Web Tech Assignment 2

Assignment 2

This assignment explores using ReactJS and the React Router to implement different views of the Environmental Data used in Assignment 1. All work is to be done **individually** and must be **your own work**. You **must not discuss or share your** **work with anyone else**. You **must only submit code you have personally written** (except for utilising bootstrap html template examples). If you use other resources, **you must include direct links to those resources** as comments in the relevant files.

Each person is to complete the Assignment by finishing the a2reactapp ReactJS application that makes use of the Web APIs in the ASP.NET Core Web application. There are several tasks to complete as outlined below. You can only use the basic features of React and the React Router – **you must not use other features like React-Bootstrap components, or other UI components not covered in the course** such as Table components etc. Your task is to implement a Single Page ReactJS Application (SPA) as explored in the practicals by creating your own components and making use of the React Router Link, BrowserRouter, Routes, Route and Output components etc. The aim is to use fetch requests and not have the page refresh between different views of the environmental data

Contents

[Assignment 2 1](#_Toc148518748)

[1. Repository Rules 2](#_Toc148518749)

[2. Download the Assignment Web Application 2](#_Toc148518750)

[3. Install the ReactJS Application 2](#_Toc148518751)

[4. Part A Submission: Design your Routes, Base Components and Basic Implementation 2](#_Toc148518752)

[Tasks 3](#_Toc148518753)

[Task 1 Plan your Routes 3](#_Toc148518754)

[**A\_Regions Controller** 3](#_Toc148518755)

[Task 2 Region List [10%]: 3](#_Toc148518756)

[**B\_Countries Controller** 3](#_Toc148518757)

[Task 3 CountryList [20%] 3](#_Toc148518758)

[**Due Date and Presentation of Part A** 3](#_Toc148518759)

[**Remainder of the Assignment** 4](#_Toc148518760)

[Task 4 Country Temperature Data [15%] 4](#_Toc148518761)

[Task 5 Country Emission Data [20%] 4](#_Toc148518762)

[**C\_Cities Controller** 4](#_Toc148518763)

[Task 6 City Search and Data [10%] 4](#_Toc148518764)

[Task 7 City Air Quality Data [15%] 4](#_Toc148518765)

[HTML Validation & JS Errors (10%) 4](#_Toc148518766)

[**Using the** SWAGGER **Web API Interface** 5](#_Toc148518767)

[Comitting Changes 6](#_Toc148518768)

[Uploading your Project. 7](#_Toc148518769)

## Repository Rules

Alright, procrastination here is like hoping a desert will magically transform into a lush rainforest overnight. We're aiming for a blossoming orchid, not some frantic last-minute shrub planting. This project is all about nurturing, just like a garden needs love, this code needs thought and care. You are expected to work on the project for the next 3 weeks during your allocated practical time and for homework.

And guess who's making a comeback? It's our trusty old friend GIT, the environmental Savior of version control. Use it to shield your code from the chaos of forgotten changes. You must have the following level of commits to obtain marks for the assignment:

Here's the scoop on scoring marks:

1. **Day is Done**: No cramming allowed! Spread those commits like seeds all over the place during the day. Show us a steadily growing and flourishing code garden, not some last-minute greenhouse mayhem.
2. **The 10-Line Eco-Boost**: Commit your code like you're planting quick-growing bamboo. Keep it swift and effective, no need for a lengthy, tangled vine of code.
3. **Functionality**: Every time you add a new component, commit it like you just planted a whole bed of fresh herbs!
4. **Route Alert**! – make a new route, test it, commit it like you mean it.
5. **The Green Break**: Take a break, go hug a tree, or have a moment with your Favourite potted plant. And don't forget to give your code a breath of fresh air with a commented commit before you take the break!
6. **The Bug Weed-Out**: Bugs are pesky garden pests. Show them the way out with a commit and bask in the glory of error-free code"

## Download the Assignment Web Application

A MVC application has been pre-built complete with the Environmental Data Model and custom models. You will need to **update the connection string** in the web application to use your local copy of the Environmental database from Assignment 1.

## Install the ReactJS Application

Using VS-Code OR Visual Studio 2022 You will need to **install** the ReactJS application which has already been created for you in the above MVC application.

1. Using the Terminal Window of Visual Studio with the MVC application loaded, navigate to the a2reactapp folder:  
   > cd assig2/a2reactapp
2. Install the react app (this will re-create the missing node\_models folder)  
   > npm install
3. Once installed, test the reactJS application works  
   > npm start

## Part A Submission: Design your Routes, Base Components and Basic Implementation

* For the **progress submission** you will need to have **all your planned routes** in place and test components showing the component + route name. Where parameters and props are involved, include test values and show these on the relevant components
* You will need to have Tasks 1-3 completed: a basic blueprint of your routing, Regions API implemented and be able to list countries associated with a selected region. During your scheduled practical class in Week 13 you must present your code for Part A (Tasks 1-3) in class to the tutor and answer questions on the functionality to obtain marks for the whole assignment.

Your ReactJS application will need to be constructed as a Single Page Application (SPA) as per the practical example and must:

* Use at least one component for each Task outlined in this document
* A main menu with two links, one each for Regions and Countries.
* Make use of the **ReactJS V6.4.x router**. (assumed to be already installed)
* Your new ReactJS application will need to consume and show the JSON data provided from the Web API as outlined in the following tasks. You must not use any additional React Plugins to complete the assignment (axios, bootstrap/UI components etc). You must only use basic ReactJS and standard HTML elements. For full marks, you need to make educated decisions about your component structures, variables used etc and data needs to be presented aesthetically using bootstrap classes (the more effort and research into the structure the better the mark).

# Tasks

### Task 1 Plan your Routes

Look carefully at the Environmental Data Web API provided (via the SwaggerUI). Note the parameters that each API accepts and the type of data it returns. Using this knowledge and your experience from Assignment 1, plan the ReactJS routes, parameters and components you think you will need to reproduce the Region -> Country -> City navigation along with Country -> Temperature + Emission Data and City -> Air Quality Data. Create Test components containing a basic heading and parameter print statements to test your routes work and pass the necessary values. Creating an initial plan will really help simplify your design before adding the more complicate fetch requests and dealing with the JSON data. **You must do this to receive any marks for the assignment.**

## **A\_Regions Controller**

### Task 2 Region List [10%]:

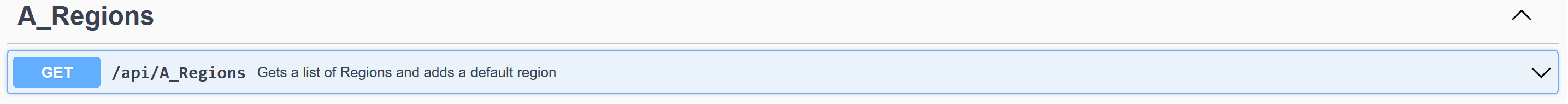
This page allows users to browse a list of Regions and their corresponding details. You need to show all of the provided regional data in a professional and aesthetically pleasing manner.

This Task must:

* Consist of at least 1 ReactJS component and make use of all the available Json data
* Make use of bootstrap cards or similar which are clickable (avoid using buttons where possible)
* Clicking on the card uses the react router to navigate to a Countries List page with the correct country data showing

**Web API Controller:** A\_RegionsController

**Controller Actions:**



## **B\_Countries Controller**

### Task 3 CountryList [20%]

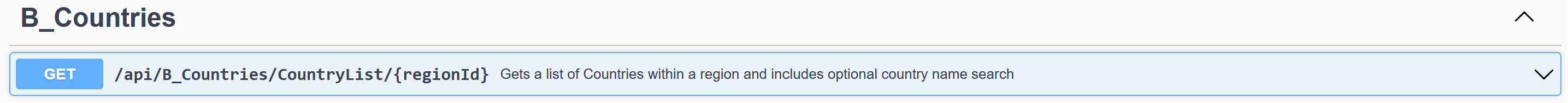
Selecting a Region on the previous page navigates to this page (without the page re-loading in the browser) which clearly shows the Region selected, the name, image and any other detail for the selected region (check the API for the available information). In addition, the page should have a textbox that allows the user to search for specific countries.

This Task must:

* Consist of at least 1 ReactJS component and make use of all the available Json data
* Load showing all the relevant countries for the selected region
* Make use of all the available Json data in a professional and aesthetically pleasing manner. Where temperature/emission data is not available, then don’t provide links to view the data
* A button to navigate back to the original search page

**Web API Controller:** B\_CountriesController

**Controller Actions:**

****

## **Due Date and Presentation of Part A**

During your scheduled practical class in Week 13 you must present your code for Part A (Tasks 1-3) in class to the tutor and answer questions on the functionality to obtain marks for the whole assignment. You must also have an outline for how you expect to finish Part B. Everyone must present their work for assessment in this way to receive a final grade for this assessment. External students will be asked to make a time on Zoom.

## **Remainder of the Assignment (Part B)**

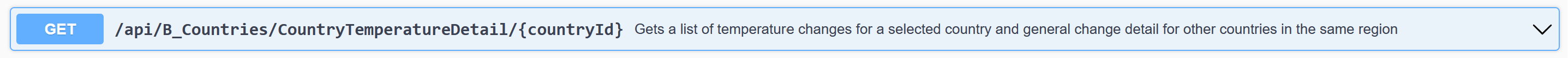
### Task 4 Country Temperature Data [15%]

For each country where temperature data is available, a link must be provided to view the Temperature Data for the selected country

This Task must:

* Consist of at least 1 ReactJS component for the temperature data and make use of all the available Json data
* Make use of all the available Json data in a professional and aesthetically pleasing manner.
* Depending on the layout, a button to navigate back to the original country list may be required loading the original country list

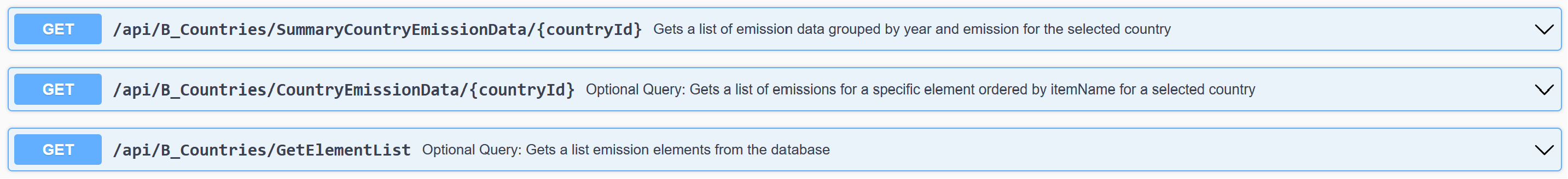
**Web API Controller:** B\_CountriesController

**Controller Actions:  
**

### Task 5 Country Emission Data [20%]

This page is accessible from the Country List assuming there is emission data and must:

* Consist of at least 1 ReactJS component and make use of all the available Json data
* Display all the available emission data grouped by year and the element measured.
* For maximum marks, make use of the additional two API actions involving Element Name (GetElementList) and the emission data breakdown (CountryEmissionData) which accepts a selected Element from the Element List. You will need to use a select element or other approach to implement this search.



## **C\_Cities Controller**

### Task 6 City Search and Data [10%]

This page is accessible from the Country List assuming there are cities available for the country and must:

* Consist of at least 1 ReactJS component and make use of all the available Json data
* Display all the cities for the selected country and the available Json data.
* Provide a back button to the original country list
* Implement a search textbox for searching through city names (e.g. USA has 300+ cities listed)
* Where Air Quality Data is available, provide a link to view this data

### Task 7 City Air Quality Data [15%]

This page is accessible from the City List assuming there is air quality data and must:

* Consist of at least 1 ReactJS component and make use of all the available Json data
* Display all the available air-quality Json data in a professional and aesthetically pleasing manner.
* Provide a back button to the original city list

### HTML Validation & JS Errors (10%)

Your pages will need to generate valid HTML and be free from JavaScript errors. We suggest you use the [W3C HTML validator](https://validator.w3.org) and your web browser debugging tools to check this! You need to use appropriate class names, bootstrap classes where possible and avoid using IDs where you can.

# **Using the** SWAGGER **Web API Interface**

1. Run the Assignment 2 MVC Web Application and navigate to the Web API to view the Swagger interface. It will list the actions in the available controllers  
   A screenshot of a computer

   Description automatically generated
2. Expand the desired action to view a list of the available parameters, their requirements and an example JSON schema produced:  
   A screenshot of a computer

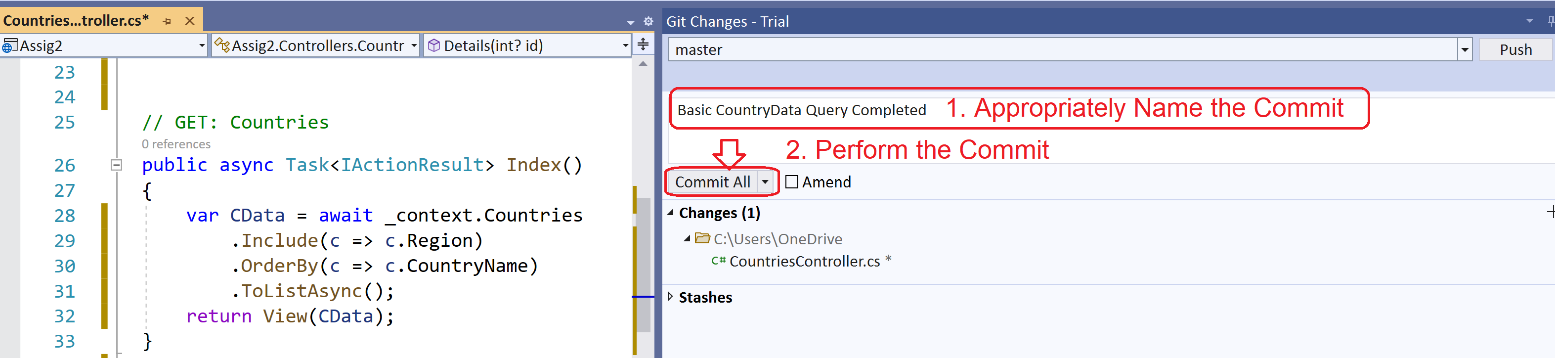
   Description automatically generated
3. Click on the “**Try it out**” button to activate the parameters, enter your test values then press “Execute” to view the structure of the data returned. The **Request URL** property will show you the exact URL required to trigger the Web API action based on the parameters passed. Examples are also available in the controller actions as comments. Check each Parameter for a red Asterix (\*) indicating if it is a non nullable [required] parameter. The **Server Response** property will show the JSON data returned:  
   A screenshot of a computer

   Description automatically generated

# Committing Changes

Committing your progress is as easy as 1,2,3!  
If your GIT window is not active, simply go to the GIT menu and select the “Commit or Stash” option.   
Then:

1. Name your commit based on what you did
2. Commit the change!



# Uploading your Project.

Once you have completed your project.

1. **Close Visual Studio** and navigate to the folder containing your whole web application
2. A screenshot of a computer

   Description automatically generatedZip up the entire web application. This folder must include your .sln file, your project directory, git directory and git support files as pictured. This is important:

A screenshot of a computer

Description automatically generated

1. Remove the **node\_modules** folder from the zip file (located under Assig2/a2webapp). If you’re feeling confident about your project, you can remove the node\_modules before zipping. You will then have to npm install to get them back :)
2. Upload to the course website