

2021

# CAB230 Assignment 1 Client Side



© Douglas Scortegagna

CAB230

Happiness API – Client Side  
Application

Matthew Weir

n10509020

27/04/2021

## Contents

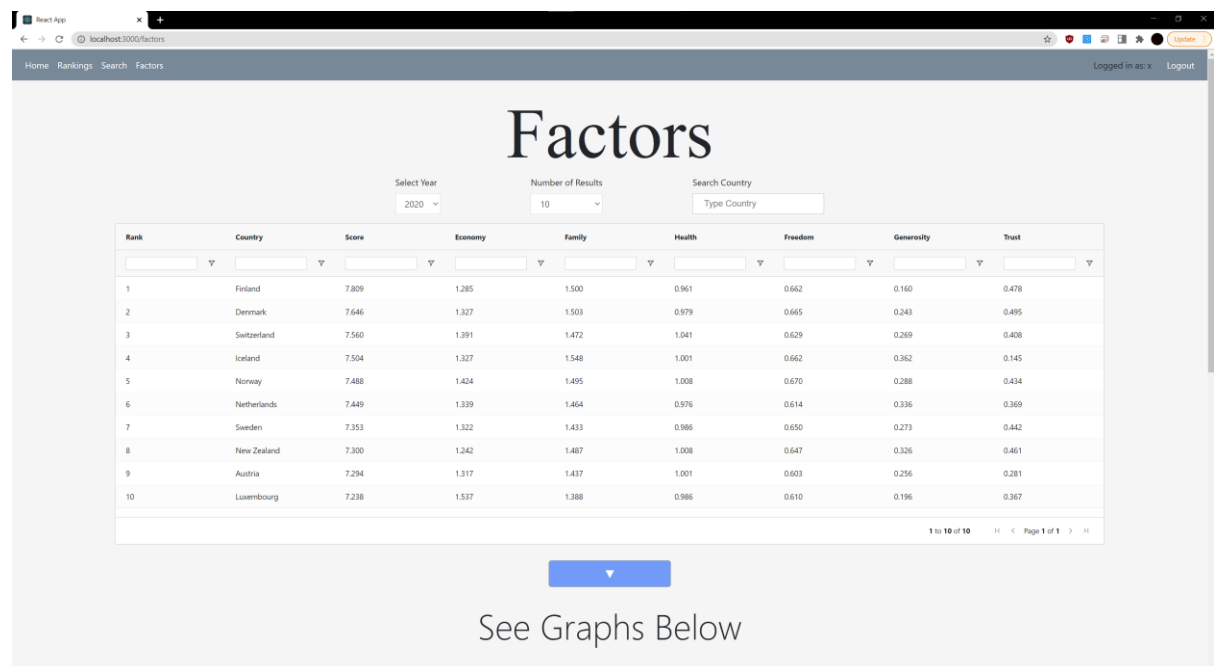
Introduction .....	2
Purpose & description.....	2
Completeness and Limitations.....	3
Use of End Points .....	4
/rankings .....	4
/countries.....	4
/factor/{year} .....	4
/user/register .....	5
/user/login .....	6
Modules Used .....	7
Ag-grid-react .....	7
React-chartjs-2 (HorizontalBar) .....	7
Reactstrap components (Form, Input, Label, FormGroup).....	7
Reactstrap components (Button, Alert).....	7
Reactstrap components (NavBar, Nav, NavItem, NavLink, NavBarText).....	7
Autosuggest .....	7
Styled.....	7
Jwt (jsonwebtoken).....	7
React Router (Router, Switch, Route) .....	7
Application Design .....	8
Navigation and Layout .....	8
Usability and Quality of Design .....	8
Accessibility.....	8
Technical Description .....	10
Architecture .....	10
Test plan.....	10
Difficulties / Exclusions / unresolved & persistent errors.....	12
Extensions (Optional).....	12
User guide .....	12
References .....	12
Appendices as you require them .....	<b>Error! Bookmark not defined.</b>

## Introduction

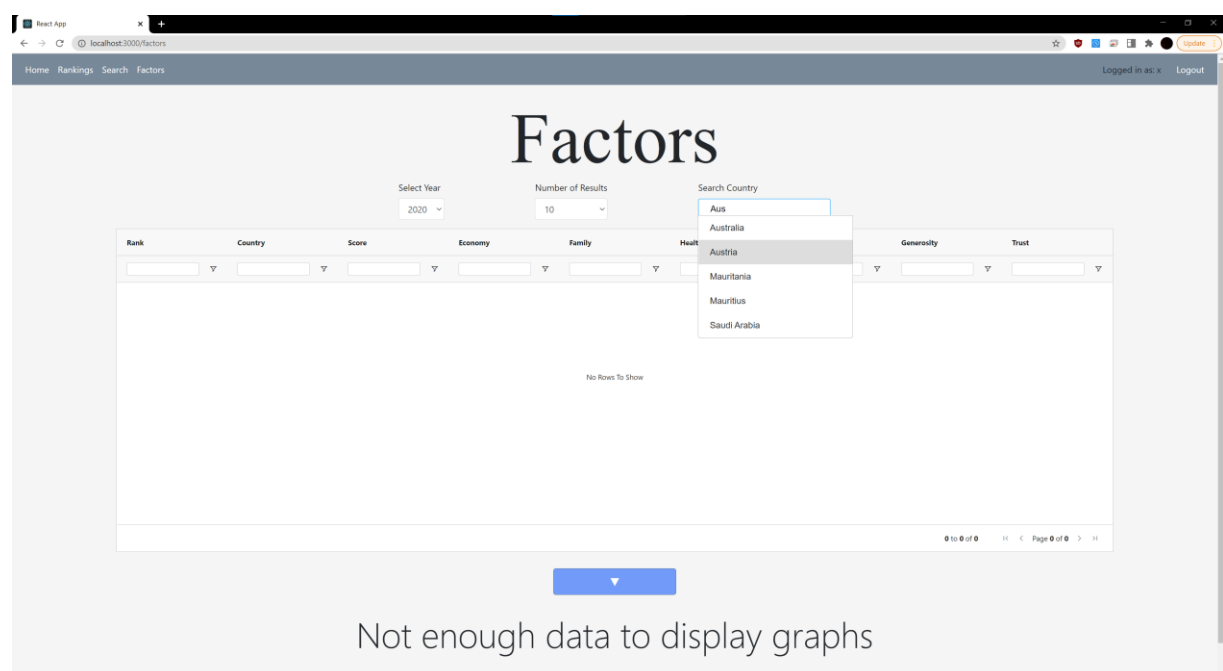
### Purpose & description

This is a React based web application, it allows the users to conveniently access and use the happiness survey data from an external database that is exposed using a REST API.

This web application supplies the information from the REST API in an aesthetic and functional form for the user to manipulate and read.



(Blue scroll down button)



(Autosuggest Input form)

A few things I am proud of when creating this web-app:

- The Scroll buttons for the Factors page,

My issue was I needed to fit all the Bar graphs on the page to help the user make sense of the information coming through in the table. I had it just simply under the table and the user had to scroll down, however it felt quite clunky and non-user friendly.

They are a very simple solution which makes the page more intuitive and user friendly.

- The Autofill/Autosuggest feature on the factors page,

Originally, I had the input as a drop down, however an obvious limitation to that feature for this dataset is it was way too big and displaying it looks ridiculous.

I researched how to do it and came across a component called Autosuggest that allowed me to use the /countries API get call to help the user fill out the input form by suggesting a country based on what they type.

## Completeness and Limitations

My World Happiness Rankings React Application is a sophisticated React Application that includes extensive styling, uses unauthenticated and authenticated data endpoints to display the data in multiple formats.

Formats include:

- AG-Grid table with client-side filtering and search bars.
- AG-Grid table with inputs that alter the data in the table based on an API call.
- Auto updating graphs based on user input & table data.
- Auto Suggest/Autofill Input field.

Within the application user authentication is implemented seamlessly so that certain functions of the application are walled behind authentication, Errors on login and registration are handled cleanly using alert banners that make sure that the web-application never crashes.

The application also handles if the user is unable to connect to the internet/API by showing a clean message in place of the features alerting the user of their connection issue.

The application navigation is set up using a clean navigation bar at the top of the screen that takes advantage of the ReactRouter component to load new web pages.

It also includes Scroll up & down features where there is too much information on the page.

The application is set up in a functional way that most of the features of the web-app are split up into clean & reusable components to avoid doubling up on code.

## Use of End Points

### /rankings



On the rankings page it displays the contents received from the /rankings API call. The user can client-side filter in multiple ways on the data.

### /countries

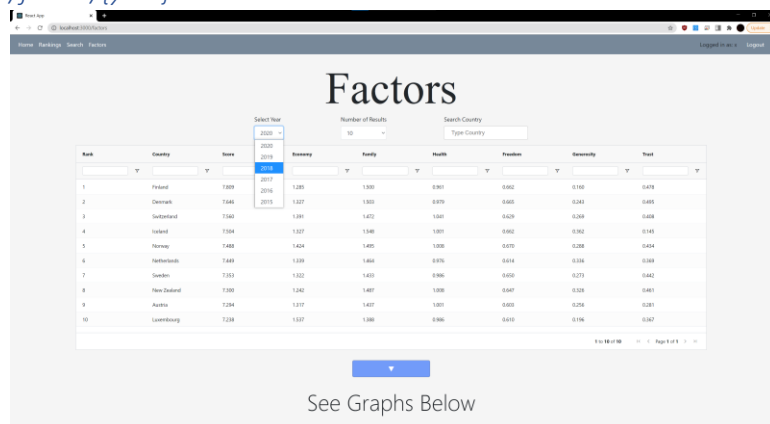


On the factors page, the user can take advantage of the /countries API by an auto suggest input form.

As the user starts to type suggestions come back from the country's API, the user can continue to type or move their mouse over a suggestion and click, and it will fill in the input form for them.

This input is then in turn passed to the factors API call to change the data in the table/graphs.

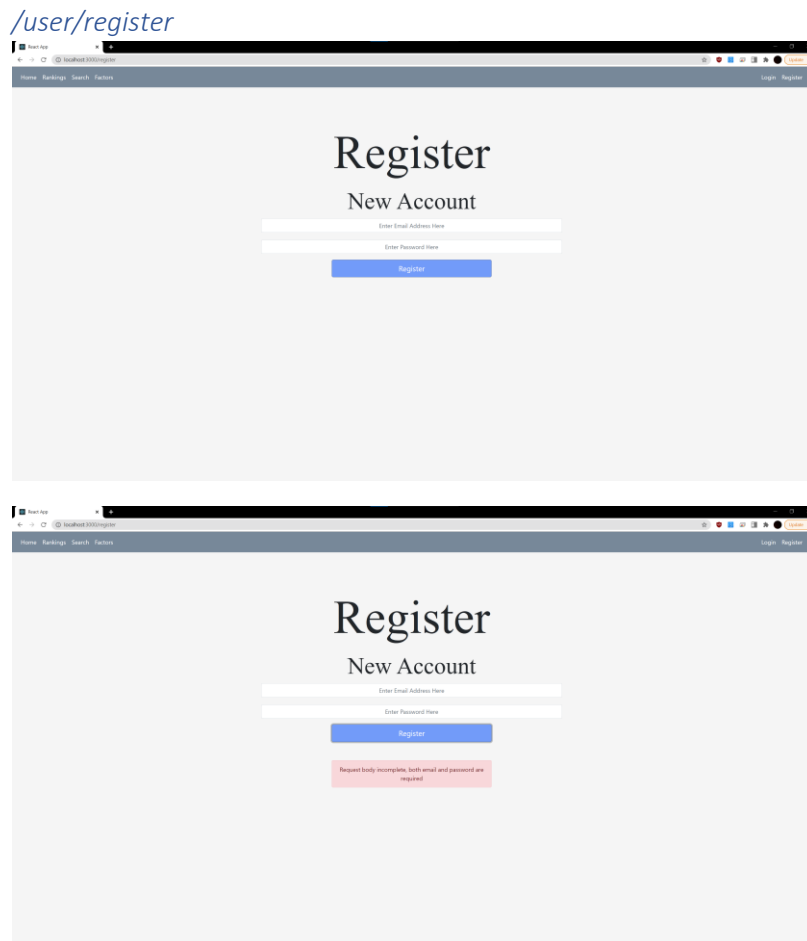
### /factor/{year}



On the factors page, it makes full use of the `/factors{year}` API call. Using the inputs above the table the user can alter the API call in a very functional and convenient way.

The first call is set to the default of Year=2020 & Results=10.

I have limited the initial results to 10 as if there is no need for the data it will only slow the response.



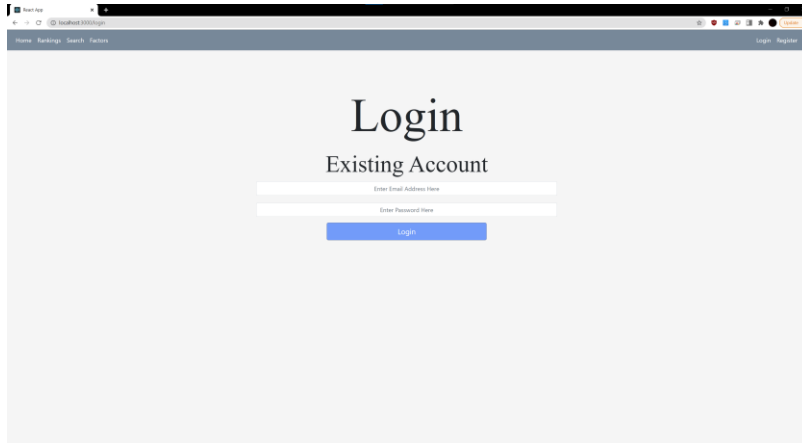
The Register page uses the register post API calls very well.

If the user inputs a unique Username, they can create a new user.

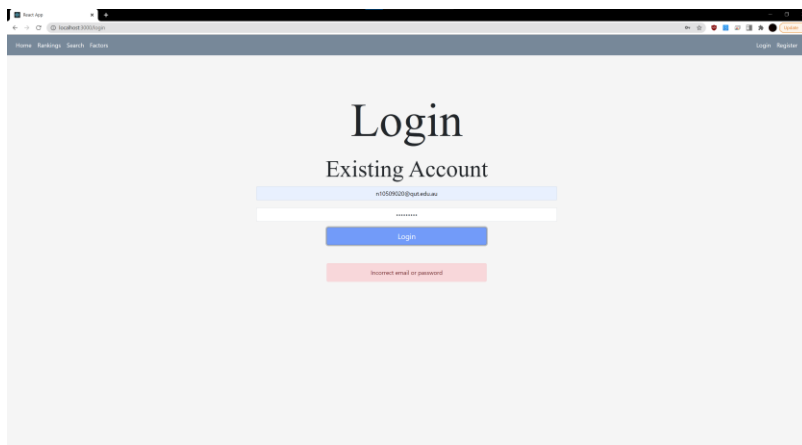
If an error occurs the page stays stable and returns the error message from the API call in an alert banner.

Once the register is successful, the user is redirected to the login page.

/user/login



A screenshot of a web browser showing the login page. The page has a dark blue header with navigation links: Home, Settings, Search, and Factors. On the right side of the header are links for Login and Register. The main content area is light gray and features the title "Login" in a large serif font, followed by "Existing Account" in a smaller serif font. Below this, there are two white input fields: the first is labeled "Enter Email Address Here" and the second is labeled "Enter Password Here". A blue "Login" button is positioned below the password field.



A screenshot of the same login page, but now with an email address "x1050802@qps.edu.au" entered in the first input field. The password field is masked with "xxxxxx". The blue "Login" button is still present. Below the button, a red error message banner displays the text "Incorrect email or password".

The Login page uses the login post API calls very well.

If a user enters the correct combination of Username and password, they can log into their account.

Similarly, to the register form if an error occurs the page stays stable and returns the error message from the API call in an alert banner.

Once the login is successful the user is redirected to the home page, their username is also shown in the navbar until they are logged out.

## Modules Used

### *Ag-grid-react*

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

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

### *React-chartjs-2 (HorizontalBar)*

Module to provide a fully-featured Horizontal Bar chart.

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

### *Reactstrap components (Form, Input, Label, FormGroup)*

These components provided a great backbone in styling and functionality to my input forms.

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

### *Reactstrap components (Button, Alert)*

These components provided a great backbone in styling and functionality to my Scroll buttons & Error Alerts.

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

### *Reactstrap components (NavBar, Nav, NavItem, NavLink, NavBarText)*

These components helped out a lot with the routing functionality of the website. They provided helpful styles and features.

### *Autosuggest*

This component used the data returned from the /countries API call to help the user input their desired country on the Factors page.

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

### *Styled*

I stumbled across the component when researching CSS and decided to give it ago. It lets you put the CSS in the component pages through a variable.

<https://styled-components.com/>

### *Jwt (jsonwebtoken)*

I used this component for secure transmission of information. I used it only in the user authentication process.

<https://jwt.io/>

### *React Router (Router, Switch, Route)*

I used this component to assist me with rendering new pages given components.

<https://reactrouter.com/>



## Application Design

### Navigation and Layout

The design of my website is very basic and allows the user to navigate between the features easily and intuitively. As basic as it is I gained my inspiration from the small example in the task sheet, I liked the idea of having a top navbar to navigate through the features easily, with the limited scope of the website it seems like a very logical way to format the website.

Originally, I had a similar feature to my scroll up/down button to move across the menu navigations, however I found this to be in the way of other features and was not necessary if I had a functioning navigation bar already.

All the important information is up front and in the middle of the display with concise headings to let the user know where they are.

My homepage design is heavily influenced from the MintyYard example in the tutorials, I liked the format of having a large hero image with some text over the top of it to set the mood of the web-app.

### Usability and Quality of Design

Overall, I am very happy as to how I have designed my first React web-application, all be it very simple I think it is well laid out and successfully displays the data from the API to the user in a format that coincides with the 6 ingredients for a successful website as seen in the lectures.

- The display is well organised with the navigation bar/links at the top of the page & the content located in the middle of the screen for the user to easily use and read.
- The navigation is successful in being obvious on how to use, the user does not have to struggle to figure out what the website does and how to get around.
- It is consistent with user expectations from using other applications as the format is a widely adopted by many content creators.
- The application uses a small number of fonts & colors that are consistent across every page of the web-application.

The application meets its needs and is user friendly, I added an extra feature on the Factors page when I thought that usability was suffering, it was not very intuitive due to the user not being able to see that the graphs existed unless scrolling down. Adding the button that scrolls and a heading that invites the user to scroll down only if there is enough data to fill the graphs solved this issue for me in a clean way.

### Accessibility

Comparing my web-application against the edited Priority 1 Accessibility Requirements, I think overall I did well, However I did not have accessibility at the forefront of my mind when designing this application so there are some major fallouts:

- My website does not provide a text equivalent for my only picture on the website, which is a background image, and does not provide an alternative for my scroll buttons, however all my other buttons are text buttons. **5/10**
- All the information I display can be conveyed without color as it is text. **10/10**
- The information provided on the home page and rankings page can be read from the HTML document, However the core feature of this web-application is the data that lies within the table and graphs and they are not able to be read from the html. **4/10**

- I do not have any text equivalents for my data, so they do not update as the site changes. **0/10**
- The screen does not flicker at any stage on the website. **10/10**
- As it is appropriate for my user base, I am using the standard language for my website, English. **10/10**
- In all three of my tables the columns are well identified, and the table makes use of a feature that highlights the row/column the user is hovering over to make it easier to read. **10/10**

Having some practice making a React Application now, If I were to make another one, I would have accessibility in my mind more as I know the foundations of how a React Application works.

I am concerned about my class names and ID names of some of my html elements, I did not have accessibility in mind when creating these names, while I think most of them will be obvious as to what they are relating to I am sure some of them are a little bit vague.

## Technical Description

### Architecture

.git	27/04/2021 10:53 AM	File folder
.vscode	27/04/2021 10:53 AM	File folder
node_modules	22/04/2021 2:59 PM	File folder
public	16/04/2021 2:14 PM	File folder
src	27/04/2021 10:53 AM	File folder
.gitignore	16/03/2021 2:17 PM	Text Document
package.json	27/04/2021 10:53 AM	JSON File
package-lock.json	27/04/2021 10:53 AM	JSON File
README.md	13/04/2021 10:04 AM	MD File
yarn.lock	13/04/2021 10:04 AM	LOCK File

Above is a picture of the overall layout of my react application, all my code to make the application is in the src directory.

images	27/04/2021 2:13 PM	File folder
index.html	14/04/2021 4:34 PM	Chrome HTML Document
logo192.png	16/04/2021 2:14 PM	PNG File
manifest.json	16/03/2021 2:17 PM	JSON File
robots.txt	16/03/2021 2:17 PM	Text Document

Within the public directory I have a folder called images which is where any images I used throughout the web-application are located.

components	27/04/2021 10:53 AM	File folder
pages	27/04/2021 10:53 AM	File folder
api.js	27/04/2021 10:53 AM	JavaScript File
App.js	27/04/2021 11:53 AM	JavaScript File
index.css	16/03/2021 2:17 PM	Cascading Style Sheet Document
index.js	16/04/2021 2:14 PM	JavaScript File
reportWebVitals.js	27/04/2021 10:47 AM	JavaScript File

Within the src directory I have my main App.js file, which is the heart of the program, this contains all the links to the pages.

This file directory also contains a JavaScript file api.js that contains all my API call functions.

BarGraph.jsx	27/04/2021 10:53 AM	JSX File
FactorsTable.jsx	27/04/2021 10:53 AM	JSX File
Footer.jsx	27/04/2021 10:53 AM	JSX File
Heading.jsx	27/04/2021 10:53 AM	JSX File
LoginForm.jsx	27/04/2021 10:55 AM	JSX File
Logout.jsx	27/04/2021 10:53 AM	JSX File
NavBar.jsx	27/04/2021 11:57 AM	JSX File
RankingsTable.jsx	27/04/2021 10:53 AM	JSX File
RegisterForm.jsx	27/04/2021 10:53 AM	JSX File
Scrolls.jsx	27/04/2021 10:53 AM	JSX File

Within the src directory I have split up my components and pages into two separate folders, all the reusable components are within the components directory and I call those components on the pages that they are relevant. This has kept my code clean and well documented.

## Test plan

Task	Expected Outcome	Result	Screenshot Appendix
Successfully load homepage	Homepage is showing	PASS	1
Able to navigate between each page using the navbar	See each unique page in a timely manner using the navbar	PASS	-
Searching a valid year on the search page.	Only results show from the input year (2015)	PASS	2
Searching an invalid year on the search page.	No results show from input year (1999)	PASS	3
Navigate to factors page while logged out	See logged out feature of page.	PASS	4
Navigate to factors page while logged in.	See logged in feature of page.	PASS	5
Incorrectly fill out registration form (Fill only one of the forms out or none).	Banner error shows	PASS	6
Register account that already exists	Banner error shows	PASS	7
Successful Register Account.	Redirects user to login page	PASS	-
Incorrectly fill out login form (Fill only one of the forms out or none).	Banner error shows	PASS	8
Login to non-existing account.	Banner error shows	PASS	9
Successful login.	Redirects user to home page and username displayed in navbar.	PASS	-
Not enough data for graphs.	A heading displaying that there is not enough data should show.	PASS	10
Enough data for graphs	Multiple bar groups should be below, with a heading that says See Graphs Below	PASS	11
Logout.	User redirected to homepage and all features that require user to be logged in are unavailable.	PASS	-

User is not connected to the API.	Tables on all pages should be replaced with text telling the user they have a connection problem after the connection request has timed out	PASS	12
-----------------------------------	---	------	----

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

Whilst creating this application I had a few roadblocks, E.g. (How to format data for the graphs, how to get an autofill/autosuggest feature to work, and in general everything to do with CSS).

As this ecosystem is extremely well documented all it took was time and persistence to find the information I needed to make features work.

An outstanding bug I have here is that the error banners are not consistent in their timing, I know this has to do with how I am using the useEffect hook to generate the timer however I have not been able to fix it.

#### Extensions (Optional)

I enjoyed creating this application, I feel as though if I were to improve on this, I could condense a lot of the features onto one page, as some of the features are trivial.

#### User guide

As an unauthenticated user to begin with you can access the Rankings & Search features with both provide filtering upon the large dataset of countries and their overall scores for every year.

If a user decides they want more features (Factors), they are required to register a unique account and then login using that account.

Within the factors feature the user can change the data in the table using multiple input fields and filtering options, if there is more than one result the graphs down the bottom will populate and the user will see a heading that invites them to click the button to see them down below.

Once the user is finished using the application, they can safely log out of their user account.

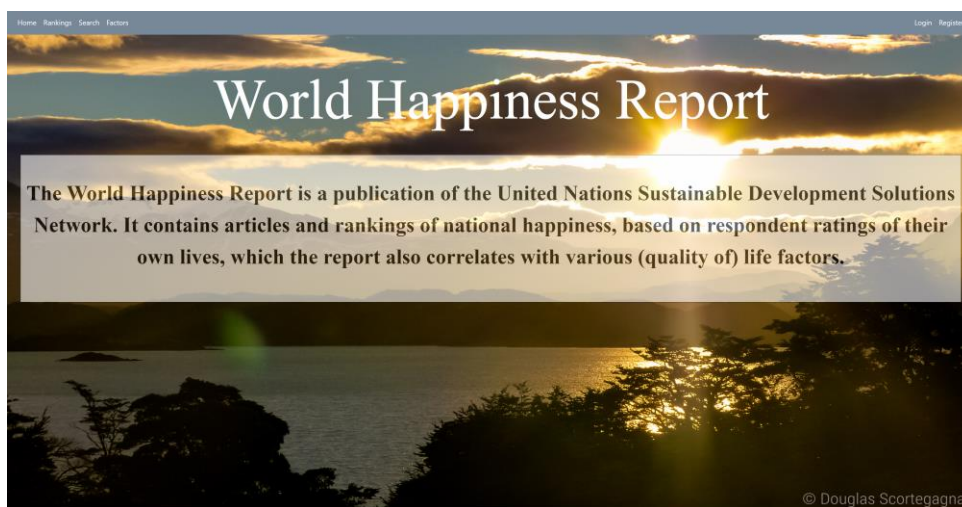
#### References

Image in title, And background image for homepage on website.

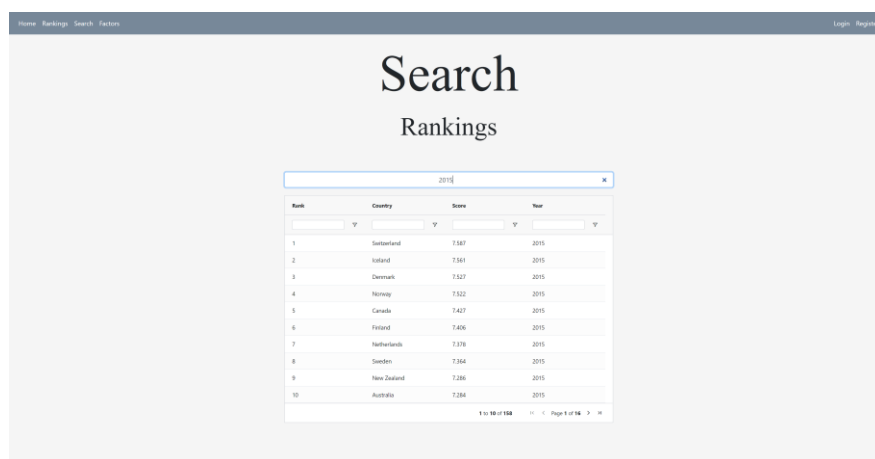
Douglas Scortegagna, D. S. (2014, November 22). *Sunset in Patagonia* [Photograph]. Flickr. <https://www.flickr.com/photos/douglucas/18855702682/in/photolist-uJdpu7-F6PNNm-FuNvdE-d85XtJ-KEULHW-315cN-zGECM-3D8XL6-eaXh2z-2aYJT5u-hxDBVp-s7Myjg-5igdho-PTpDNw-QLsR3Q-8mA2Zn-bUpqGv-qHbiuX-5bEkPC-6MjE8M-82N9pJ-CiAcXk-desXk6-KEULzQ-5J2zDE-cxrJLL-W76uSG-31LAFC-6e2fRC-9jc3DS-32XBfe-9akM8w-5TWQPd-7TeiL3-KAsoYP-ACeG-ApCKEe-4cZQ2q-DvCdd-JAstfu-ojstJu-fw2gGy-81QNsY-vFEv9f-5H8Nxq-QGEcnF-QGEcbD-3dhvG7-pztBQM-QLsR89>

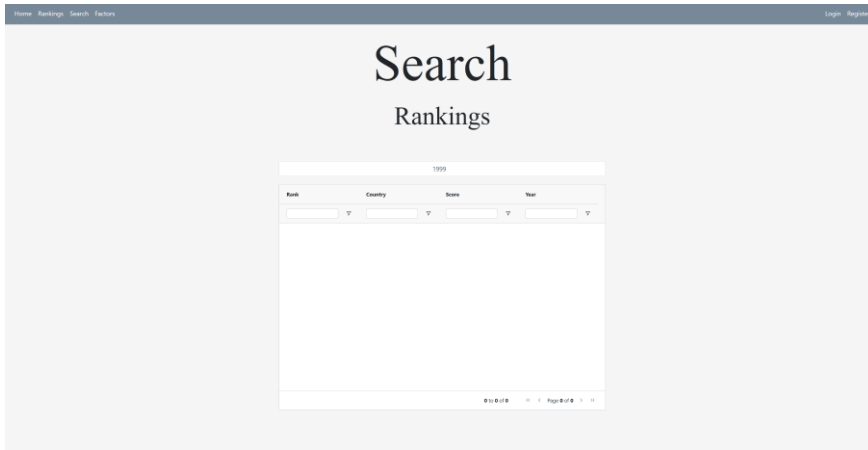
## Appendix

1.

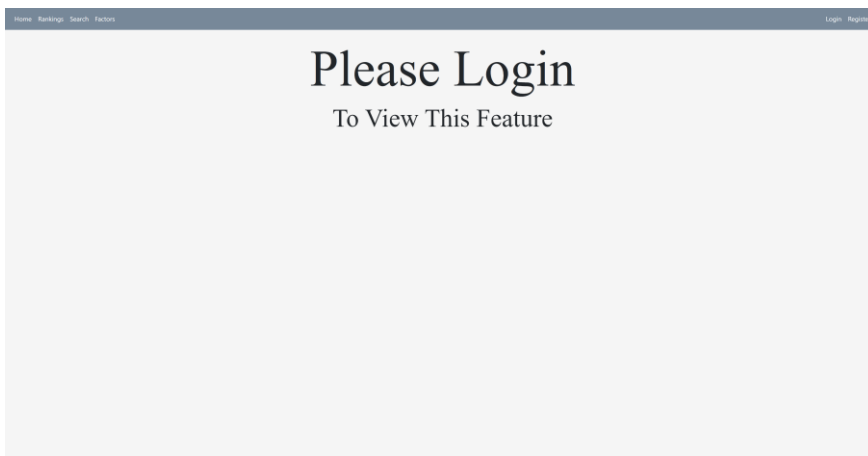


2.





3.

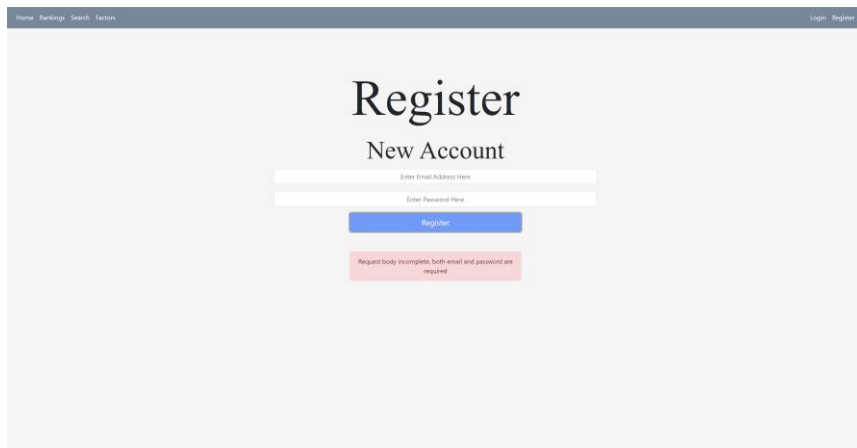


4.



5.

6.



Register  
New Account

Enter Email Address Here

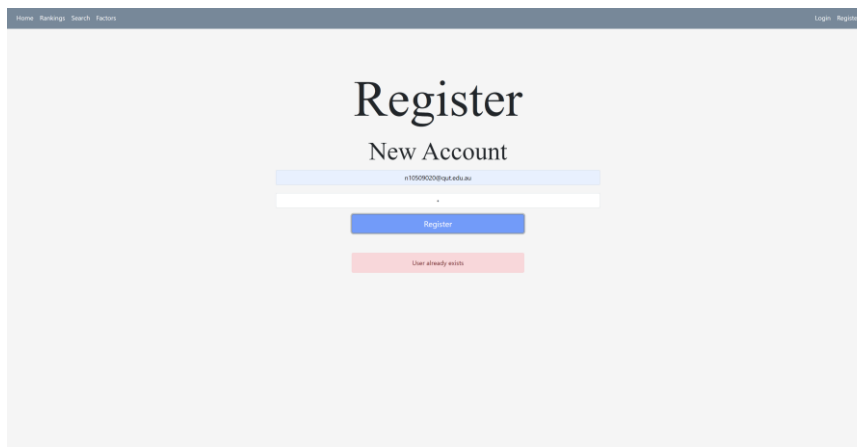
Enter Password Here

Register

Request body incomplete, both email and password are required

This screenshot shows a web form for registering a new account. The form has a title 'Register' and a subtitle 'New Account'. It contains two input fields: 'Enter Email Address Here' and 'Enter Password Here'. Below the password field is a blue 'Register' button. A red error message is displayed below the button, stating 'Request body incomplete, both email and password are required'. The form is set against a light gray background with a dark blue header bar containing links for Home, Rankings, Search, and Factors, and a Login/Register link on the right.

7.



Register  
New Account

e10509020@qat.edu.qa

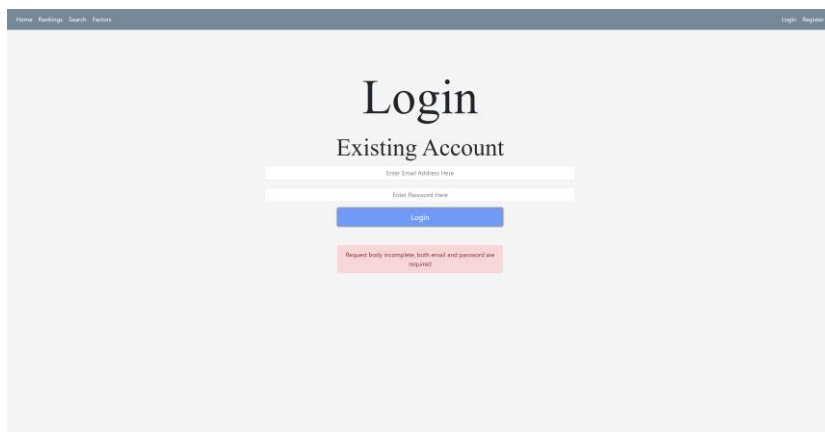
.

Register

User already exists

This screenshot shows the same registration form as in step 6, but with the email field pre-filled with 'e10509020@qat.edu.qa'. The password field now contains a single dot '.'. The 'Register' button is still present, and a red error message below it states 'User already exists'. The layout and header are consistent with the previous step.

8.



Login  
Existing Account

Enter Email Address Here

Enter Password Here

Login

Request body incomplete, both email and password are required

This screenshot shows a web form for logging into an existing account. The form has a title 'Login' and a subtitle 'Existing Account'. It contains two input fields: 'Enter Email Address Here' and 'Enter Password Here'. Below the password field is a blue 'Login' button. A red error message is displayed below the button, stating 'Request body incomplete, both email and password are required'. The form is set against a light gray background with a dark blue header bar containing links for Home, Rankings, Search, and Factors, and a Login/Register link on the right.



Home Rankings Search Factors Login Register

# Login

## Existing Account

9.

Home Rankings Search Factors Logged in as x Logout

# Factors

Select Year: 2020  
 Number of Results: 10  
 Search Country: Australia

Rank	Country	Score	Economy	Family	Health	Freedom	Generosity	Trust
12	Australia	7.223	1.310	1.477	1.003	0.622	0.305	0.396

1 to 1 of 1 Page 1 of 1

Not enough data to display graphs

10.

Home Rankings Search Factors Logged in as x Logout

# Factors

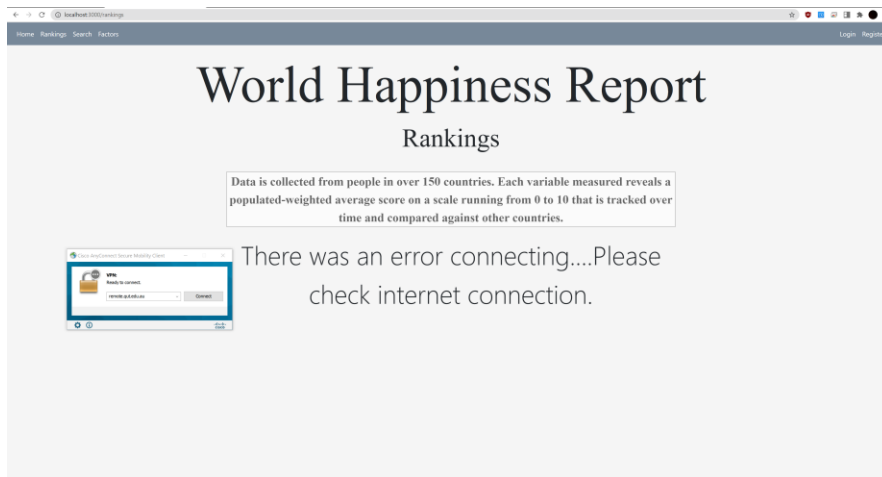
Select Year: 2020  
 Number of Results: 10  
 Search Country: Type Country

Rank	Country	Score	Economy	Family	Health	Freedom	Generosity	Trust
1	Ireland	7.809	1.285	1.500	0.961	0.662	0.160	0.476
2	Denmark	7.646	1.327	1.503	0.979	0.665	0.240	0.495
3	Switzerland	7.560	1.391	1.472	1.041	0.629	0.269	0.408
4	Iceland	7.504	1.327	1.548	1.001	0.662	0.362	0.145
5	Norway	7.488	1.424	1.495	1.008	0.670	0.288	0.404
6	Netherlands	7.449	1.339	1.464	0.976	0.614	0.336	0.369
7	Sweden	7.353	1.322	1.433	0.986	0.650	0.273	0.442
8	New Zealand	7.300	1.242	1.487	1.008	0.647	0.326	0.461
9	Austria	7.294	1.317	1.437	1.001	0.603	0.256	0.281
10	Luxembourg	7.238	1.537	1.388	0.986	0.610	0.196	0.367

1 to 10 of 10 Page 1 of 1

See Graphs Below

11.



12.