BTI425 Assignment 2

Submission Deadline:

Sunday, Feb 27 @ 11:59pm

Assessment Weight:

8% of your final course Grade

Objective:

To continue to work with our "Weather" API on the client-side to produce a rich user interface for accessing data. We would like to redesign the Assignment1 (jQuery) using React.
For this assignment, we will be leveraging our knowledge of React to create an interface for *viewing* weather/country info. **Please Note**: You're free to style your app however you wish, however the following specification outlines how we can use the React-Bootstrap v3 Components. If you wish to add additional images, styles or functionality, please go ahead.

Specifications: The application should have the following features/functionality:

1) When user starts the app, It automatically extract the current date/time/location and display the local weather info (even before user search for any city).
    a. This should be consider as Home Route
2) User can enter “city name” or “city_name,country_code” in any letter-case with any number of space (between city_name and comma and country code).
    - If the city is not exist, the error message should be appeared below the search_box.
    - User should able to press the search button or press “Enter” to start the search.
3) The search result (list of cities) appears on the screen, controlled by pagination (3 record per page).
    a. The search result only includes a summary of cities’ info in table or any other format. Here are the info: Country Flag, City Weather status (proper weather icon -- check API) and current temperature.

        Note: For the Flag, use openWeatherMap as follow: flag = "http://openweathermap.org/images/flags/" + country.toLowerCase() + ".png"; //get flag picture

    b. If user clicked on each city, the program displays the detail info about each city (including: weather condition, current temperature, max/min temperature, wind speed, humidity, pressure, sunrise and sunset, last updated).
        i. You have the flexibility of using any technique in designing the detail-window (can be a separate window or like expand-collapse window)
        ii. Use your creativity ☺

   c. When user visits the detail of each city-weather-info, the app should keep the city_id and all that to the “visited_city_list”. There is a menu item in App Navigational Menu for accessing to “visited_city”. This menu-item is empty at the beginning. You can consider a limit for the numbe rof item in this list. User can select any city from the list of “visited_city” and check the details again (without entering city_name in the search box)
      Hint→This may require a visited_ROUTE with “:id”

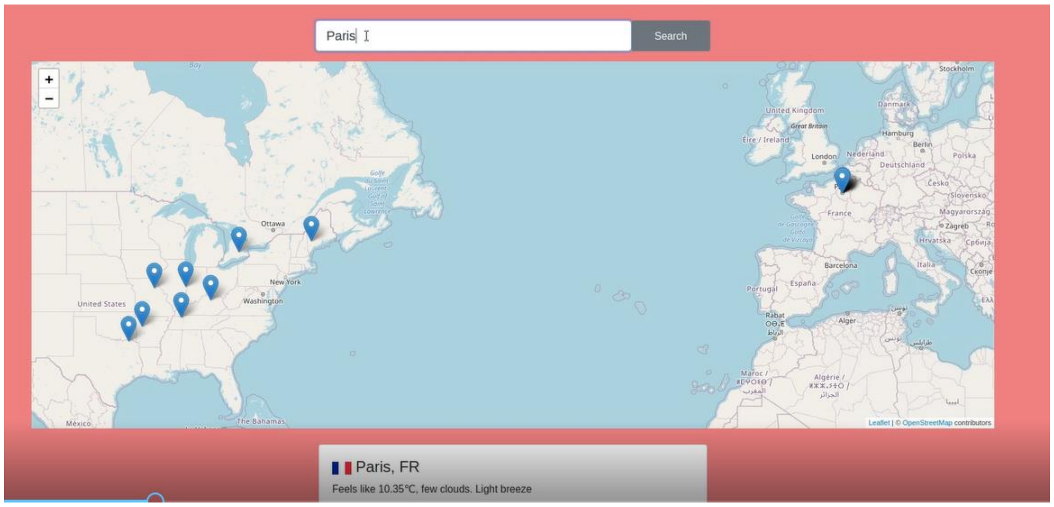
4) The App should design to have minimum API call.
   a. Note: While it is important to get the updated weather info, but wherever possible, try to cache the data to reduce API calls.
   b. You may use any of the following API calls:
      i. api.openweathermap.org/data/2.5/weather?
      ii. api.openweathermap.org/data/2.5/find?
      iii. api.openweathermap.org/data/2.5/group?
   c. Use &cnt : Number of cities around the point that should be returned. The default number of cities is 5, the maximum is 50.
5) Wherever possible, use React Bootstrap for implementing the app layout
6) Add NavBar to assist user to navigate between Home page and “Visited City”. Use proper Route to manage user navigation (consider NotFound route as well)
   a. To begin, add the following import statements
      i. { Navbar, Nav, NavItem, NavDropdown, MenuItem, FormGroup, FormControl, Grid, Row, Col } from react-bootstrap
      ii. { Link, Routes, Redirect, Route } from react-router-dom
      iii. { LinkContainer } from react-router-bootstrap
   b. we may add the following "state" values to our App component:
      i. recentlyViewed – default value: []
      ii. searchId – default value: ""
   c. To track which cities have been recently viewed by the client, we may add two internal functions to the App component; the first of which is called called viewedCity(id) with the following specification:
      i. Creates a copy of the recentlyViewed array, ie: "allRecentlyViewed"
      ii. Pushes the value of "id" into the "allRecentlyViewed" array only if it's not already in the array
   d. To desing the NavBar, you may use the following JSX code (in the return value of our App component) This will allow us to show a navbar built using React-Bootstrap 3 components.

```
<div>
  <Navbar inverse collapseOnSelect staticTop>
    <Navbar.Header>
      <LinkContainer to="/">
        <Navbar.Brand>
          BTI425 - Weather
        </Navbar.Brand>
      </LinkContainer>
      <Navbar.Toggle />
     </Navbar.Header>
     <Navbar.Collapse>
      <Nav>
      <LinkContainer to="/Cities">
       <NavItem>All Cities</NavItem>
      </LinkContainer>
       <NavDropdown title="Previously Viewed" id="basic-nav-dropdown">
       {recentlyViewed.length > 0 ?
         recentlyViewed.map((id, index)=>(
          <LinkContainer to={`/City/${id}`} key={index}>
            <MenuItem >City: {id}</MenuItem>
          </LinkContainer> )) :
         <MenuItem>...</MenuItem>}
       </NavDropdown>
      </Nav>
      <Navbar.Form pullRight>
       <FormGroup>
         <FormControl type="text" onChange={updateSearchId} placeholder="City ID" />
       </FormGroup>{' '}
       <Link className="btn btn-default" to={"/City/" + searchId}>Search</Link>
      </Navbar.Form>
     </Navbar.Collapse>
    </Navbar>
   </div>
```

7) To manage Async Ajax call, you may need a utility function called getData(page) which returns a **promise** that only resolves once the following ajax request is completed.
   a. After the component has been "mounted", we must call the "getData(page)" function
   b. You may use async/await wherever is required ☺

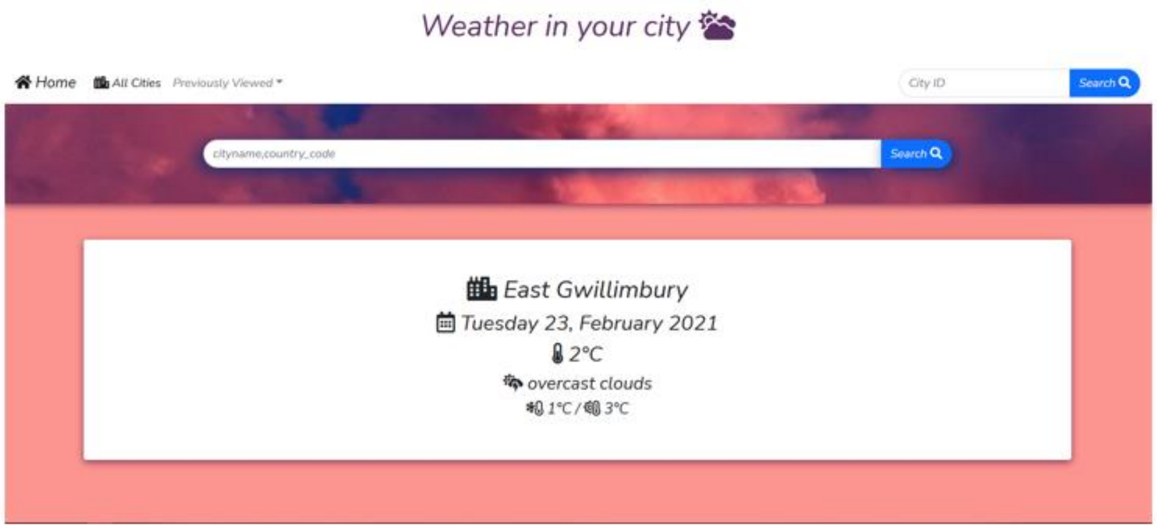
8) Deploy the app into Heroku.

9--bonus) Locate/display the city on the map. This feature adds a map whenever detailed weather info for a particular city is being displayed.

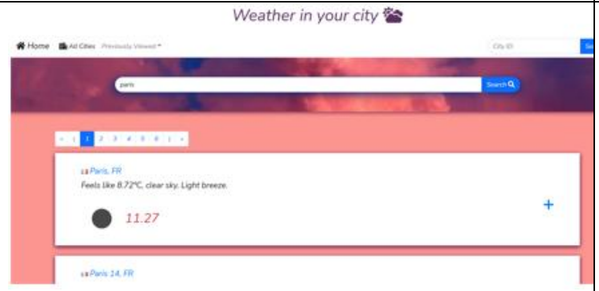
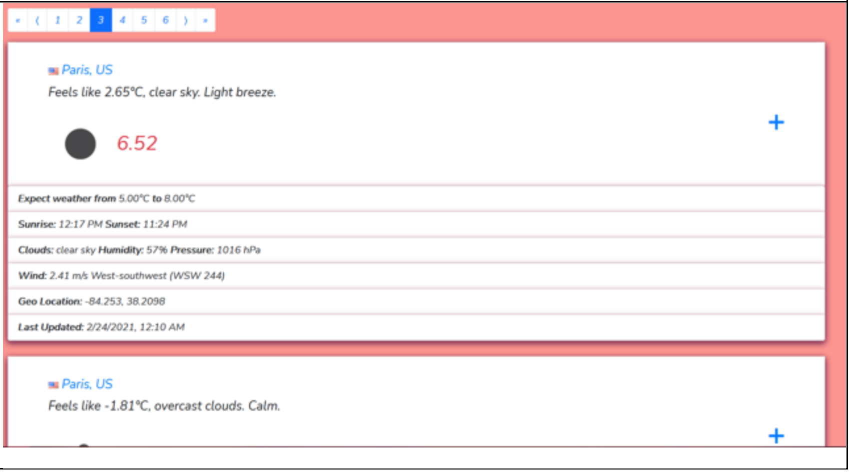


Application at a glance: (the following snapshots only explain the layout to the style)

Home page:



East Gwillimbury
Tuesday 23, February 2021
2°C
overcast clouds
1°C / 3°C

Search for a city (summary of weather info)	When click on '+' for more info
	

Paris, FR
Feels like 8.72°C, clear sky. Light breeze.
11.27

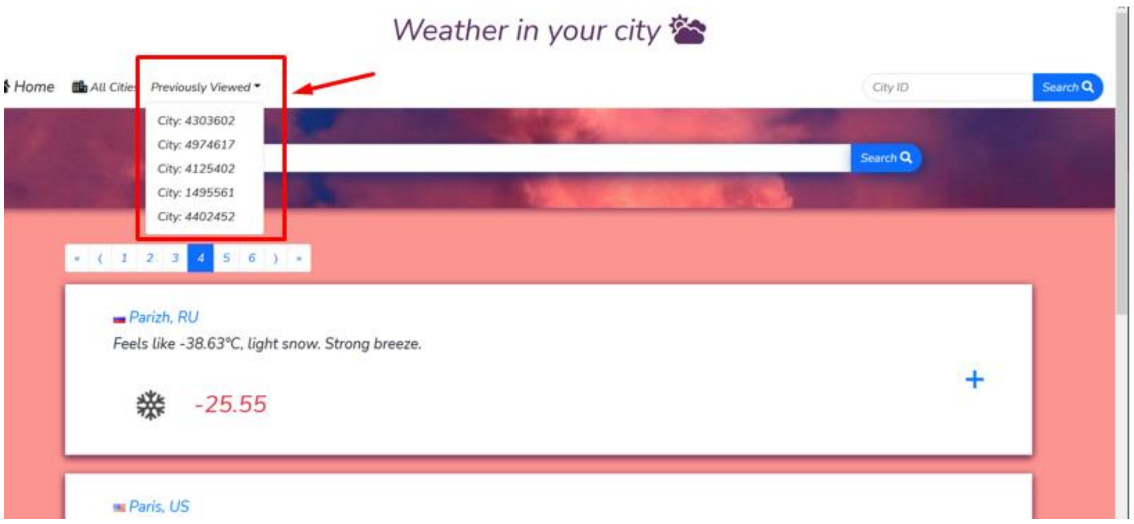
Paris 14, FR

Paris, US
Feels like 2.65°C, clear sky. Light breeze.
6.52

Expect weather from 5.00°C **to** 8.00°C
Sunrise: 12:17 PM **Sunset:** 11:24 PM
Clouds: clear sky **Humidity:** 57% **Pressure:** 1016 hPa
Wind: 2.41 m/s West-southwest (WSW 244)
Geo Location: -84.253, 38.2098
Last Updated: 2/24/2021, 12:10 AM

Paris, US
Feels like -1.81°C, overcast clouds. Calm.

List of visited cities



Parizh, RU
Feels like -38.63°C, light snow. Strong breeze.
-25.55

Paris, US

Rubric:

- Use OpenWeatherMap API (JSON/XML data), collect/process all required data. (1)
- Error Handling(API, search keyword) (1)
- Layout (Menu + Show result in Summary/Detail layout) (1)
- Use React Bootstrap in design (1)
- Use React Route + NavBar + visited_cities(1.5)
- Apply React Pagination (Display three records per page) (1.5)
- Try to reduce the number of API calls (0.5)
- Coding style + Presentation (walkthrough, challenges, self evaluation) (0.5)

Assignment Submission:

- Add the following declaration at the top of your index.js file

```
/*********************************************************************************
*  BTI425 – Assignment 2
*  I declare that this assignment is my own work in accordance with Seneca Academic Policy.
*  No part of this assignment has been copied manually or electronically from any other source
*  (including web sites) or distributed to other students.
*
*  Name: ______________________ Student ID: __________________ Date: ____________________
*  Heroku Link: ____________________
*
********************************************************************************/
```

- Compress (.zip) the files in your Visual Studio working directory **without node_modules** (this is the folder that you opened in Visual Studio to create your client side code).

Important Note:

- Submitted assignments must run locally, ie: start up errors causing the assignment/app to fail on startup will result in a **grade of zero (0)** for the assignment.
- LATE SUBMISSION policy: There is a deduction of 10% for Late assignment submissions, and after three days it will grade of zero (0).
- Assignments should be submitted along with a video-recording which contains a detailed walkthrough of solution. Without recording, the assignment can get the maximum of 1/3 of the total