2021

CAB230 Assignment 1 Client Side

CAB230

Happiness API – Client Side Application

Hyeonwoo Lee (Leo)

n10661531

5/10/2021

Contents

Introduction

Purpose & description

The purpose of this app is about showing data to users which is about ‘happiness survey’. This web application supposed to make client could analyze data in various ways. Such as user could get a data by year or country and so on. The ‘happiness survey’ data is from QUT’s provided API.

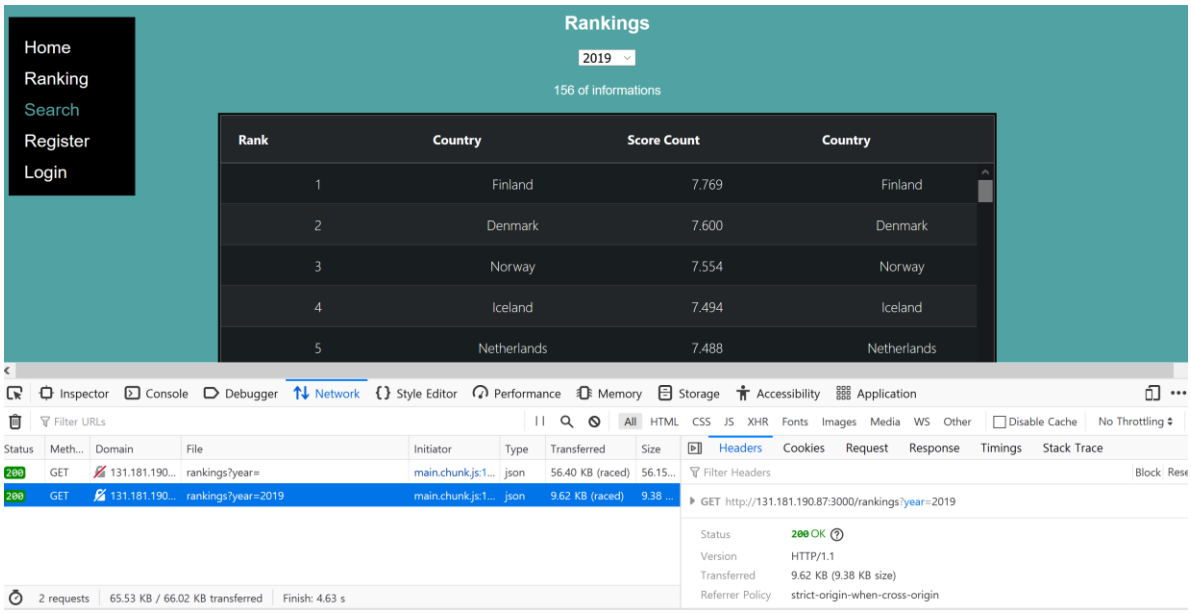
When user goes into this app, they will head to main page, at this page there will be some visible buttons that could be provide to user choices. There is menu that are ranking, search, register and logins. And on the main home page, the feature called "Do you want to see extra Factor lists? than sign in! There is further information!" is mentioned so that users can expect and induce sign-up. Also, beside that comment there are login and register button to let user login or register immediately.

Completeness and Limitations

From my judgment, the basics are in place and work smoothly however, there were some trying make specification in some part. Firstly, the get error part in register, what user must insert while register, it is email and password. In this part if email form is “abc” it supposed to catch it is not valid form because email supposed to be “abc@example.com”. It was one of limited part of this app. The second one is about designed because if the tap size get smaller some characters and buttons are overlapped so it is har to see and hard to process function.

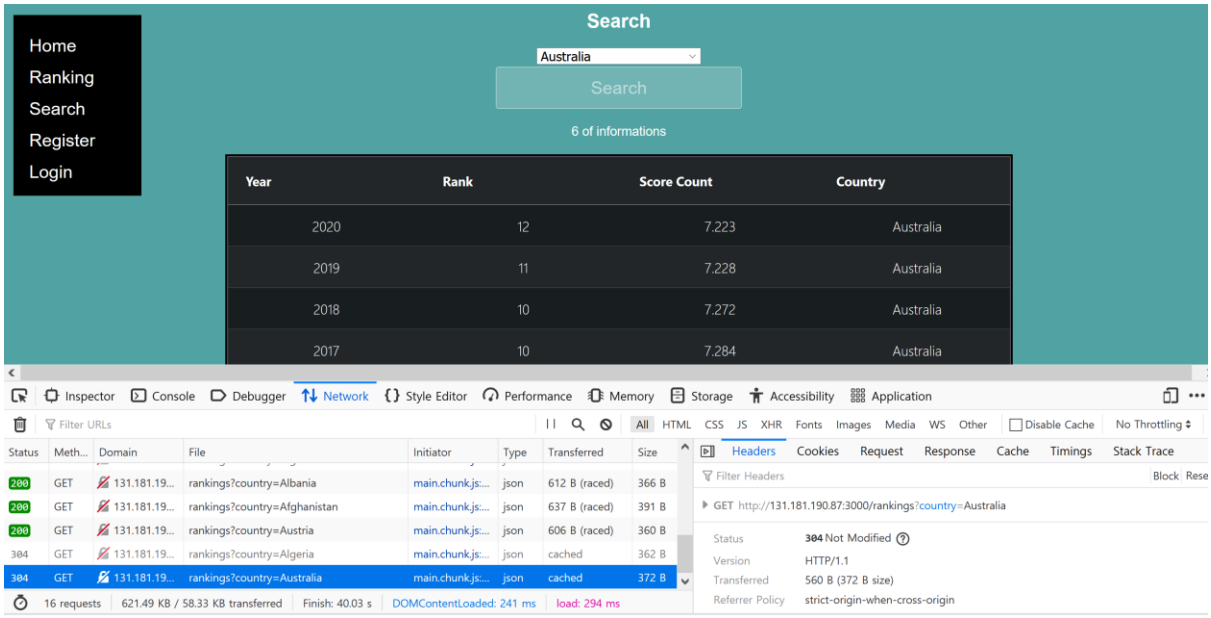
Use of End Points

/rankings

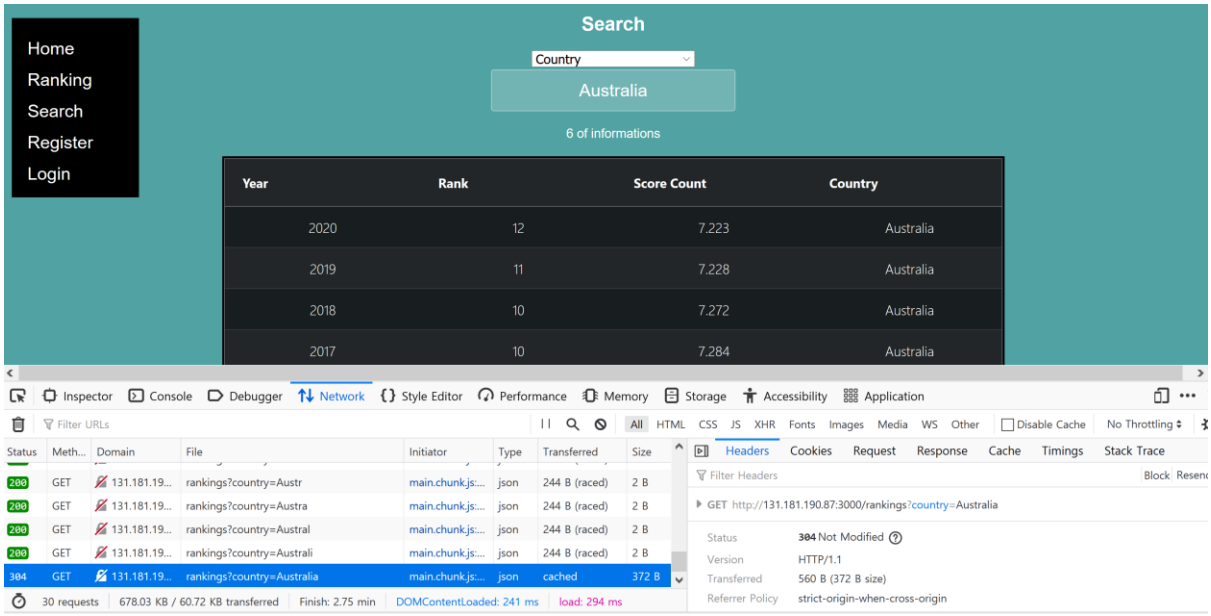


In Ranking page, it shows the GET method, as it shows, the data that requested went correctly into the API link.

/countries

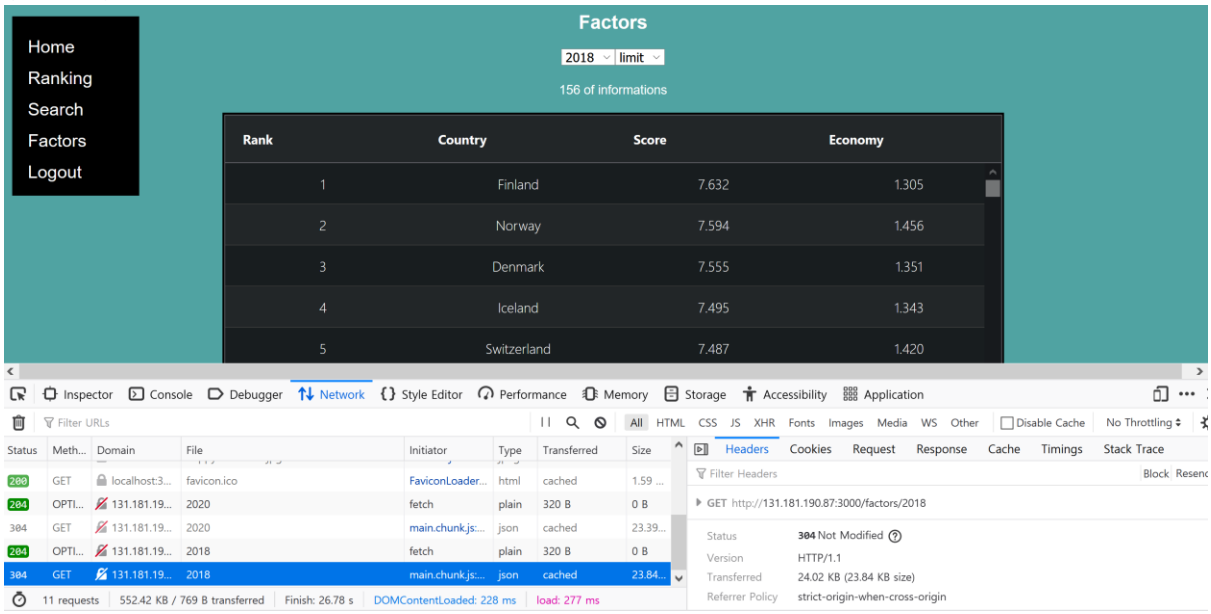


In search page, it shows the GET method, as it shows, the data that requested (by dropdown bar) went correctly into the API link.

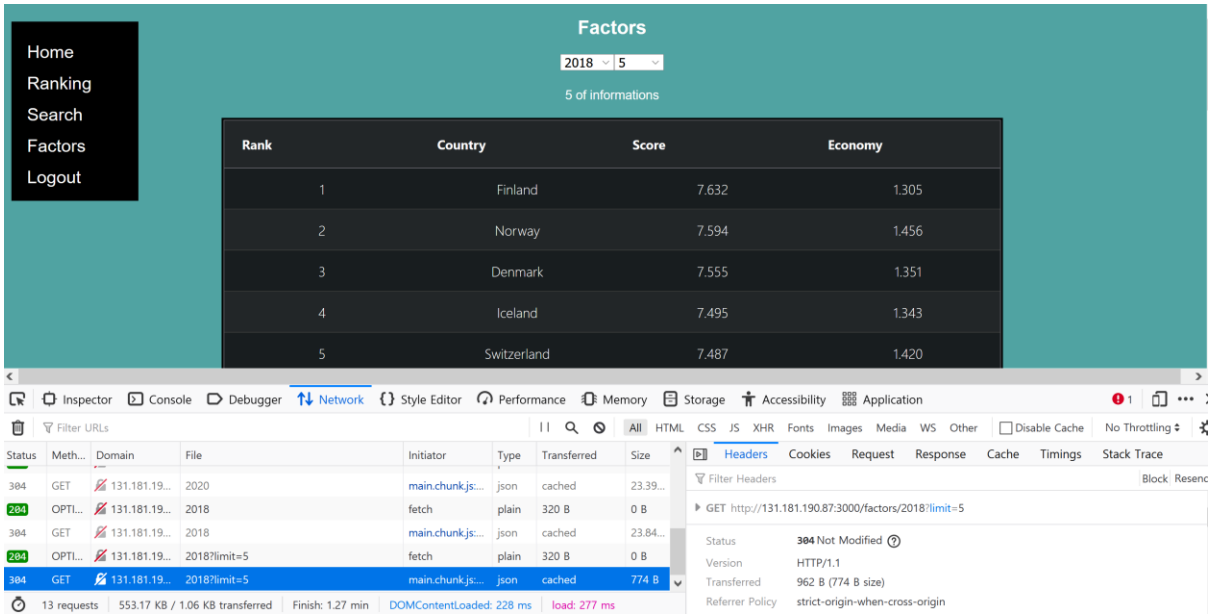


In search page, it shows the GET method, as it shows, the data that requested (by search) went correctly into the API link.

/factor/{year}

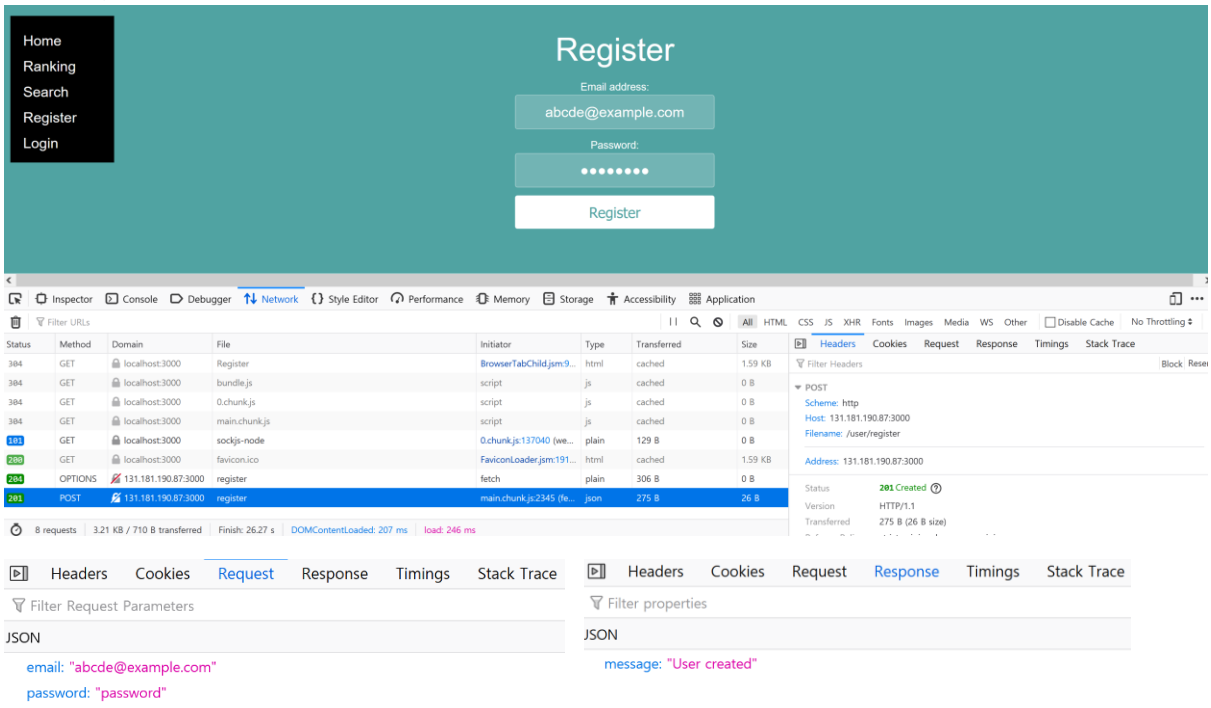


In this factor page, it shows the GET method, as it shows, the data that requested (by year dropdown bar) went correctly into the API link.



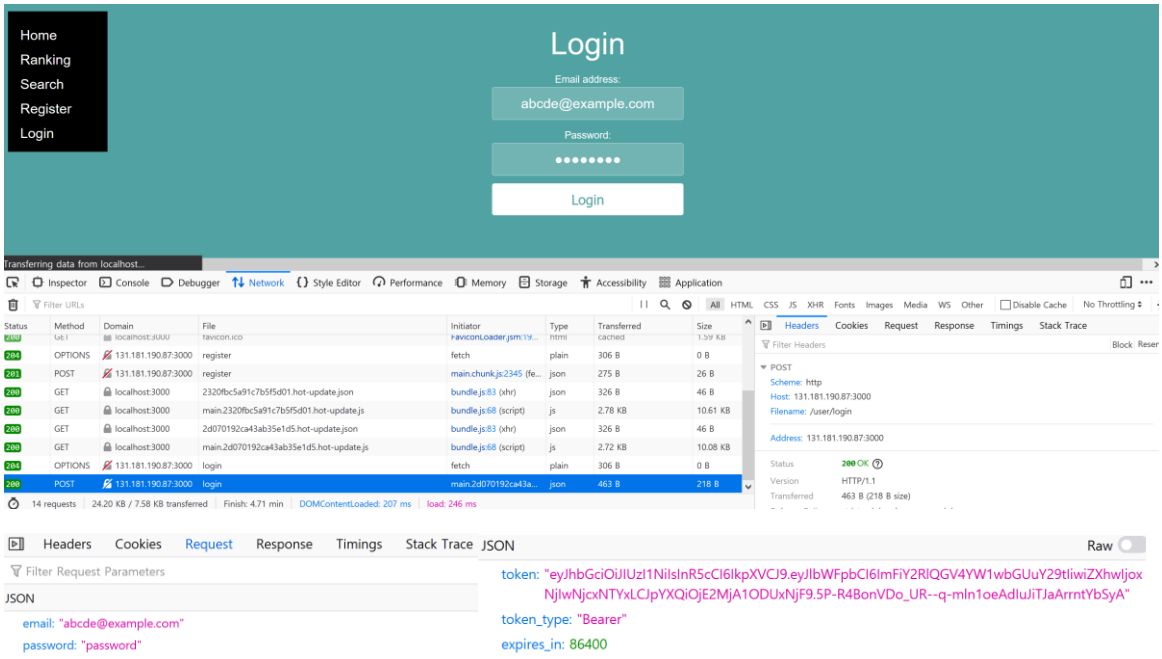
In this factor page, it shows the GET method, as it shows, the data that requested (by year dropdown bar and limit dropdown bar) went correctly into the API link.

/user/register



In this Register page, it shows the POST method, and it gets user data into local storage and send to API than get response of error check. In image above user data went correctly into the API link.

/user/login



In this Login page, it shows the POST method, and it check user from data from local storage by token and get response of error check. The user data went correctly into the API link.

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/

reactstrap

Module quickly build forms. Especially to use on Badge function.

https://reactstrap.github.io/

ag-grid-community/dist/styles/ag-grid

Module from AG Grid Community to build valuable data lines.

https://www.ag-grid.com/javascript-grid/modules/

ag-grid-community/dist/styles/ag-theme-alpine-dark.css

Module from AG Grid Community to change data lines colour.

https://www.ag-grid.com/javascript-grid/themes-provided/

react-chartjs-2

The npm packages to create chart in React js.

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

react-router

Module to use router function in react.

https://reactrouter.com/

react-dom

The package that can be used in react about DOM specific methods.

https://reactjs.org/docs/react-dom.html

Application Design

Navigation and Layout

The process of design could be divided by two big part, the first part is login and logout statement part, second part is the data showing part. Firstly, I made a homepage, there are navigation var and home page contents, in here user could do login through login option, if logged in, login button and register button will be disappear and factors button and logout button will come out which is the only function that allows to log in users so, users could go to factors or logout pages.

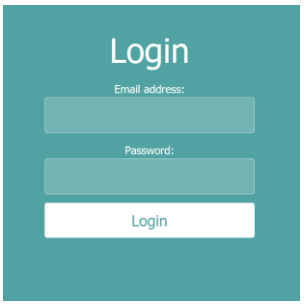
In data showing part, there are some cases which are ranking page, search page and factor page. In ranking page data showed but it could be sorted by years and in search page, data could be shown through searching by specific countries. At last, in factors page, data will be shown by years and limit's dropdown var selection.

The mock designed.

First menu nav var shows the register and login button before logged in. And after logged in, navigation var will become to factors button and logout button.

If user do register, page will go immediately into login page.

======= >>

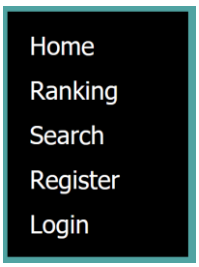


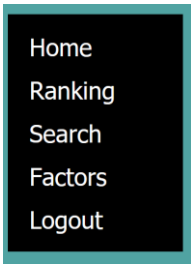
Usability and Quality of Design

- Is your display well organized? Is the layout clean or clumsy?
  My display is well organized because I choose to use nav by a vertical version which is for organize well. However, when screen size gets smaller than expected size, designes are overlapped and crashes. So, it does not look well organized.

<< == if it website's page become smaller, than overlapped like this.

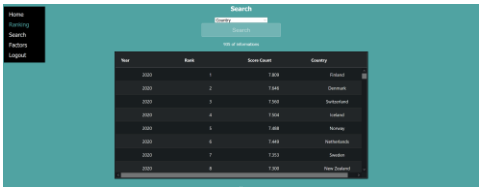
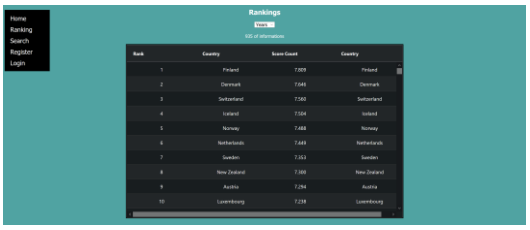
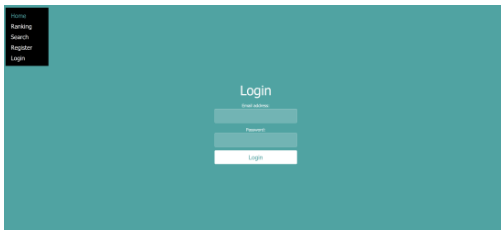
- Is the navigation clear and intuitive?
  Yep, because as picture downside, the main options are all for link to necessary functions. menus are considered by user's logged in statement, so if user does not log in it does not show invalid menu buttons like factors and logout.



and


- Is the application consistent with user expectations from other apps?
  Basic parts are generally similar to other apps; however, the weakness part of this app is the specifications the amount of specification details are lacked in our app. Such like find ID password function, video function sound function and so on.

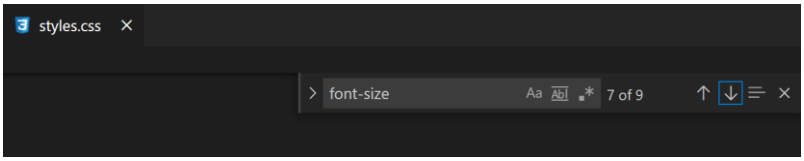
- Is the visual design consistent across screens? Are there too many colors?
  This app's pages are tried to design by same concept which is like same background color same way to feature data.

- Do you use a small number of fonts and font sizes?

  In this app only two fonts were used "Merriweather", serif and "Source Sans Pro", sans-serif.

  However, to fit handle data case by case, several font sizes were used.



- Comment on the usability of your design – are there compromises that make it awkward to use? How might you improve those?
  As this application is for data analysis, people may feel it is difficult. By putting an animated character image on the main screen that looks happy, it makes a connection with happiness, and at the same time, it allows users to feel less difficult, use comfortably, and increase accessibility.

Accessibility

- Provide a text equivalent for every non-text element – alternatives to images, symbols, scripts, graphical buttons, sounds, audio and video files and so on.

  YES

- Ensure that all information conveyed with color is also available without color, for example from context or markup.

  YES

- 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.

  NO

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

  YES

- Avoid causing the screen to flicker.

  YES

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

  YES

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

  YES

From question “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.” It is not available in this app. Because style shit and document must be connected with one another. It will interrupt to do code if I use this on app.

The part that helps to design is especially the question “Ensure that all information conveyed with color is also available without color, for example from context or markup.”. It can be used without color like markup, design process became simple and efficient.

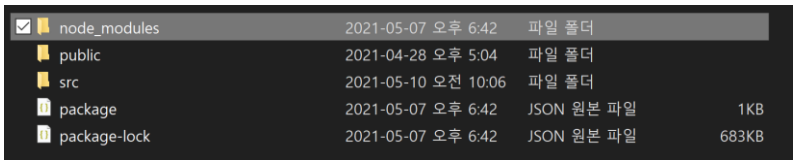
The question “Avoid causing the screen to flicker.” Is one of my problem by miss code, how ever O foxed this by using the comment history. push () function which is from react-router module.

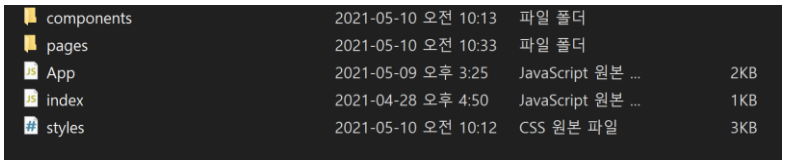
Technical Description

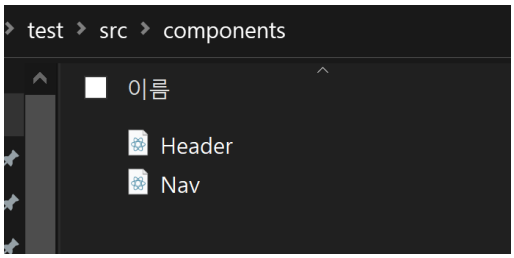
Architecture

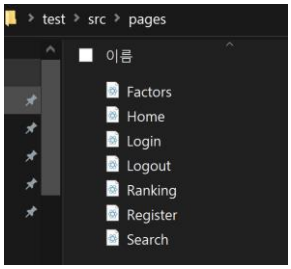
```
import Home from "./pages/Home";
import Ranking from "./pages/Ranking";
import Search from "./pages/Search";
import Factors from "./pages/Factors";
import Register from "./pages/Register";
import Login from "./pages/Login";
import Logout from "./pages/Logout";

export default function App() {
  return (
    <Router>
      <div className="App">
        <Header />
        <Switch>
          <Route exact path="/">
            <Home />
          </Route>
          <Route path="/Ranking">
            <Ranking />
          </Route>
          <Route path="/Search">
            <Search />
          </Route>
          <Route path="/Factors">
            <Factors />
          </Route>
          <Route path="/Register">
            <Register />
          </Route>
          <Route path="/Login">
            <Login />
          </Route>
          <Route path="/Logout">
            <Logout />
          </Route>
        </Switch>

      </div>
    </Router>
  );
}
```

As this left picture shows, this web is constructed like this. There are several imports to show a different page. Which means there are extra js react pages.

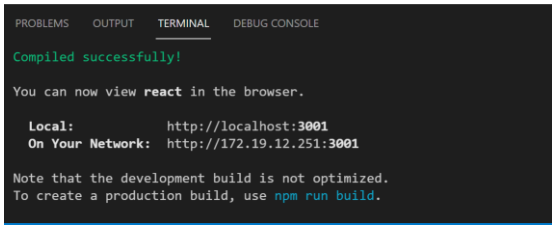


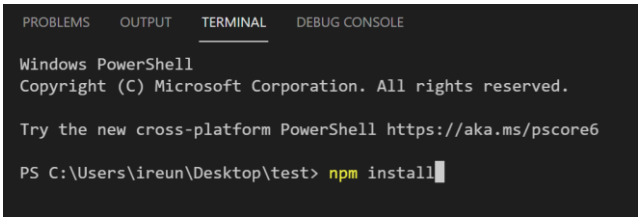






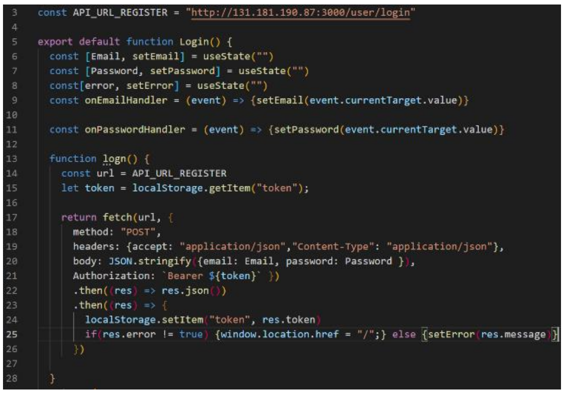
If goes into src folder there are some piles firstly, there is components folder, inside that, there is a folder which are navigation var. Secondly, there is a folder which name is pages. These are as mentioned above, files for each individual pages. And there is App file. This file is main of this application. Every information gathered in this file and processed. And the last one is styles file which is for website designing CSS file.





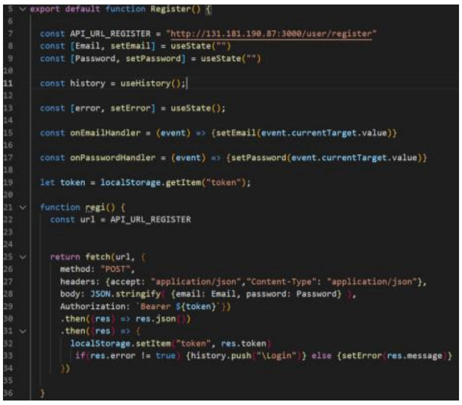
Application could be handled in terminal through code like npm install and npm start than it could be available to open application.

Login



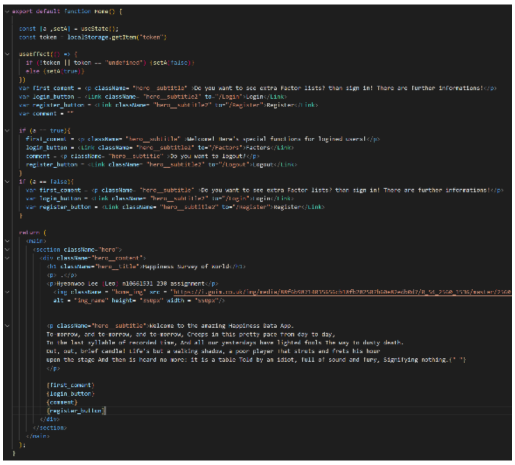
```
const API_URL_REGISTER = "http://131.181.190.87:3000/user/login"

export default function Login() {
  const [Email, setEmail] = useState("")
  const [Password, setPassword] = useState("")
  const[error, setError] = useState("")
  const onEmailHandler = (event) => {setEmail(event.currentTarget.value)}

  const onPasswordHandler = (event) => {setPassword(event.currentTarget.value)}

  function logn() {
    const url = API_URL_REGISTER
    let token = localStorage.getItem("token");

    return fetch(url, {
      method: "POST",
      headers: {accept: "application/json","Content-Type": "application/json"},
      body: JSON.stringify({email: Email, password: Password }),
      Authorization: `Bearer ${token}` })
      .then((res) => res.json())
      .then((res) => {
        localStorage.setItem("token", res.token)
        if(res.error != true) {window.location.href = "/";} else {setError(res.message)}
      })

  }
```

Register



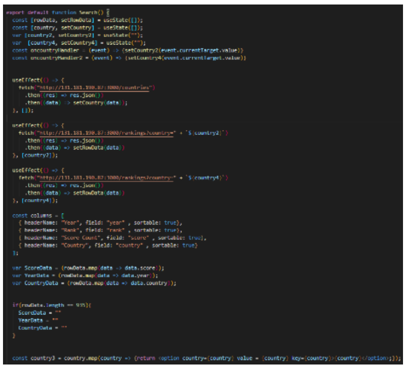
```
export default function Register() {

  const API_URL_REGISTER = "http://131.181.190.87:3000/user/register"
  const [Email, setEmail] = useState("")
  const [Password, setPassword] = useState("")

  const history = useHistory();

  const [error, setError] = useState();

  const onEmailHandler = (event) => {setEmail(event.currentTarget.value)}

  const onPasswordHandler = (event) => {setPassword(event.currentTarget.value)}

  let token = localStorage.getItem("token");

  function regi() {
    const url = API_URL_REGISTER


    return fetch(url, {
      method: "POST",
      headers: {accept: "application/json","Content-Type": "application/json"},
      body: JSON.stringify( {email: Email, password: Password} ),
      Authorization: `Bearer ${token}`})
      .then((res) => res.json())
      .then((res) => {
        localStorage.setItem("token", res.token)
        if(res.error != true) {history.push("\Login")} else {setError(res.message)}
      })

  }
```

Home



Factors

Search



Ranking

```
export default function Ranking() {
  const [rowData, setRowData] = useState([]);

  const [year, setYear] = useState("")
  const onyearHandler = (event) => {setYear(event.currentTarget.value)}

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

  const columns = [
    { headerName: "Rank", field: "rank", sortable: true },
    { headerName: "Country", field: "country" , sortable: true},
    { headerName: "Score Count", field: "score" , sortable: true},
    { headerName: "Country", field: "country" , sortable: true}
  ];
```

Logout

```
export default function Logout() {
  function loggoutt() {

    localStorage.removeItem('token')
    window.location.href = "/";
  }
```

Login service: storage token of user information and authorize from API.

Register service: storage user information to local storage for login.

Logout service: Delete data (token, user information) from local storage.

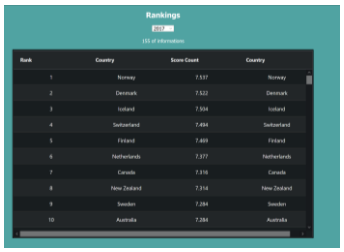
Search: Show data with sort by country search engine and country dropdown bar.

Rank: Show data with Year dropdown bar selection

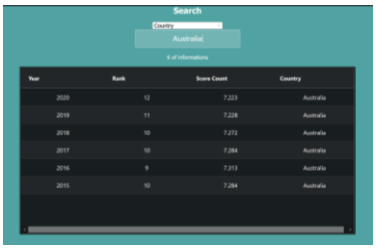
Factors: Show data with Year dropdown bar selection and limit dropdown bar selection.

Test plan

1.

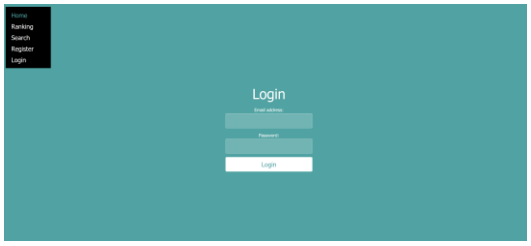


2.

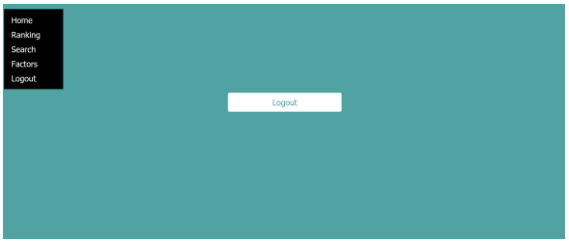


3.

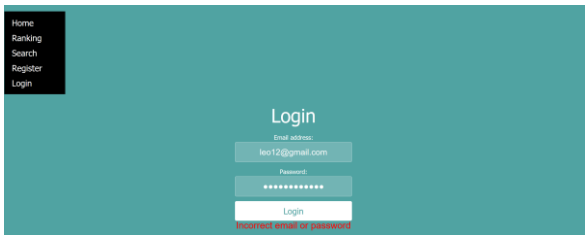
4.



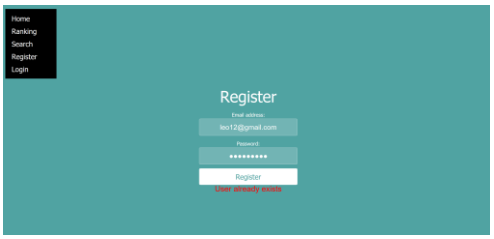
5.



6.



7.



8.

Cases	/Task	/expected outcome	/result	/screenshot.
Positive cases	/Get year value.	/Got year value and show correct data table	/ Pass	/ 1.
Positive cases	/Get country value.	/Got country value and show correct data table	/Pass	/ 2.
Positive cases	/Get year and limit value.	/Got year and limit value and show correct data table	/Pass	/3.
Positive cases	/Click Login	/User created and goes through homepage	/Pass	/4.
Positive cases	/Click Logout	/delete the token that saved in local storage and goes homepage	/Pass	/5.
Negative cases	/Login error	/show error response if login failed	/Pass	/ 6.
Negative cases	/Register error	/show error response if register failed	/Pass	/7.
edge cases	/show graph	/show graph cleary and specifically	/False	/8.

Difficulties / Exclusions / unresolved & persistent errors /

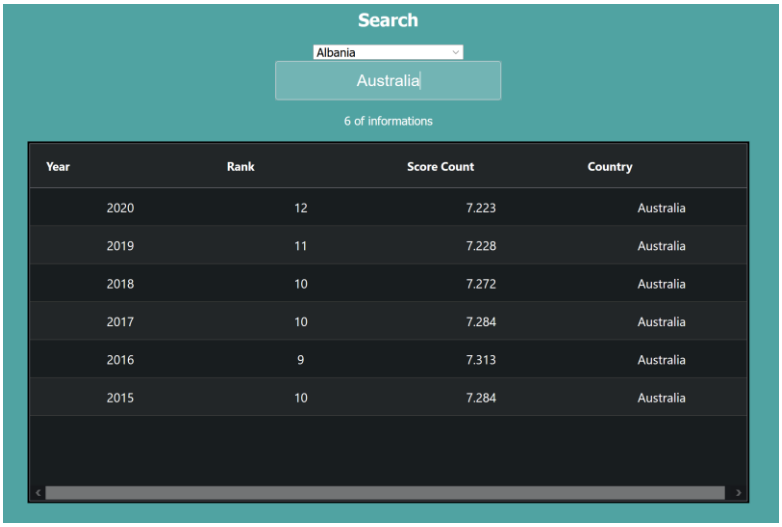
- What were your major roadblocks / how did you resolve them?

The major roadblock of this assignment is catching a null part especially in Factor function.

If webpage loaded, the year input will be null until I select from dropdown bar. And Limit dropdown bar as well. For API, Tear is necessary value to get data. So, I was pretty struggling to handle this part. I put if statement and handled this like if (year == “”) {year = "2020"}.

- Any functionality you didn’t or couldn’t finish and the technical issues encountered.

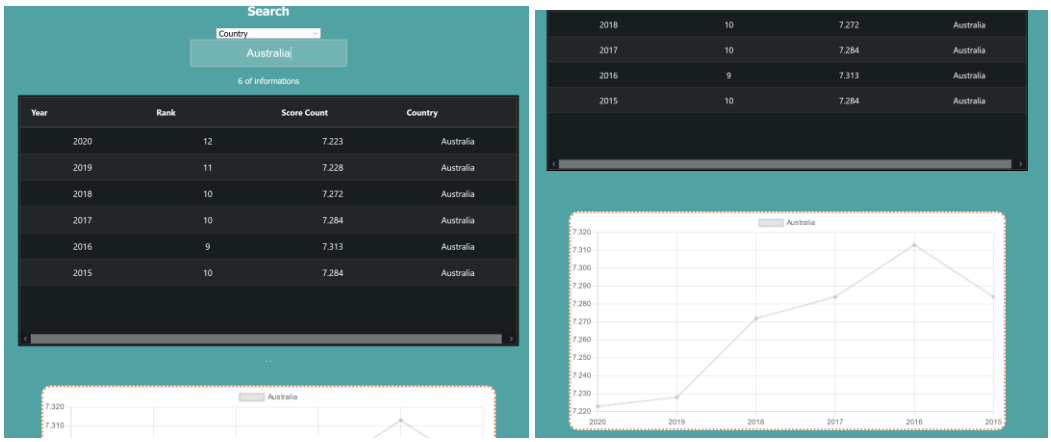
I think it is not that big error, however in search function, if I Search by dropdown bar and after that I search country by search comment input, the data are perfectly showing out. However, the dropdown var’s character is still on that drop down bar. I was trying to solve that in not paradox way, but because lack of my passion and skill, I failed to solve this problem.

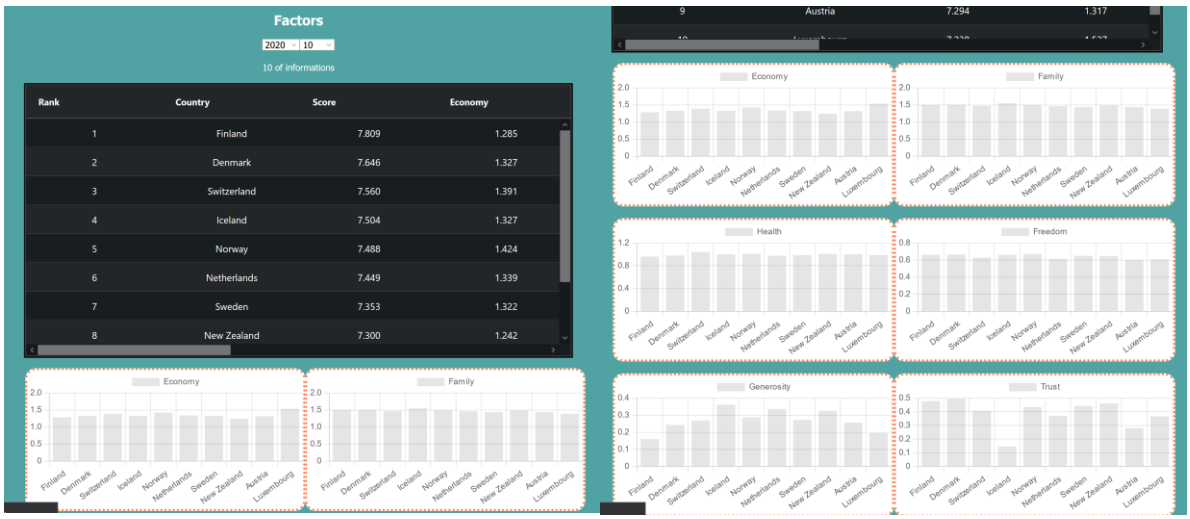


- Are there any outstanding bugs?
  There is no outstanding bug.

Extensions (Optional)

Where could you take this:

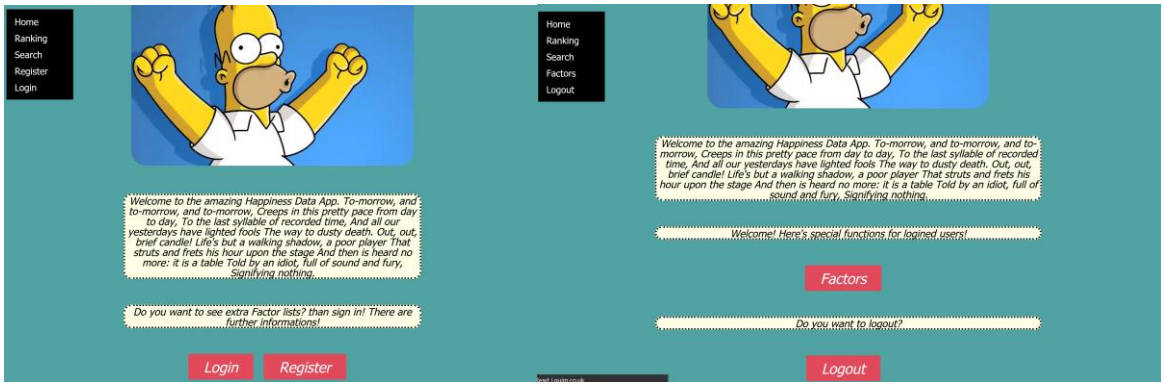




Firstly, as image above, I made graphs that reflects the current value on table to graph. There is one line graph that shows Country's information (year and ranking) on the search page. In the factors part, I made 6 bar graphs that interact with every table values.

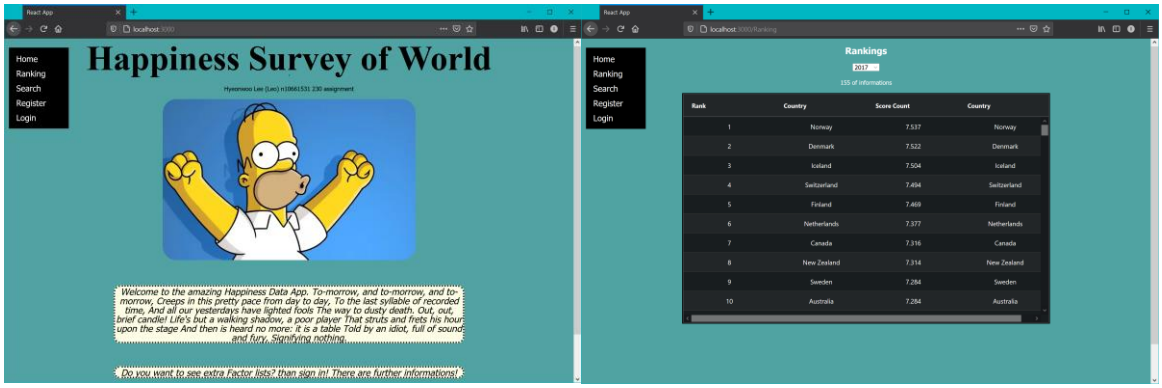
(FamilyData, EconomyData, HealthData, FreedomData, GenerosityData and TrustData)

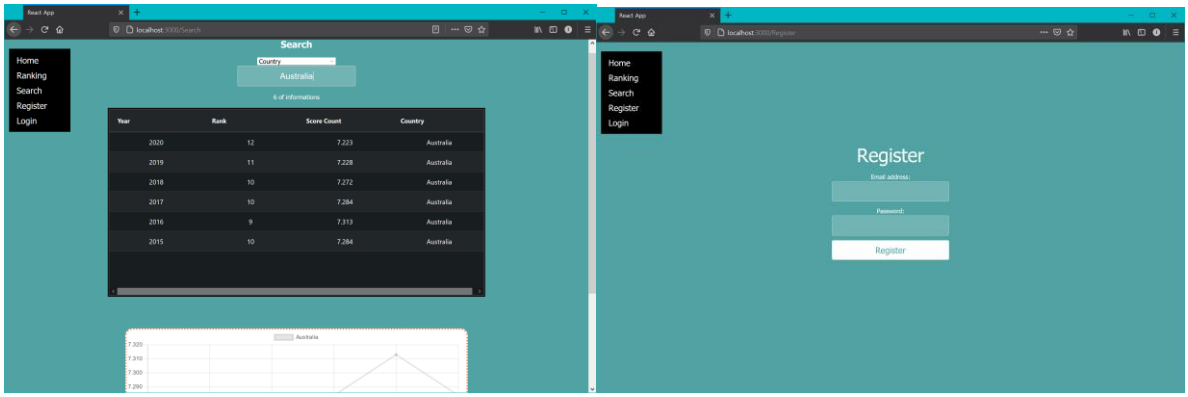
< - - - -> CountryData

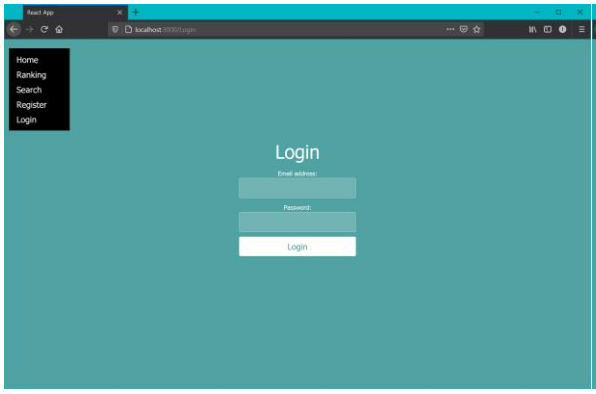


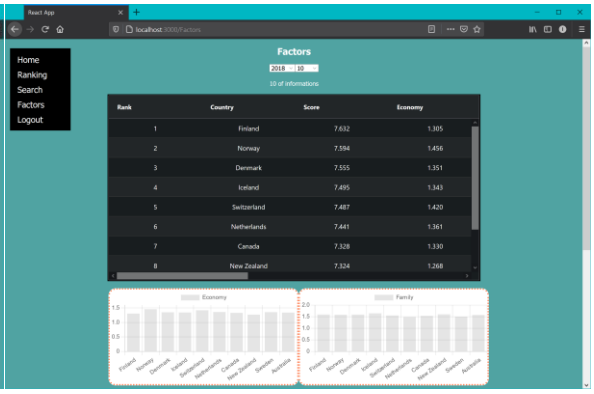
Secondly, In homepage there is a button that ask to user to login and register, and if user logged in, button changes into Factors button and Lougout button. I used same lgic as

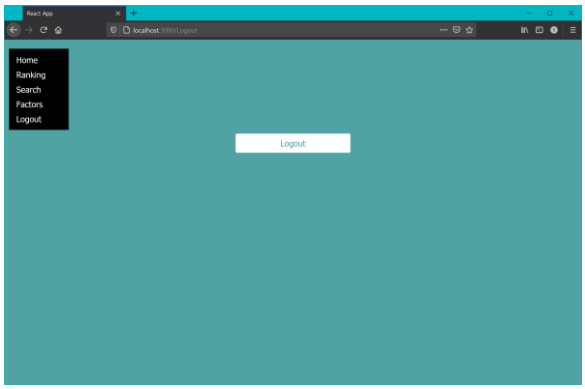
User guide











For user guide, this app is looking for a user who is looking for a Happiness Survey of World. In this website, it provides several functions such like Ranking, search and Factors. But If user wants to access into factor data, user has to login into this app, by this app's login and page. If user does not have an account, user could create account in register page. User also could go to logout after they logged in. In factor page user could see and analyze the data with graphs sorting with years and limits. In ranking page it shows data and it could be short by years. In search page, user could search data by put in country name.

References

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

CSS Tutorial. (n.d.). W3schools.com. Retrieved from https://www.w3schools.com/css/.

HTML Tutorial. (n.d.). W3schools.com. Retrieved from https://www.w3schools.com/html/.

JavaScript Tutorial. (n.d.). Retrieved from https://www.w3schools.com/js/.

Appendices as you require them