

2021

# CAB230 Assignment 1 Client Side

## World Happiness Data 2015-2020

"The present moment is filled with joy and happiness. If you are attentive, you will see it."  
—Thich Nhat Hanh

More Details



CAB230

Happiness API – Client Side  
Application

<Minh Tuan Hoang>

<10537848>

5/10/2021

## Contents

Introduction .....	3
Purpose & description.....	3
Completeness and Limitations.....	3
Use of End Points .....	4
/rankings .....	4
/countries.....	5
/factor/{year} .....	5
/user/register .....	6
/user/login .....	8
Modules Used .....	8
Ag-grid-react .....	9
Bootstrap .....	9
React-strap.....	9
React-icon .....	9
Style-components .....	9
Line Bar .....	9
Application Design .....	10
Navigation and Layout .....	10
Usability and Quality of Design .....	11
Accessibility.....	11
Priority 1 checkpoints.....	11
Technical Description.....	13
Architecture .....	13
Test plan.....	14
Difficulties / Exclusions / unresolved & persistent errors.....	16
Extensions (Optional).....	16
User guide .....	17
References .....	18
Appendices as you require them .....	18

*This template is adapted from one created for a more elaborate application. The original author spends most of his professional life talking to clients and producing architecture and services reports. You may find this a bit more elaborate than you are used to, but it is there to help you get a better mark*

*This report should be around 10 pages or so including screenshots – there is no formal page limit, and the length will depend a lot on the number of screen shots, but you won't get any extra marks for a really long report.*

## Introduction

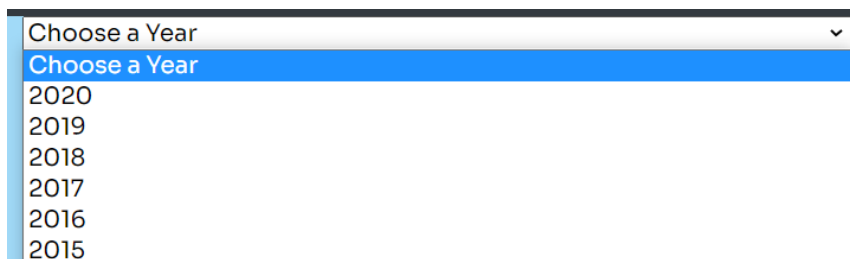
### Purpose & description

The app is designed to present viewers with the data collected from the 'happiness survey', which allows users to select the information and analyze the provided happiness data from the web. This React-based web application was established using the REST API.

As a new student in this web field, I recognized that following the structure and fulfilled it only was a tough challenge. Thus, I stick with the instructions, reorganize that to make them logical following my understand and research.

### Completeness and Limitations

So far, I have successfully implemented the Data route of the Swagger (well-worked with the registration and login endpoints). Sorting data completely with ag-grid React table. I have made some dropdowns fit with the content of the pages (free users and sign-in users) which is to sort the data base on that components: Year, Country or Limitation (In Factors session).



I also make the layout and the design flexible, not too hardcoded so that when we use in different platform or resizing the web page, it goes weird.

About charts, a line chart in Search but does not optimizing, no chart in Factors yet as I was not think about fetching the limitation of the country again and push each of that information into 6 different components like instruction. What I have done just to fetch the data of that factors and put all of that into the line chart, which makes it to display whole 153 countries data and make the page go wrong as I was lose in the running with time:

```
/* <Bar
data = {{
  labels: rowData.map((n)=>(n).country),
  datasets: [{
    label: 'Economy',
    data: rowData.map((n)=>(n).economy),
    backgroundColor: [
      | 'rgba(255, 99, 132, 0.2)',
    ],
    borderColor: [
      | 'rgba(255, 99, 132, 1)',
    ],
    borderWidth: 2
  ]
},
],
}}
```

Furthermore, there is no search section for the "Search" pages, and my dropdown country was not optimizing as all 153 countries fall into that.

## Use of End Points

/rankings

The screenshot shows a web application with a navigation bar containing 'Home', 'Rankings', 'Search', 'Register', and 'Login'. A dropdown menu is open, showing the year '2020'. Below the dropdown is a table with the following data:

Rank	Country	Score
1	Finland	7.809
2	Denmark	7.646
3	Switzerland	7.560
4	Iceland	7.504
5	Norway	7.488
6	Netherlands	7.449
7	Sweden	7.353
8	New Zealand	7.300
9	Austria	7.294
10	Luxembourg	7.238

At the bottom of the table, it says '1 to 10 of 153' and 'Page 1 of 16'.

Default of the rankings, which contains data of the /rankings URL in the Swagger in the Table.

The screenshot shows the same web application, but the dropdown menu is open, showing the year '2018'. The table below the dropdown shows the following data:

Rank	Country	Score
1	Finland	7.632
2	Norway	7.594
3	Denmark	7.555
4	Iceland	7.495
5	Switzerland	7.487
6	Netherlands	7.441
7	Canada	7.328
8	New Zealand	7.324
9	Sweden	7.314
10	Australia	7.272

At the bottom of the table, it says '1 to 10 of 156' and 'Page 1 of 16'.

In this page has the dropdown session for the year, which will use to change the API URL of the data in the exact year that we want to fetch to the table and show it

```
// fetch normal data
fetch(`http://131.181.190.87:3000/rankings?year=${year}`)
  .then(res => res.json())
  .then((data) => setRowData(data));
}, [year]);
```

By default, the year was `const [year, setYear] = useState(["2020"]);`, so the URL will be `/rankings?year=2020`, and if we choose the year of 2018 like in the picture, it will change to `/rankings?year=2018`.

/countries

```
// use to fetch data for the countries dropdown
useEffect(() => {
  fetch(`${API_URL}/countries`)
    .then(res => res.json())
    .then((data) => setCountries(data));
}, []);
```

Like the description in the picture, I use that for fetching all the countries into the dropdown.

Then what country I choose in the dropdown, it will change in the URL:

```
url = `${API_URL}/rankings?country=${selectedCountry}`
```

⇒

General

Request URL: http://131.181.190.87:3000/rankings?country=Argentina

/factor/{year}

```
url = `${API_URL}/factors/${selectedYear}`
```

The screenshot shows the application interface with a table of country rankings for 2020. The table has columns for Rank, Country, Score, Economy, Family, Health, and Freedom. The data is as follows:

Rank	Country	Score	Economy	Family	Health	Freedom
1	Finland	7.809	1.285	1.500	0.961	0.662
2	Denmark	7.646	1.327	1.503	0.979	0.665
3	Switzerland	7.660	1.391	1.472	1.041	0.629
4	Iceland	7.504	1.327	1.548	1.001	0.662
5	Norway	7.488	1.424	1.455	1.008	0.570
6	Netherlands	7.449	1.339	1.464	0.976	0.614
7	Sweden	7.353	1.322	1.433	0.986	0.650

The network log shows the request to the API for the selected country (Argentina).

The default of the /factors contain year and I set the default year to 2020

The screenshot shows the application interface with a table of country rankings for 2020. The table has columns for Rank, Country, Score, Economy, Family, Health, and Freedom. The data is as follows:

Rank	Country	Score	Economy	Family	Health	Freedom
153	Afghanistan	2.567	0.301	0.356	0.266	0.000

The network log shows the request to the API for the selected country (Afghanistan).

/factor with year and country

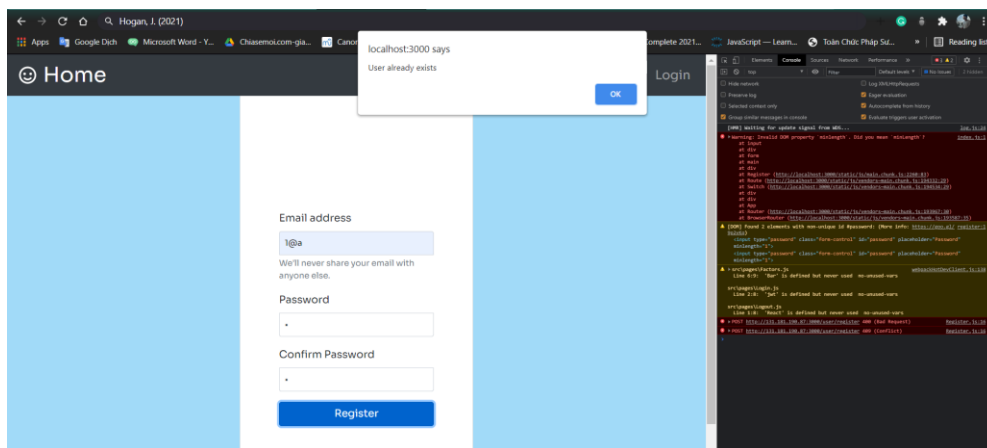
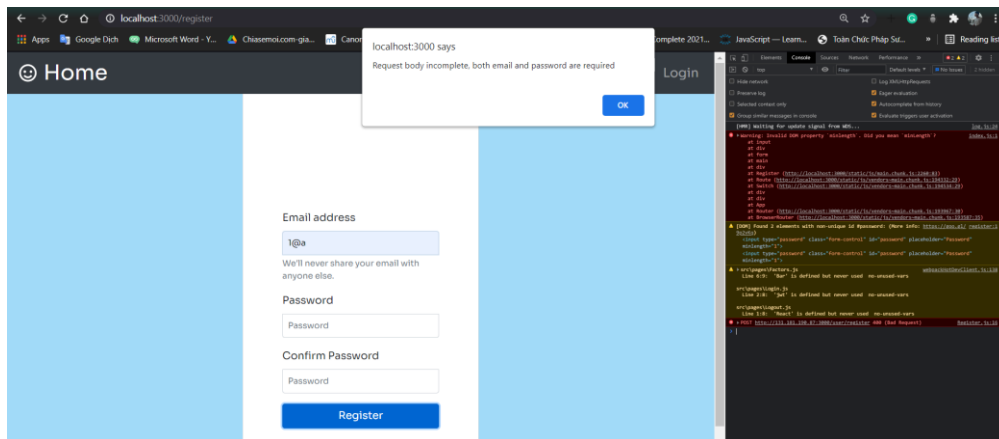
Rank	Country	Score	Economy	Family	Health	Freedom
1	Finland	7.809	1.285	1.500	0.961	0.662
2	Denmark	7.646	1.327	1.503	0.979	0.666
3	Switzerland	7.560	1.391	1.472	1.041	0.629
4	Iceland	7.504	1.327	1.548	1.001	0.662
5	Norway	7.488	1.424	1.495	1.008	0.670
6	Netherlands	7.449	1.339	1.464	0.976	0.614
7	Sweden	7.353	1.322	1.433	0.986	0.650

The screenshot displays a web application interface. On the left, a 'Home' page features a search bar and a table of rankings. The table has columns: Rank, Country, Score, Economy, Family, Health, and Freedom. The first row shows Argentina with a score of 5.975. To the right, a sidebar shows a 'Rankings Search Factors Logout' menu and a 'Request Headers' section with various headers like 'Request-URL', 'Referer', 'User-Agent', etc.

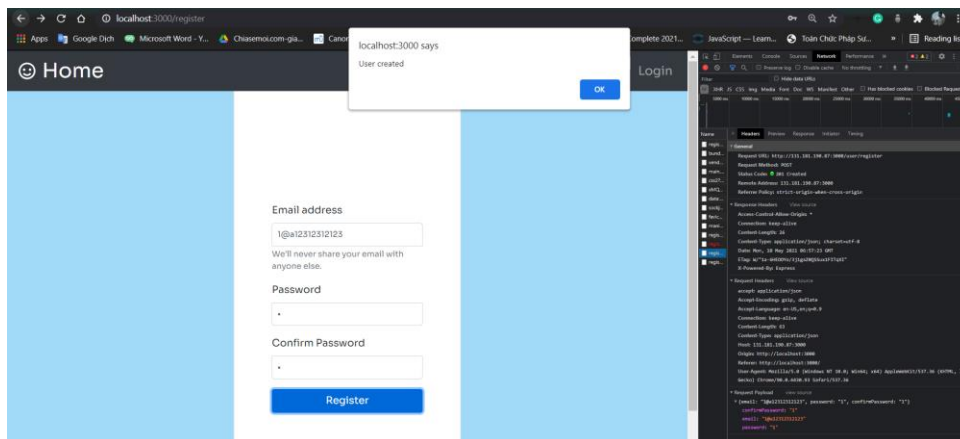
Rank	Country	Score	Economy	Family	Health	Freedom
55	Argentina	5.975	1.028	1.373	0.850	0.521

The sidebar on the right includes a 'Rankings Search Factors Logout' menu and a 'Request Headers' section with various headers like 'Request-URL', 'Referer', 'User-Agent', etc.

```
/user/register
```



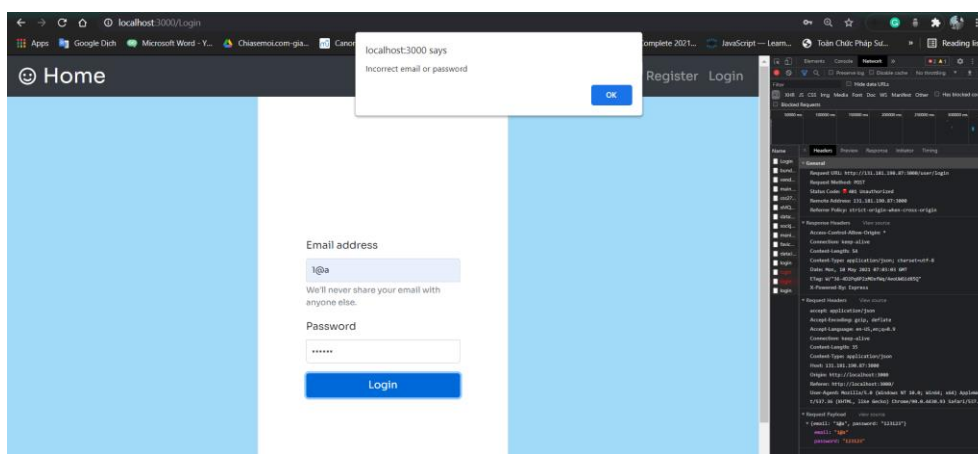
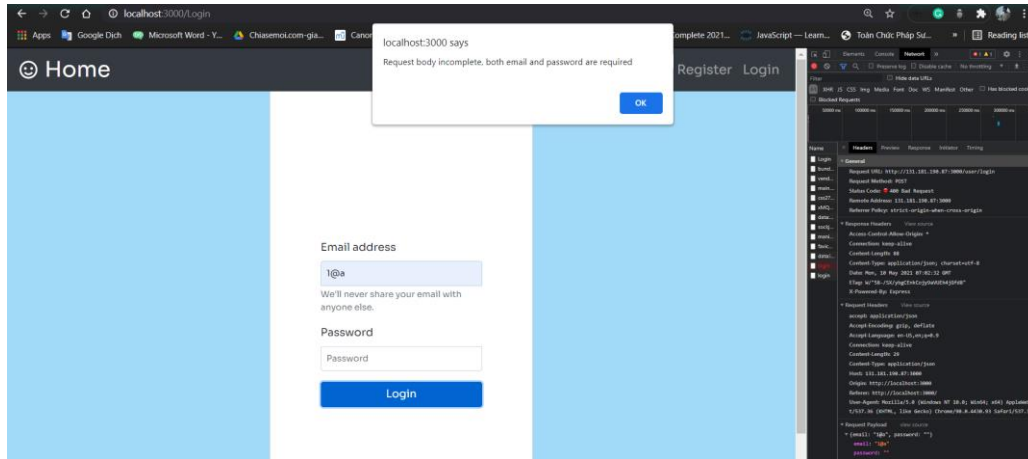
Two errors catch by the Response given by the Swagger.



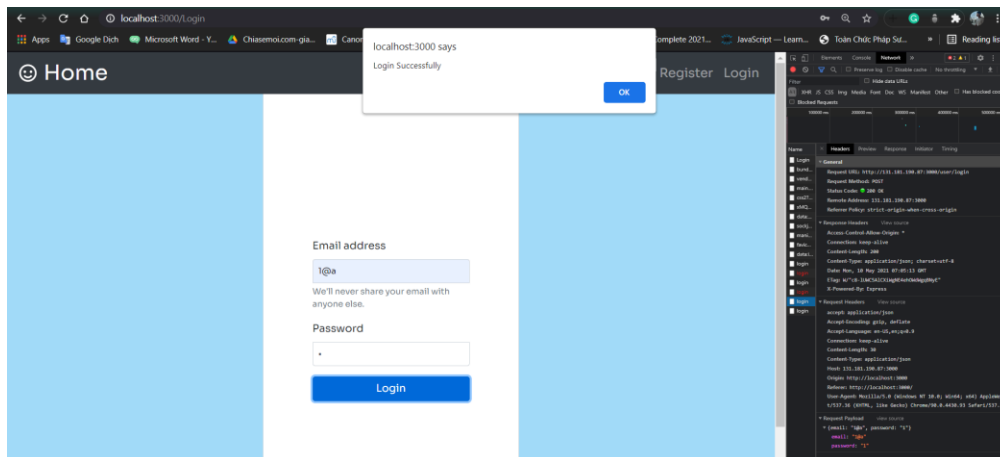
Correct use of register



/user/login



Two errors catch by the Response given by the Swagger.



Correct use of Login.

## Modules Used

This is just a list of the external modules that you have used. You need not specify core React modules. In each case, ***we want the name, a brief description, and a link to the docs at npm or github or wherever.*** The example is `ag-grid-react` as most people will be using this. Just copy that style and add more as necessary.

#### *Ag-grid-react*

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

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

#### *Bootstrap*

Use to support the “Hamburger” dropdown menu.

<https://getbootstrap.com/>

#### *React-strap*

Use for Defining and styling the Navigation.

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

#### *React-icon*

Get the icon for the Navigation to Home.

<https://react-icons.github.io/react-icons/>

#### *Style-components*

Style the homepage

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

#### *Line Bar*

Create a line bar in the Search Section.

<https://www.chartjs.org/>

## Application Design

### Navigation and Layout

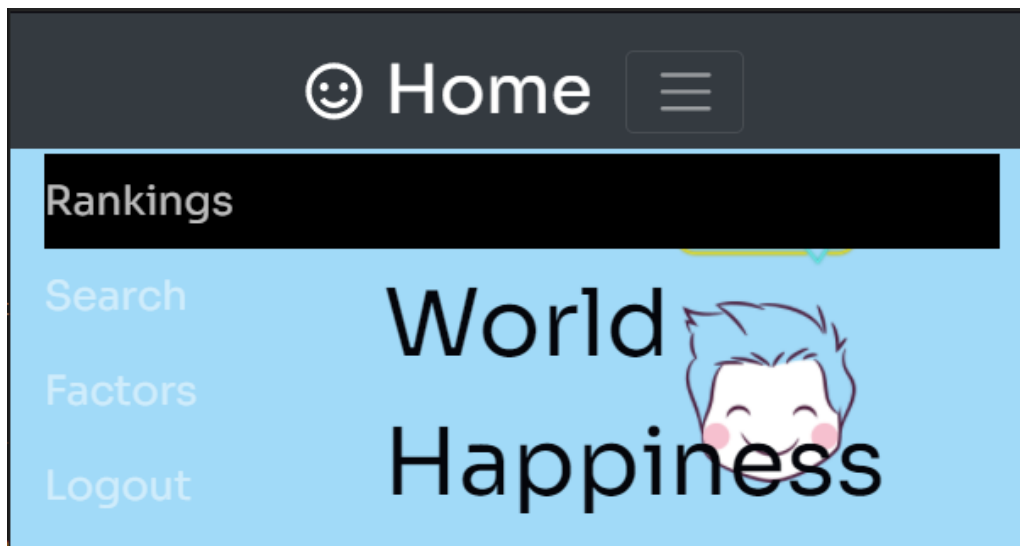
At first, my application has only 1 navbar, which contains all the pages: Home, Rankings, Search, Factors, Register, Login and Logout. Then it causes a problem that when I login, the navbar still appear the Login page, so that the user able to log in once again, which makes duplicate without any meaning. After a while, I think about split the navbar into 2 like the picture below to avoid that error. Moreover, it makes my application look more logical, the user can recognize if they are login or not.



Nav number 1: Represent with free users.



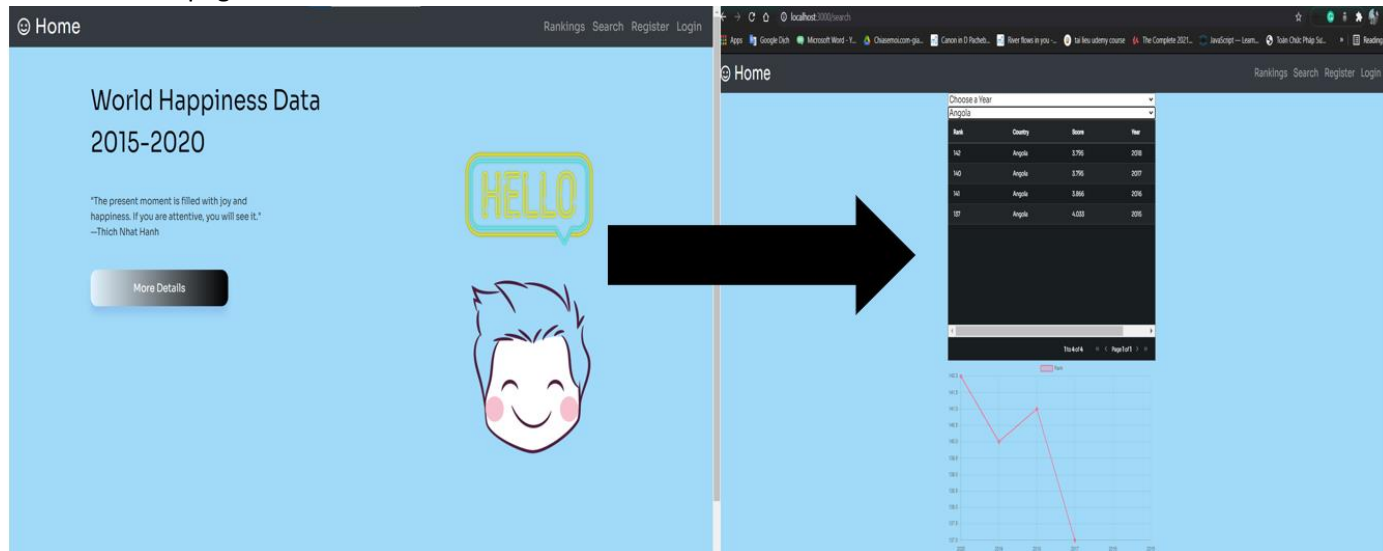
Nav number 1: Log-in users.



Hamburger navigation when resize the dropdown to small.

This navbar is built mainly with bootstrap function, and some CSS to add the hover for example.

Flow from one page to others:



From the homepage, if we click to the Search option in the Navbar, it will redirect to the page /Search like that.

### Usability and Quality of Design

- Is your display well organized? Is the layout clean or clumsy? Kind of
- Is the navigation clear and intuitive? Yes
- Is the application consistent with user expectations from other apps? Nearly
- Is the visual design consistent across screens? Do you use a small number of fonts and font sizes? Are there too many colours? I tried to make as less colour as I can, most of visual design should be consistent.

### Accessibility









## Priority 1 checkpoints

In General (Priority 1)	Yes	No	N/A
<a href="#">1.1</a> Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). <i>This includes:</i> images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video.	x		
<a href="#">2.1</a> Ensure that all information conveyed with color is also available without color, for example from context or markup.	x		
<a href="#">4.1</a> Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).	x		
<a href="#">6.1</a> 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.		x	

<a href="#">6.2</a> Ensure that equivalents for dynamic content are updated when the dynamic content changes.			x
<a href="#">7.1</a> Until user agents allow users to control flickering, avoid causing the screen to flicker.			x
<a href="#">14.1</a> Use the clearest and simplest language appropriate for a site's content.	x		
<b>And if you use images and image maps (Priority 1)</b>	<b>Yes</b>	<b>No</b>	<b>N/A</b>
<a href="#">1.2</a> Provide redundant text links for each active region of a server-side image map.		x	
<a href="#">9.1</a> Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape.	x		
<b>And if you use tables (Priority 1)</b>	<b>Yes</b>	<b>No</b>	<b>N/A</b>
<a href="#">5.1</a> For data tables, identify row and column headers.	x		
<a href="#">5.2</a> For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells.	x		
<b>And if you use frames (Priority 1)</b>	<b>Yes</b>	<b>No</b>	<b>N/A</b>
<a href="#">12.1</a> Title each frame to facilitate frame identification and navigation.	x		
<b>And if you use applets and scripts (Priority 1)</b>	<b>Yes</b>	<b>No</b>	<b>N/A</b>
<a href="#">6.3</a> Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page.		x	
<b>And if you use multimedia (Priority 1)</b>	<b>Yes</b>	<b>No</b>	<b>N/A</b>
<a href="#">1.3</a> Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation.		x	
<a href="#">1.4</a> For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation.		x	
<b>And if all else fails (Priority 1)</b>	<b>Yes</b>	<b>No</b>	<b>N/A</b>
<a href="#">11.4</a> If, after best efforts, you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information (or functionality), and is updated as often as the inaccessible (original) page.		x	

## Technical Description

### Architecture

 .git	5/3/2021 10:48 PM	File folder	
 node_modules	5/10/2021 10:18 AM	File folder	
 public	5/9/2021 9:07 PM	File folder	
 src	5/10/2021 11:07 AM	File folder	
 .gitignore	10/26/1985 6:15 PM	Text Document	1 KB
 package.json	5/10/2021 10:18 AM	JSON File	2 KB
 package-lock.json	5/10/2021 10:18 AM	JSON File	701 KB
 README.md	10/26/1985 6:15 PM	MD File	4 KB

The structure of the coding part is divided into 3 folders, which are components, img, and pages with the CSS file go along with components and pages.







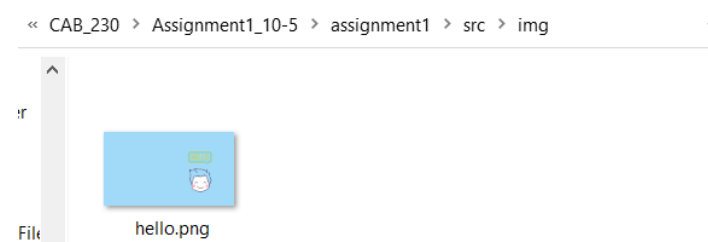




 components	5/10/2021 11:51 AM	File folder	
 img	5/10/2021 11:07 AM	File folder	
 pages	5/5/2021 9:27 PM	File folder	
 App.css	5/9/2021 11:32 PM	Cascading Style Sh...	1 KB
 App.js	5/10/2021 11:31 AM	JS File	2 KB
 index.js	5/7/2021 12:04 PM	JS File	1 KB

Image folder will include all the picture which utilize in the application, right now just contains one for the background.











Components have files that are the basic structure of a website: footer, navbar, and header (in this assignment, I combine navbar and header into the navigation folder). Moreover, with the navigation, I split it into 2 files, navigation with the navbar for the status of the user is unsigned, navigation2

represents when they are log in with the registered account.

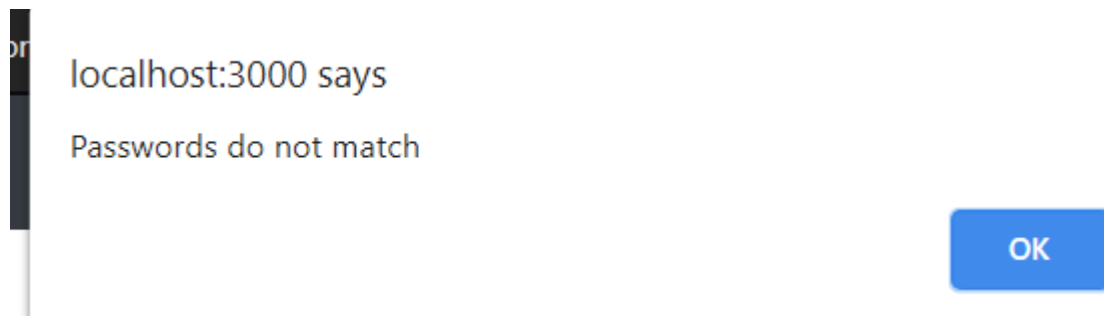
<< CAB_230 > Assignment1_10-5 > assignment1 > src > components <span>▼</span> <span>↺</span> <span>🔍 Search components</span>				
Name	Date modified	Type	Size	
 components.css	5/10/2021 10:56 AM	Cascading Style Sh...	1 KB	
 Footer.js	5/9/2021 9:07 PM	JS File	1 KB	
 Navigation.js	5/10/2021 10:54 AM	JS File	2 KB	
 Navigation2.js	5/10/2021 11:07 AM	JS File	2 KB	

In the pages folder was all the pages that contain in this Website and have a function exactly like their name: Factors, Home, Login, Logout, Rankings and Search.

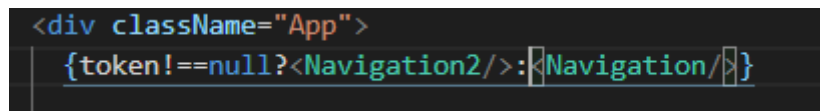
CAB_230 > Assignment1_10-5 > assignment1 > src > pages <span>▼</span> <span>↺</span> <span>🔍 Search pages</span>				
Name	Date modified	Type	Size	
 Factors.js	5/10/2021 5:10 PM	JS File	6 KB	
 Home.js	5/10/2021 12:35 PM	JS File	2 KB	
 Login.js	5/10/2021 5:10 PM	JS File	3 KB	
 Logout.js	5/9/2021 6:56 AM	JS File	1 KB	
 Rankings.js	5/10/2021 4:01 PM	JS File	2 KB	
 Register.js	5/10/2021 4:44 PM	JS File	3 KB	
 Search.js	5/10/2021 12:38 PM	JS File	5 KB	
 styles.css	5/10/2021 12:36 PM	Cascading Style Sh...	1 KB	

### Test plan

Task	Expected Outcome	Result	Screenshot/s
Register new account	User Created – Response 201	Pass	Endpoint session
Handle Register incomplete	Bad request – Response 400	Pass	Endpoint session
Handle Register old account	User already exists – Response 409	Pass	Endpoint session
Handle type in wrong confirm password	Passwords to not match	Pass	1
User log in	Login successfully – Response 200	Pass	Endpoint session
Handle User incomplete	Bad request – Response 400	Pass	Endpoint session
Handle User wrong password	Unauthorized – Response 401	Pass	Endpoint session
Check log-in user	Navigation based in token status	Pass	2
Sort for exact link	Fetching correct URL from Swagger	Pass	3 + 4
Click “More details” button	Move to Wikipedia web about happiness data	Pass	5
Factors Authorization, fix the link with the end /factors	“You have to log in to see this content” alert	Pass	



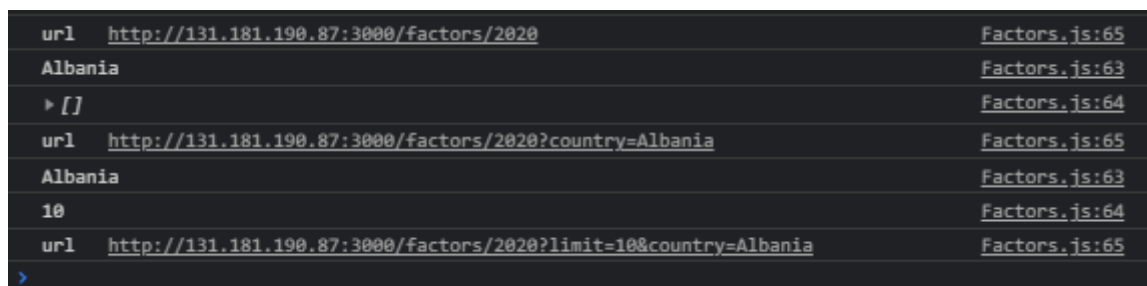
Picture 1



Picture 2

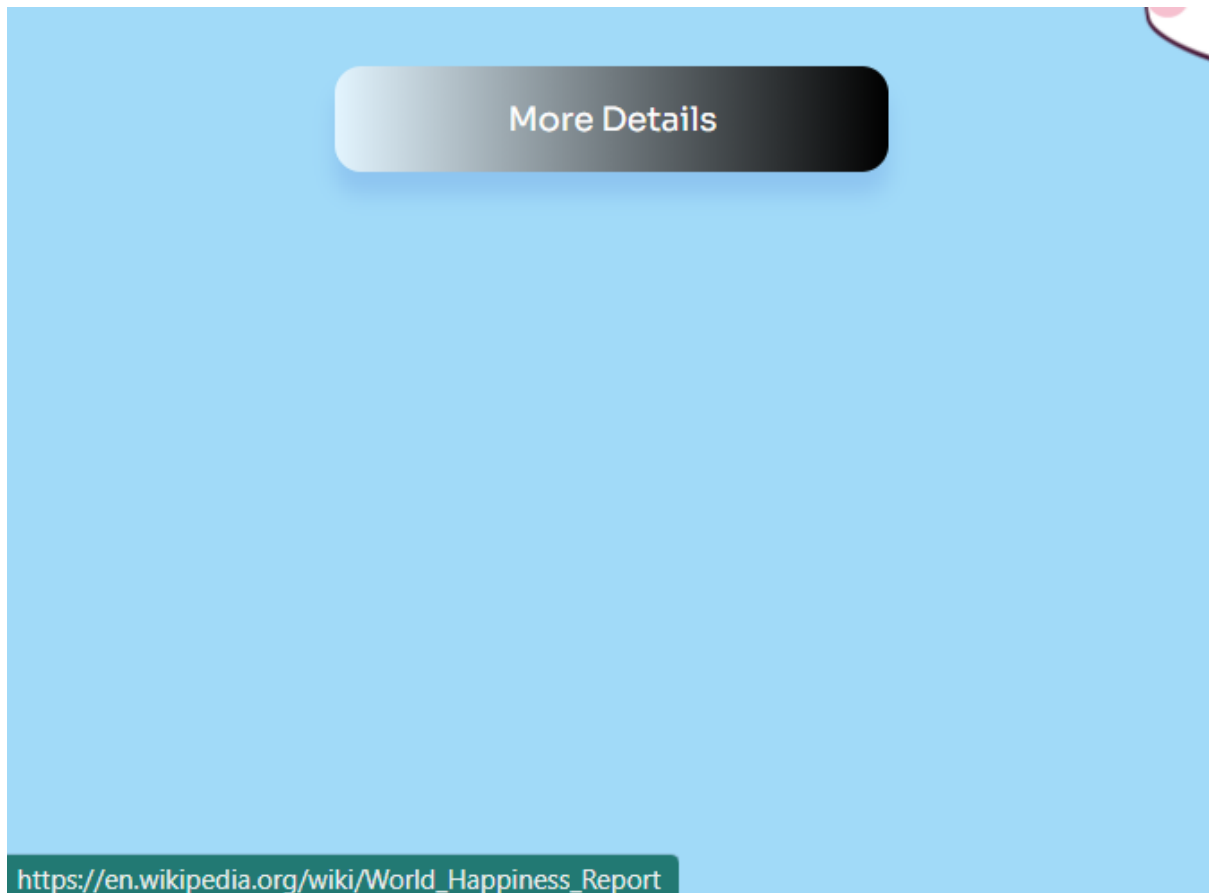


Picture 3



Picture 4





**Picture 5**

Difficulties / Exclusions / unresolved & persistent errors /

- What were your major roadblocks / how did you resolve them? Make a dynamic condition for fetching data from the API Swagger, I solved them by watching youtube instructions, the practical through week and ask my practical tutor about problems.
- Any functionality you didn't or couldn't finish and the technical issues encountered? – Chart for Factors pages. The idea is to have 6 different graphical charts for economy, family, health, freedom, generosity and trust Factors, and we should just visualize some of them (limitation down to 5-10 of each for example).
- Are there any outstanding bugs? Not really.

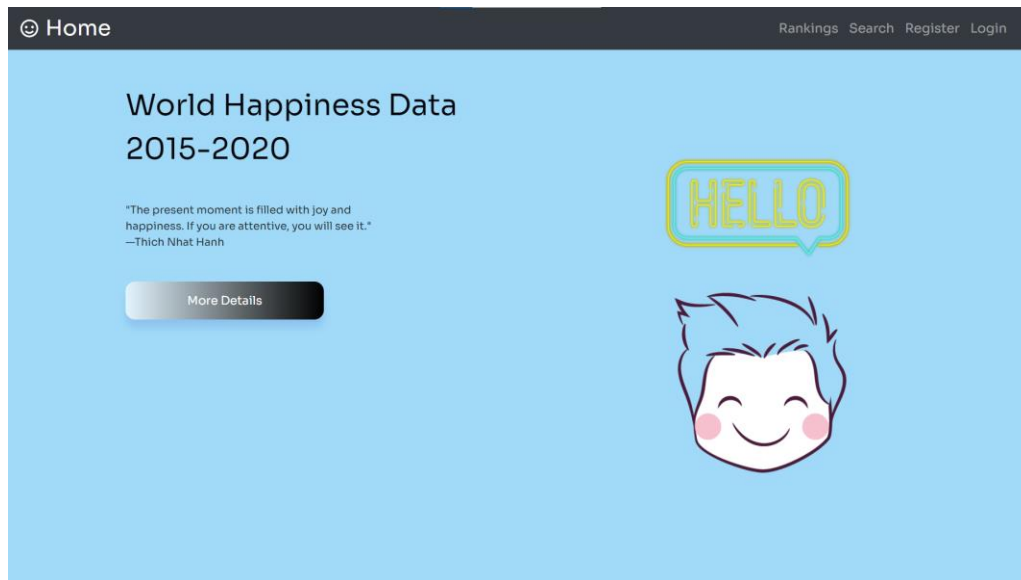
#### Extensions (Optional)

- Make it more optimizing and well-visualized.
- Complete the charts and make it work smoothly.
- Adding the search text in Search pages.
- Sorting countries by region dropdown function.
- Minimize the Register and Login Pages, combine them and add the "Forgot Password" function.

## User guide

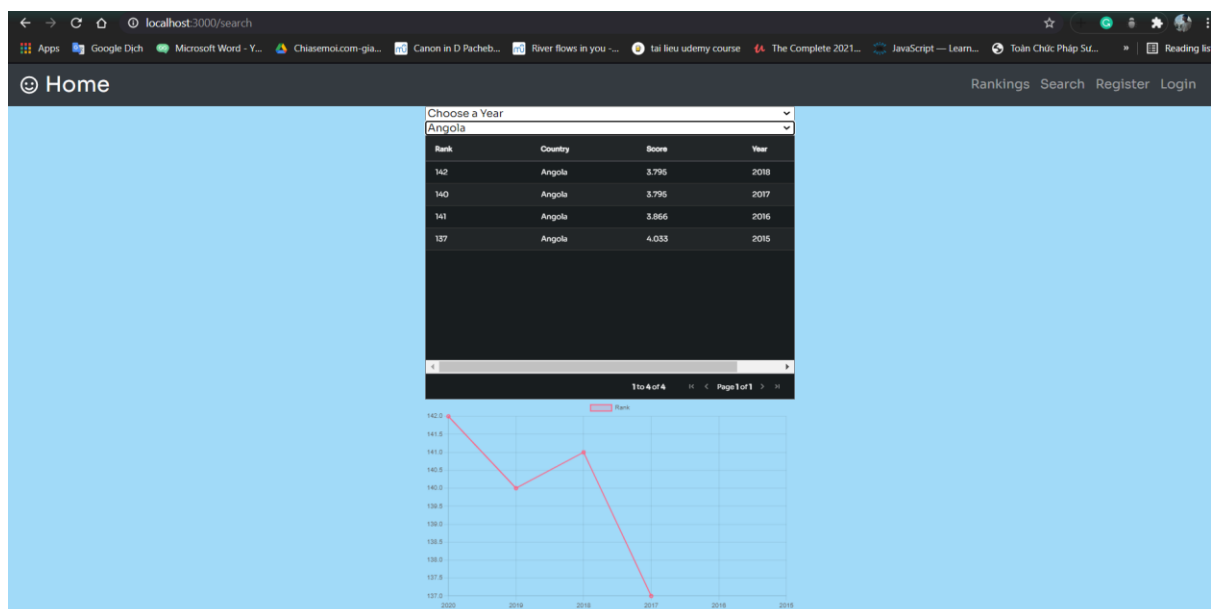
```
PS E:\QUT_Bachelor\CAB_230\Assignment1_10-5\assignment1> npm start
```

Go to the assignment1 folder and call “npm start” to open that, make sure to connect by the QUT network, or use the VPN to redirect to QUT network.



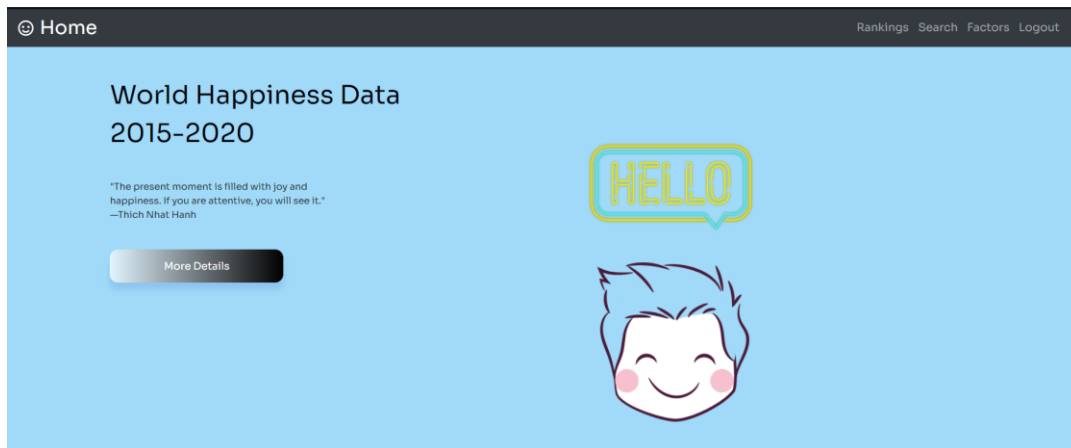
Then it will pop up a homepage like this (scroll down for footer). The more details button will redirect to the happiness data in wikipedia.

The Rankings and Search are pages that showing the free data, we can simply click on that through the navbar on top to explore.



Pop-up page when you click in “Search”.

If you want further information, slightly go in Register account to create a new one. If success, it will redirect to the Login page, otherwise it will catch some errors.



When successfully log in the page, it will redirect back to the homepage, but you can see that the navbar was changed, it changes to the second navbar which use for the sign-in user.

Rank	Country	Score	Economy	Family	Health	Freedom	Generosity	Trust
1	Norway	7.537	1.616	1.534	0.797	0.635	0.362	0.316
2	Denmark	7.522	1.482	1.551	0.793	0.626	0.355	0.401
3	Iceland	7.504	1.481	1.611	0.834	0.627	0.476	0.154
4	Switzerland	7.494	1.565	1.517	0.858	0.620	0.291	0.367
5	Finland	7.469	1.444	1.540	0.809	0.618	0.245	0.383
6	Netherlands	7.377	1.504	1.429	0.811	0.585	0.470	0.283
7	Canada	7.316	1.479	1.681	0.835	0.611	0.636	0.287

This is the Factors page when you click on in the navbar, you can sort data around years, countries and limit in the agrid table.

After finishing, just slightly click the Logout Button, it will redirect back to the homepage like the beginning.

## References

Use a standard approach to referencing – see the guidance at <https://www.citewrite.qut.edu.au/cite/>.

Appendices as you require them