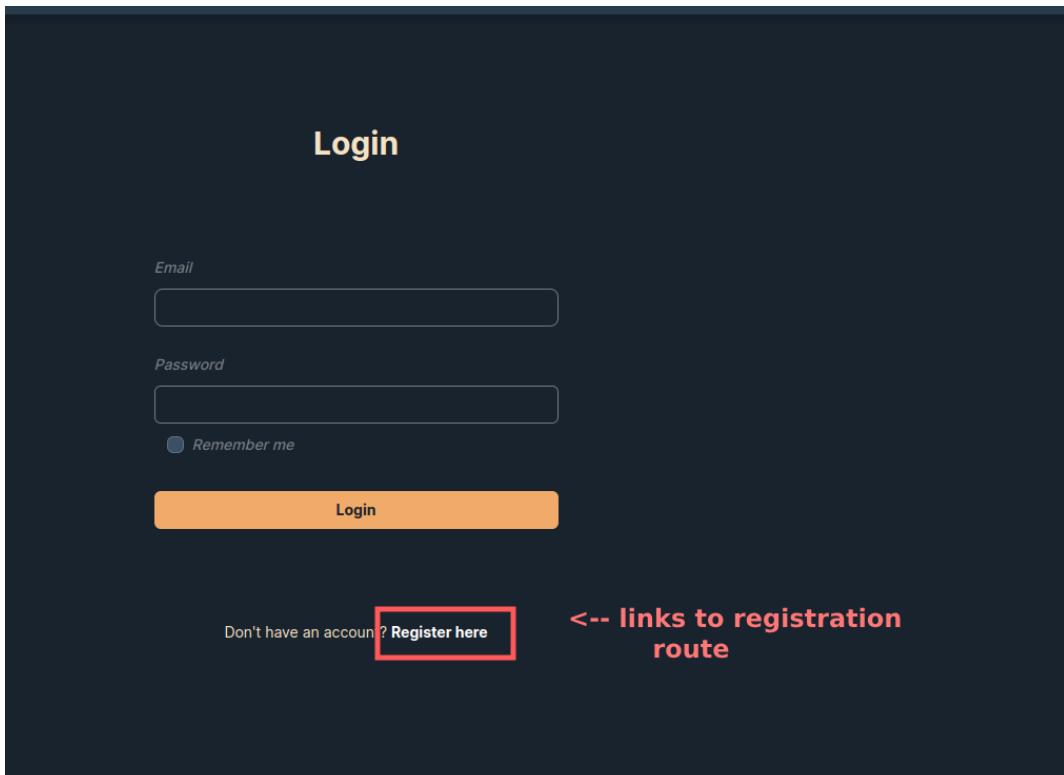
Otherwise, *use case b* allows a user to quickly access the countries list from the NavBar, filtering the list of countries with a search term or moving forward/backward through the paginated country list.



The user can then choose a country by clicking 'view volcano list', and continue to a chosen volcano from this list using the 'view data' button in the right-hand column.



If this user is not signed in and wants to view the account-gated population data, there is a link to login on the volcano data page to take the user to the client side `/account/login` endpoint, where they can either log in to an existing account. On arriving at this page, users without an account can easily access the register page via the link below the login form.



References

ag-grid. (n.d.). *ag-grid/ag-grid: The best JavaScript Data Table for building Enterprise Applications.*

Supports React / Angular / Vue / Plain JavaScript. GitHub. Retrieved May 9, 2024, from
<https://github.com/ag-grid/ag-grid>

mariusandra. (n.d.). *mariusandra/pigeon-maps: ReactJS Maps without external dependencies.* GitHub.
Retrieved May 9, 2024, from <https://github.com/mariusandra/pigeon-maps>

SvelteJS. (2023, September 17). *svelte/transition • Docs • Svelte.* Svelte. Retrieved May 10, 2024, from
<https://svelte.dev/docs/svelte-transition>

Tailwind Labs. (n.d.). *Forced Color Adjust.* Tailwind CSS. Retrieved May 10, 2024, from
<https://tailwindcss.com/docs/forced-color-adjust>

Tailwind Labs. (n.d.). *Screen Readers.* Tailwind CSS. Retrieved May 10, 2024, from
<https://tailwindcss.com/docs/screen-readers>

Tailwind Labs. (n.d.). *tailwindlabs/tailwindcss: A utility-first CSS framework for rapid UI development.*
GitHub. Retrieved May 9, 2024, from <https://github.com/tailwindlabs/tailwindcss>

Tailwind Labs. (n.d.). *tailwindlabs/tailwindcss-forms.* GitHub. Retrieved May 9, 2024, from
<https://github.com/tailwindlabs/tailwindcss-forms>

Vite. (n.d.). *vitejs/vite: Next generation frontend tooling. It's fast!* GitHub. Retrieved May 9, 2024, from
<https://github.com/vitejs/vite>

Vite. (n.d.). *Vite: Shared Options - Resolve Alias.* Vite. Retrieved May 10, 2024, from
<https://vitejs.dev/config/shared-options.html#resolve-alias>

Screenshot Appendices

(starting on the following page)

Appendix 01

Welcome to
Volcano DB

Pictured above: 'Abu', in Japan

Pictured above: 'Okataina', in New Zealand

Or, check out this randomly selected volcano in United States, named

Mauna Kea

Mauna Kea (United States)

{ ← go back }

Mauna Kea

Region:
Sub-region:
Last erupted:
Summit:
Elevation:
Coordinates:

Hawaii and Pacific Ocean
Hawaiian Islands
2460 BCE
4205 m
13796 m
19.8200 lat, -155.4700 long

Population data within 5km

Sign in to access population data

Map showing the location of Mauna Kea on the Big Island of Hawaii.

Appendix 02

The screenshot shows the VolcanoDB homepage with a dark theme. At the top, there is a navigation bar with links for Home, Volcano List, and Account (which is currently active, showing a dropdown menu for My Account and Logout). Below the navigation is a welcome message: "Welcome to Volcano DB". Two volcano images are displayed: one of 'ADU' in Japan (a green, forested mountain) and one of 'Okataina' in New Zealand (a multi-colored, stratified mountain). Below the images, text reads "Pictured above: 'ADU', in Japan" and "Pictured above: 'Okataina', in New Zealand". A link at the bottom says "Or, check out this randomly selected volcano in Chile, named Lonquimay".

The screenshot shows the Lonquimay (Chile) profile page. The top navigation bar includes links for Home, Volcano List, and Account. The main title is "Lonquimay (Chile)". Below the title is a map of South America with Chile highlighted. On the left, there is a sidebar with detailed information about the volcano, including its Region (South America), Sub-region (Central Chile and Argentine), Last erupted (1990 CE), Summit elevation (2832 m), Elevation (9291 m), and Coordinates (-38.3790 lat, -71.5860 long). A population chart titled "Population data" shows the number of people living within 5km of the volcano, with a value of 96. The chart has four categories: < 5km, < 10km, < 30km, and < 100km.

Appendix 03

The image contains two screenshots of a web application named VolcanoDB.

Screenshot 1: Login Page

This screenshot shows the login interface of the VolcanoDB application. The title bar says "localhost:5173/account/login". The header includes the VolcanoDB logo and navigation links for Home, Volcano List, and Account. The main area has a dark background with the word "Login" in white. It features input fields for "Email" (containing "mike@gmail.com") and "Password" (containing "*****"), a "Remember me" checkbox, and a large orange "Login" button.

Screenshot 2: Account Page and DevTools Application Panel

This screenshot shows the account page for a user named "Mike". The title bar says "localhost:5173/account/me". The header includes the VolcanoDB logo and navigation links for Home, Volcano List, and Account. The main area displays "Hello, Mike!" and a "Log out" link. On the right, the Chrome DevTools Application panel is open, showing the "Storage" section. Under "Local storage", there is an entry for "http://localhost:5173" containing a JSON object: {"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6..."}.

Appendix 04

(Pre-logout)

The screenshot shows a web browser window titled "VolcanoDB" displaying the "Atakor Volcanic Field (Algeria)" page. The URL is localhost:5173/volcanoes/Algeria/90. The page includes a map of North Africa with a blue marker indicating the location of the Atakor Volcanic Field near Tindouf, Algeria. To the left of the map is a detailed information card for the volcano:

Atakor Volcanic Field
Region: Africa and Red Sea
Sub-region: Africa (northern)
Last erupted: Unknown
Summit: 2918 m
Elevation: 9573 m
Coordinates: 23.3300 lat, 5.8300 long

Below the card is a "Population data" section with a dropdown menu set to "within 5km". It displays a bar chart showing the population distribution near the volcano:

Distance Range	Population
< 5km	0
< 10km	0
< 30km	0
< 100km	120,000

The browser's developer tools are open, showing the Network tab with a request to <http://localhost:5173>. The request details show the token used for authentication.

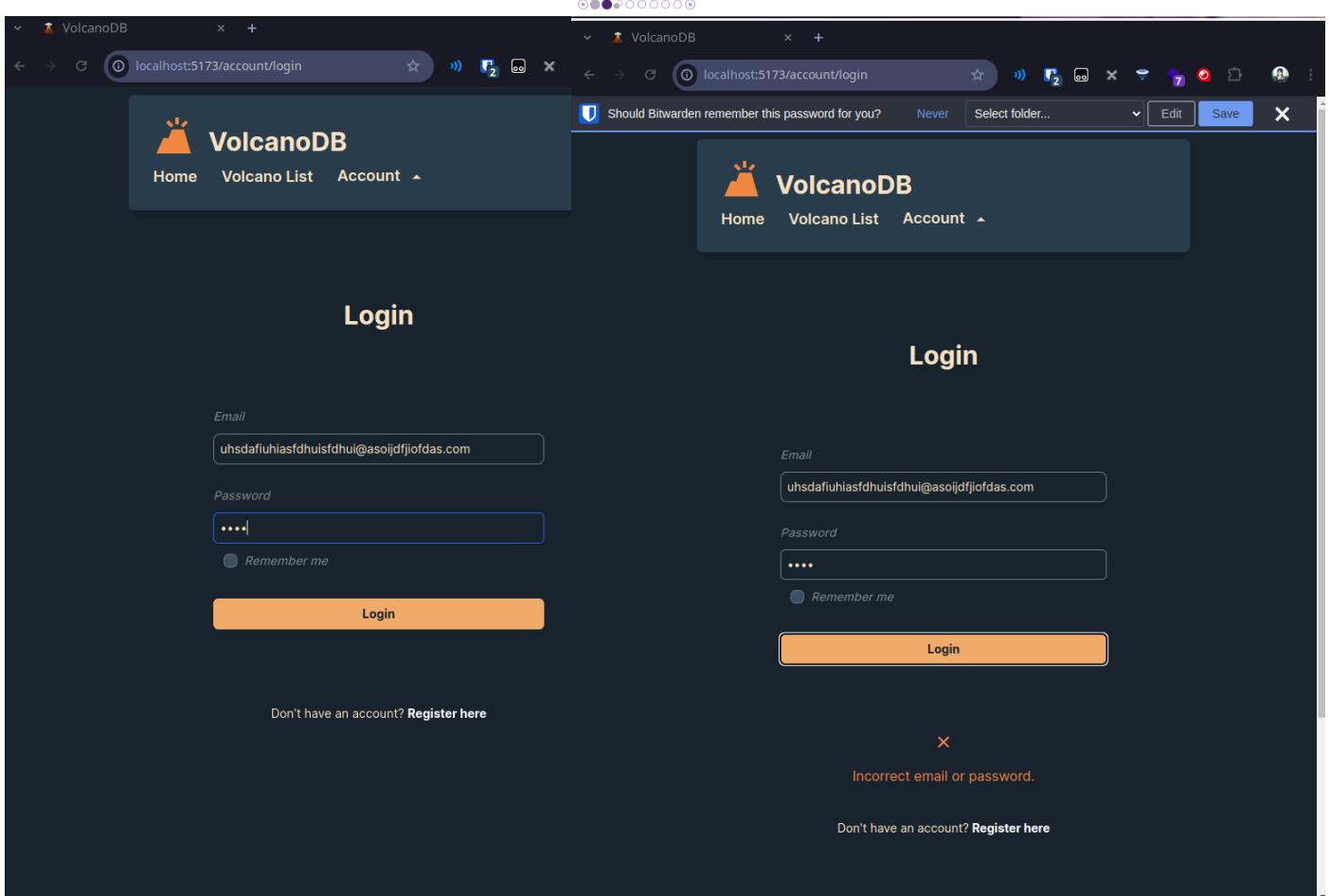
Appendix 05

(Post-logout)

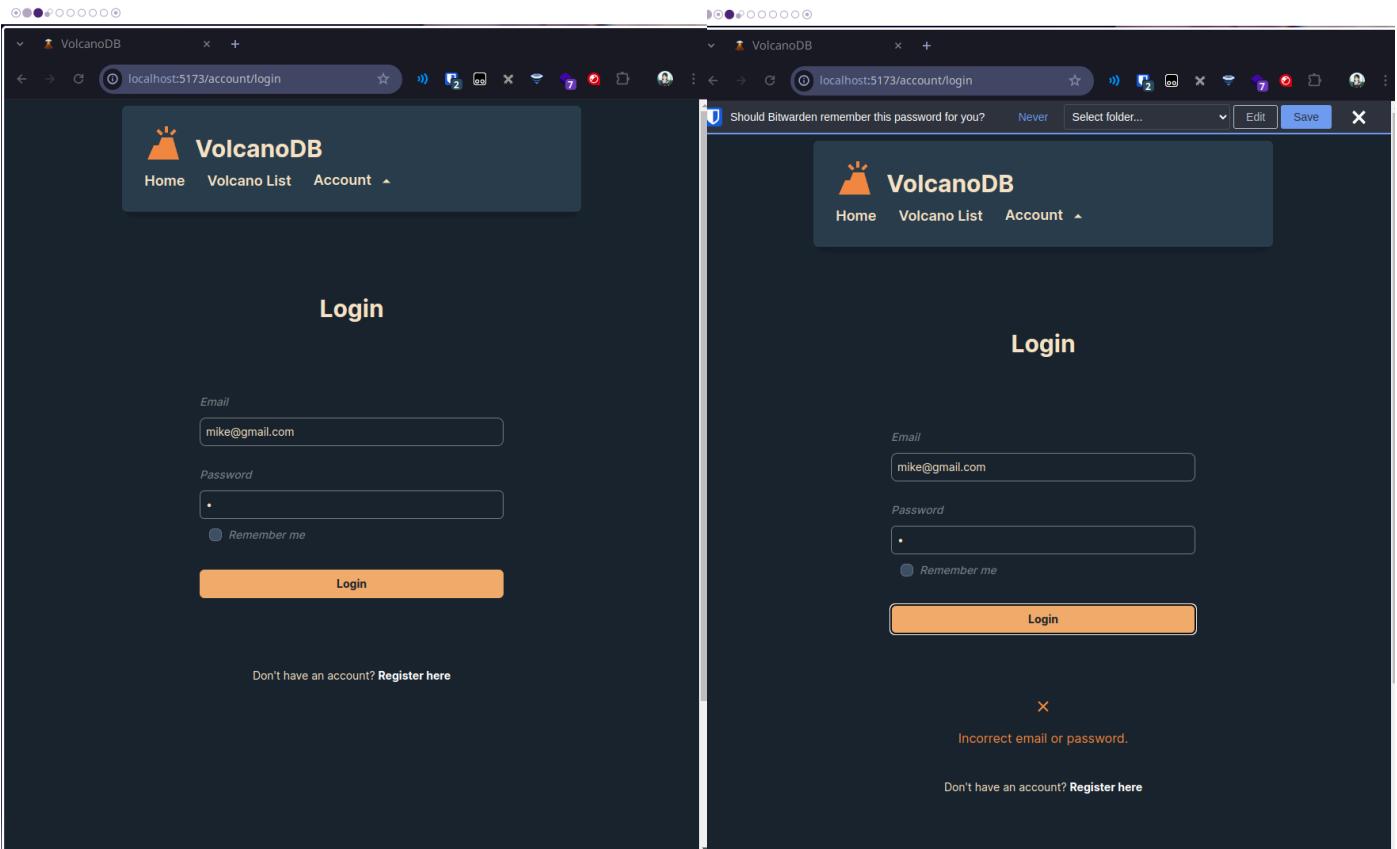
The screenshot shows a browser window for the VolcanoDB application at localhost:5173/account/login. The left side displays the 'Login' page with fields for 'Email' and 'Password', a 'Remember me' checkbox, and an orange 'Login' button. The right side shows the Chrome DevTools Application tab open, with the Storage section expanded to show Local storage for the domain <http://localhost:5173>. The bottom of the DevTools panel displays several developer settings checkboxes.

The screenshot shows a browser window for the VolcanoDB application at localhost:5173/volcanoes/Algeria/90. The page title is 'Atakor Volcanic Field (Algeria)'. It features a map of North Africa with a yellow-shaded area indicating the Atakor Volcanic Field. A callout box highlights the location near Tommanras. To the left, a sidebar provides detailed information about the field, including its region (Africa and Red Sea), sub-region (Africa (northern)), last eruption date (Unknown), summit height (2918 m), elevation (9573 m), and coordinates (23.3300 lat, 5.8300 long). A dropdown menu for 'Population data' is set to 'within 5km', and a link to 'Sign in to access population data' is present.

Appendix 06



Appendix 07



Appendix 08

Login

Email

delugelocalcl ! Please fill out this field.

localhost hello

Login

This screenshot shows a dark-themed login interface. At the top is a large orange "Login" button. Below it is a text input field labeled "Email". A validation error message, "Please fill out this field.", is displayed in a white box with a red exclamation mark icon above the input field. To the left of the input field is the word "Email" and to the right is a blue shield icon. Below the input field is a dropdown menu with two items: "delugelocalcl" and "localhost". The "localhost" item has a "hello" option expanded. At the bottom is another orange "Login" button.

Login

Email

a@a.com

Password

delugelocalcl ! Please fill out this field.

localhost hello

This screenshot shows a dark-themed login interface. At the top is a large orange "Login" button. Below it is a text input field labeled "Email" containing "a@a.com". Below the "Email" field is a password input field labeled "Password", which is currently empty. A validation error message, "Please fill out this field.", is displayed in a white box with a red exclamation mark icon above the password field. To the left of the password field is the word "Password" and to the right is a blue shield icon. Below the password field is a dropdown menu with two items: "delugelocalcl" and "localhost". The "localhost" item has a "hello" option expanded. At the bottom is another orange "Login" button.

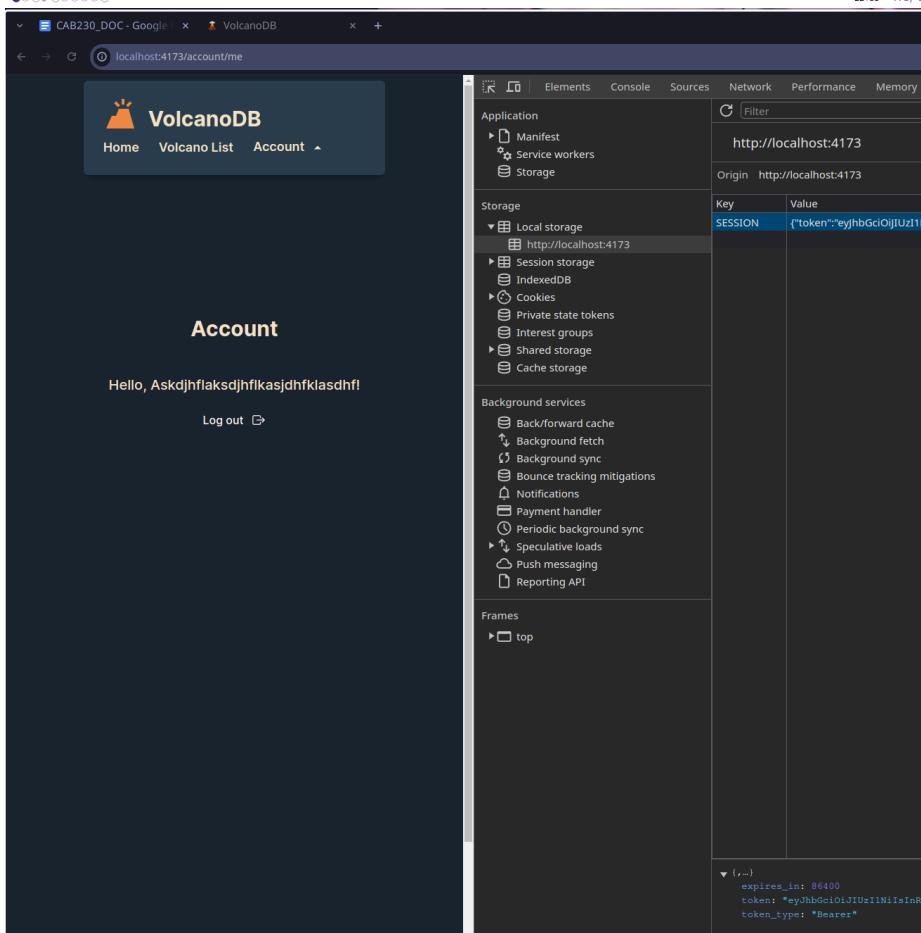
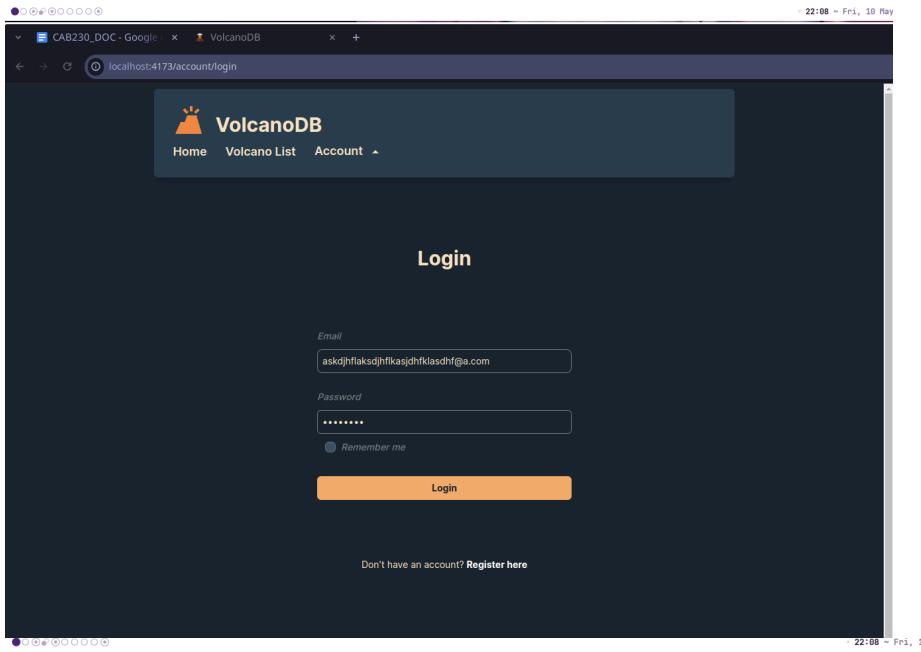
Appendix 09

The screenshot shows a browser window with the URL `localhost:4173/account/register`. The page title is "VolcanoDB" with a volcano icon. The main heading is "Register". There are three input fields: "Email" (containing `askdjhflaksdjhfikasjdhfklasdhf@a.com`), "Password" (containing `*****`), and "Confirm password" (containing `*****`). Below the fields is an orange "Register" button. At the bottom left, there is a link "Already have an account? [Login here](#)". On the right side of the browser, the developer tools' "Console" tab is open, showing the following log entry:

```
Fetch finished loading: POST *http://localhost:4173/account/register*
```

This screenshot is identical to the one above, showing the registration page and the developer tools console output. However, it includes a success message at the bottom of the page: "✓ Registration success! Click here to continue to the login page".

Appendix 10



Appendix 11

The image displays three vertically stacked screenshots of a mobile application's registration screen. The top two screenshots show the registration process, while the bottom one shows an error message.

Screenshot 1 (Top): The screen title is "Register". It contains fields for "Email" (mike@gmail.com), "Password" (represented by six dots), and "Confirm password" (also represented by six dots). A green checkmark icon is positioned next to the "Confirm password" field. A large orange "Register" button is at the bottom.

Screenshot 2 (Middle): This screenshot is identical to the first one, showing the registration fields and the "Register" button.

Screenshot 3 (Bottom): This screenshot shows the same registration fields, but the "Email" field now contains "mike@gmail.com" and has a red "X" icon to its left. Below the fields, an error message reads "An account with this email already exists." At the bottom, there is a link "Already have an account? [Login here](#)".

Appendix 12

Register

Email

Password

Confirm password ✓

Register

X

Password must be 8 or more characters.

Already have an account? [Login here](#)

Already have an account? [Login here](#)

Appendix 13

Register

Register

Email

Password

Confirm password

Register

Confirm password

Passwords don't match X

Register

X

Password doesn't match the confirmation.

Already have an account? [Login here](#)

Already have an account? [Login here](#)

Appendix 14

The screenshot shows a web browser window with the title "CAB230_DOC - Google" and the URL "localhost:4173/volcanoes". The page has a dark blue header with the "VolcanoDB" logo and navigation links for "Home", "Volcano List", and "Account". The main content area is titled "Global Catalog of Volcanoes" with the sub-instruction "Select a country to see a list of volcanoes for that region". A search bar contains the text "uni". Below the search bar is a paginated navigation bar with links for "[first]", "[prev]", "page: [1]", "[next]", and "[last]". A horizontal line separates the navigation from a list of countries. The countries listed are "Papua New Guinea", "United Kingdom", and "United States", each followed by a link "[view volcano list →]".

Appendix 15

The screenshot shows a browser window with the title bar "CAB230_DOC - Google" and the address bar "localhost:4173/volcanoes". The main content area displays the "VolcanoDB" logo and navigation links for "Home", "Volcano List", and "Account". Below this is the heading "Global Catalog of Volcanoes" and the sub-instruction "Select a country to see a list of volcanoes for that region". A search bar contains the placeholder "Search for a country" and a sorting button "[dsc]". Navigation controls include page numbers from 1 to 6, and links for "[first]", "[prev]", "[next]", and "[last]". A list of countries is shown, each with a "[view volcano list ↗]" link:

- Algeria
- Antarctica
- Argentina
- Armenia
- Australia
- Bolivia

The right side of the screen shows the browser's developer tools open in the "Elements" tab, displaying the application's storage structure:

- Application
 - Manifest
 - Service workers
 - Storage
- Storage
 - Local storage
 - http://localhost:4173
 - Session storage
 - IndexedDB
 - Cookies
 - Private state to
 - Interest groups
 - Shared storage
 - Cache storage
- Background services
 - Back/forward cache
 - Background fetch
 - Background sync
 - Bounce tracking
 - Notifications
 - Payment handling
 - Periodic background
 - Speculative loading
 - Push messaging
 - Reporting API

Appendix 16

The screenshot shows a web browser window with the title "CAB230_DOC - Google" and the URL "localhost:4173/volcanoes". The main content is a dark-themed web application titled "VolcanoDB" with a volcano icon. The header includes "Home", "Volcano List", and "Account" buttons. Below the header is the title "Global Catalog of Volcanoes" and the sub-instruction "Select a country to see a list of volcanoes for that region". A search bar contains "Search for a country" and a sorting button "[asc] ↑". A navigation bar below the search bar includes links for "first", "prev", "page: [1] 2 3 4 5 6", "next", and "last". The main list displays the following countries with their respective links:

- Yemen [view volcano list ↗]
- Vietnam [view volcano list ↗]
- Vanuatu [view volcano list ↗]
- United States [view volcano list ↗]
- United Kingdom [view volcano list ↗]
- Undersea [view volcano list ↗]
- Uganda [view volcano list ↗]
- Turkey [view volcano list ↗]
- Tonga [view volcano list ↗]

The browser's developer tools sidebar on the right shows various tabs like Application, Storage, and Background.

Appendix 17

The image displays two side-by-side screenshots of a web application titled "VolcanoDB". Both screenshots show the "Global Catalog of Volcanoes" page, which lists countries and their corresponding volcano lists.

Screenshot 1 (Left): This screenshot shows the initial state of the catalog. The page title is "Global Catalog of Volcanoes" and the sub-instruction is "Select a country to see a list of volcanoes for that region". A search bar at the top right contains the placeholder "Search for a country" and includes keyboard navigation keys "[esc]" and "[down arrow]". Below the search bar is a page navigation bar with links "[first]", "[prev]", "page: [1] 2 3 4 5 6", "[next]", and "[last]". The main content area lists countries with their respective volcano lists:

- Algeria [view volcano list ↗]
- Antarctica [view volcano list ↗]
- Argentina [view volcano list ↗]
- Armenia [view volcano list ↗]
- Australia [view volcano list ↗]
- Bolivia [view volcano list ↗]
- Burma (Myanmar) [view volcano list ↗]
- Cameroon [view volcano list ↗]
- Canada [view volcano list ↗]
- Cape Verde [view volcano list ↗]
- Chad [view volcano list ↗]
- Chile [view volcano list ↗]
- China [view volcano list ↗]

Screenshot 2 (Right): This screenshot shows the catalog after a search for "Colombia". The search bar now contains "Colombia" and includes the same navigation keys. The page navigation bar shows "page: 1 [2] 3 4 5 6" with "[2]" highlighted in orange. The main content area lists countries starting with "Colombia" and their respective volcano lists:

- Colombia [view volcano list ↗]
- Comoros [view volcano list ↗]
- Costa Rica [view volcano list ↗]
- DR Congo [view volcano list ↗]
- Djibouti [view volcano list ↗]
- Dominica [view volcano list ↗]
- Ecuador [view volcano list ↗]
- El Salvador [view volcano list ↗]
- Equatorial Guinea [view volcano list ↗]
- Eritrea [view volcano list ↗]
- Ethiopia [view volcano list ↗]
- Fiji [view volcano list ↗]
- France [view volcano list ↗]

Appendix 18

The image displays two side-by-side screenshots of a web application titled "VolcanoDB". Both screenshots show the same "Global Catalog of Volcanoes" page.

Left Screenshot: The browser tab is "localhost:4173/volcanoes". The page title is "Global Catalog of Volcanoes". Below it says "Select a country to see a list of volcanoes for that region". A search bar contains "Search for a country" with a dropdown arrow. Below the search bar is a page navigation bar with links: "[first]", "< [prev]", "page: 1 2 3 4 5 [6]", "> [next]", and "[last]". A horizontal line separates this from a list of countries. Each country name is in a dark blue box with a white border, followed by a "[view volcano list ↗]" link. The countries listed are: Taiwan, Tanzania, Tonga, Turkey, Uganda, Undersea, United Kingdom, United States, Vanuatu, Vietnam, and Yemen.

Right Screenshot: The browser tab is "localhost:4173/volcanoes". The page title is "Global Catalog of Volcanoes". Below it says "Select a country to see a list of volcanoes for that region". A search bar contains "Search for a country" with a dropdown arrow. Below the search bar is a page navigation bar with links: "[first]", "< [prev]", "page: 1 2 3 4 5 [6]", "> [next]", and "[last]". A horizontal line separates this from a list of countries. Each country name is in a dark blue box with a white border, followed by a "[view volcano list ↗]" link. The countries listed are: Taiwan, Tanzania, Tonga, Turkey, Uganda, Undersea, United Kingdom, United States, Vanuatu, Vietnam, and Yemen.

Appendix 19

The screenshot shows a web browser window for the VolcanoDB application. The address bar displays "localhost:5173/volcanoes/Algeria". The page title is "VolcanoDB" with a volcano icon. The main heading is "Global Catalog of Volcanoes" with the sub-instruction "Select a country to see a list of volcanoes for that region". A button labeled "[← go back to countries]" is visible. Below it, a section titled "Volcanoes near" lists "Algeria". A table provides detailed information for three volcanic fields:

name	volcano region	sub-region	volcanic data
Atakor Volcanic Field	Africa and Red Sea	Africa (northern)	view data →
Manzaz Volcanic Field	Africa and Red Sea	Africa (northern)	view data →
Tahalra Volcanic Field	Africa and Red Sea	Africa (northern)	view data →

Appendix 20

CAB230_DOC - Google | x VolcanoDB x +

localhost:4173/volcanoes/Uganda/52

VolcanoDB

Home Volcano List Account ▾

Bufumbira (Uganda)

[← go back]

Bufumbira

Region:	Africa and Red Sea
Sub-region:	Africa (central)
Last erupted:	Unknown
Summit:	2100 m
Elevation:	6890 m
Coordinates:	-1.3040 lat, 29.6830 long

Population data within 5km

Sign in to access population data

A map of East Africa highlighting Uganda. A blue marker indicates the location of Bufumbira volcano near the northern border of Uganda. The map also shows the borders of neighboring countries: South Sudan to the north, Kenya to the east, Tanzania to the southeast, and Rwanda to the southwest. Major cities like Juba, Nairobi, Arusha, Mombasa, Dar es Salaam, and Kampala are labeled. The Nile River is visible flowing through the region. The map includes labels for various provinces and districts within Uganda, such as Western, Eastern, and Northern Provinces.

Appendix 21

22:32

CAB230_DOC - Google | x VolcanoDB x +
localhost:4173/volcanoes/Uganda/52

VolcanoDB

Home Volcano List Account ▾

Bufumbira (Uganda)

[← go back]

Bufumbira

Region: Africa and Red Sea
Sub-region: Africa (central)
Last erupted: Unknown
Summit: 2100 m
Elevation: 6890 m
Coordinates: -1.3040 lat, 29.6830 long

Population data within 5km

436,134 people live within 5km of Bufumbira.

Population near this volcano

Distance Category	Population Count
< 5km	~100,000
< 10km	~100,000
< 30km	~100,000
< 100km	~1,000,000