Web Tech Assignment 2

Assignment 2

This assignment explores using ReactJS to implement the Expiation Code Search and Local Service Area Expiations from Assignment 1. The assignment is broken into two equally weighted parts as outlined below. This assignment is for all students regardless of their study mode and the assignment **must be done individually**. As with any assignment, if you utilise any examples from online in your code, you must acknowledge the original source by providing a comment reference above the relevant sections.

Each person is to complete the Assignment 2 by creating a new ReactJS application that makes use of the Wab APIs in the SAExpiationsA2 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.

* Part A: Expiation Code Search and Monthly breakdown
* Part B: Local Service Area listing and Expiations breakdown

Contents

[Assignment 2 1](#_Toc117022231)

[1. Download the Assignment Web Application 1](#_Toc117022232)

[2. Create a new ReactJS Application 1](#_Toc117022233)

[Tasks 2](#_Toc117022234)

[**Part A Controller [45%]** 2](#_Toc117022235)

[Task 1 Expiation Code Search: 2](#_Toc117022236)

[Task 2 Expiation Code Detail 2](#_Toc117022237)

[**Due Date and Presentation of Part A** 3](#_Toc117022238)

[**Part B Controller [45%]** 3](#_Toc117022239)

[Task 3 Local Service Area Expiations List 3](#_Toc117022240)

[Task 4 Local Service Area Expiations Detail 4](#_Toc117022241)

[HTML Validation (10%) 4](#_Toc117022242)

[**Using the SWAGGER WebAPI Interface** 4](#_Toc117022243)

## Download the Assignment Web Application

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

## Create a new ReactJS Application

Using VS-Code OR Visual Studio 2022 You will need to create a new ReactJS application independent of the above WebAPI. Your ReactJS application will need to be constructed as a Single Page Application (SPA) and must:

* Use at least one component for each Task outlined in this document
* A main menu with two links, one each for Task 1 and 3 (the Expiation Code Search and the Local Service Area Listing).
* Make use of the **ReactJS V6.4.x router**.
* Your new ReactJS application will need to consume and show the JSON data provided from the WebAPI 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 styles the better the mark).

# Tasks

## **Part A Controller [45%]**

### Task 1 Expiation Code Search:

This page allows users to browse a list of expiation codes, their descriptions and to search for specific codes/descriptions using words like “Exceed” and “fail”. You can provide a single textbox for this search..

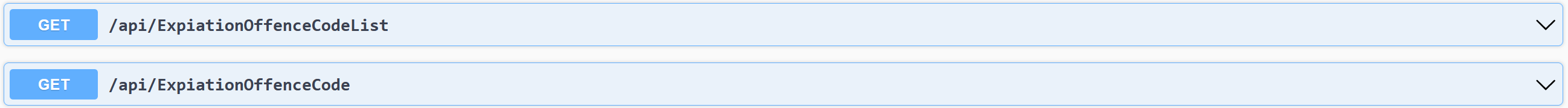
When the page first loads, the codes and descriptions should be in order of the Expiation Code.

This Task must:

* Consist of at least 1 ReactJS component
* The expiation code search text box must include [auto-complete](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/datalist) functionality implemented using a basic HTML5 data list. It should sow the list of matching expiation codes as the user enters search terms.
* Have a button to trigger the search request to the WebAPI
* The page must initially load showing the expiation codes and descriptions in alphabetical order
* [Hovering over a row](https://getbootstrap.com/docs/5.0/content/tables/) in the table/card showing the expiation code + description highlights the row/card
* Clicking on the row/card uses the react router to navigate to a Detail page with the correct expiation data showing
  + The Expiation Code, description and category (if available) selected
  + The List as outlined in Task 2

**WebAPI Controller:** PartAController

**Controller Actions:**



### Task 2 Expiation Code Detail

Selecting a Expiation Code on the previous page navigates to this page (without the page re-loading in the browser) which clearly shows the Expiation Code selected, the description and any other detail for the selected expiation code (check the API for the avilable information). In addition, the page should have a drop down list that defaults to the current year. This can be programmed to show the current year minus 2 (ie, have the options of selecting 2020, 2021, 2022) or use the year list option available through the API. You need to make sure the current year is generated from the server date, and then the other years calculated from that (so next year it defaults to 2021, 2022, 2023)

This Task must:

* Consist of at least 1 ReactJS component
* Provide an option to select a year to view the breakdown of notice status descriptions for the selected expiation offence code
* Load showing a monthly breakdown of expiations counts for the selected expiation code, year and grouped by the Notice Status Description.
* Each Month name should appear only once in the table/Card and then each status description with count under that month.
* Clearly colour the Month rows and apply [bootstrap styles](https://getbootstrap.com/docs/5.0/content/tables/) or your own custom CSS in the App.css file for the notice status descriptions so the same status can be easily identified for each month (e.g. Withdrawn is always a pale red background etc). Note that the API can be used to generate classes of the format NSD\_x for each status description to simplify styling.
* Each month should be accompanied by a Total number of expiations matching the selected expiation code for the month (irrelevant of status).
* At the bottom of the page, the total number of expiations matching the selected code for the year-to-date
* Must only load if an expiation code is passed from the previous page
* A button to navigate back to the original search page

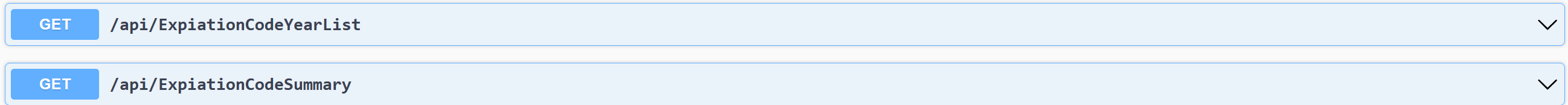
**Example: Expiation Code Detail** showing grouping by Month and Notice Status Description (for demo purposes only).

*Note: Custom Classes starting with "NSD\_x" have been implemented in the controller if you want to use them. Simply set the private property IncludeClassNames = true. You can then use the App.css file to implement backaground colours for your rows instead of using the bootstrap classes.*

|  |  |  |
| --- | --- | --- |
| **Month** | **Notice Status Description** | **Status Count** |
| **January** |  | Total: 618 |
|  | COURTS ENFORCEMENT (pending or enforced) | 31 |
|  | COURTS RELIEF STATUS | 47 |
|  | EXPIATED | 131 |
|  | ISSUED OR CAUTIONED | 401 |
|  | WITHDRAWN | 8 |
| **February** |  | Total: 579 |
|  | COURTS ENFORCEMENT (pending or enforced) | 19 |
|  | COURTS RELIEF STATUS | 43 |
|  | EXPIATED | 111 |
|  | ISSUED OR CAUTIONED | 398 |
|  | WITHDRAWN | 8 |
|  |  | Year To Date: 1,197 |

**WebAPI Controller:** PartAController

**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 in class to the tutor and answer questions on the functionality. 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 as previously.

## **Part B Controller [45%]**

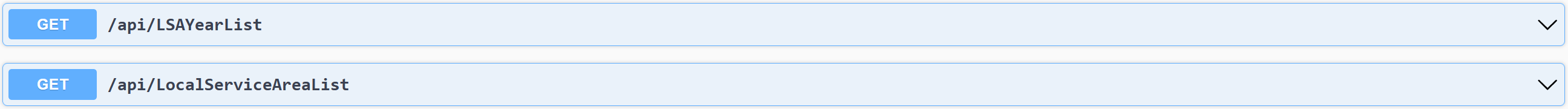
### Task 3 Local Service Area Expiations List

This page is accessible from the main menu and shows a List of Local Service Areas (in alphabetical order) and the total number of expiations for Aach area with the option to filter by the selected year. As with the previous View, this page must make professional use of bootstrap and:

* Consist of at least 1 ReactJS component
* Contain a drop down list that defaults to the current year. This can be programmed to show the current year minus 2 (ie, have the options of selecting 2020, 2021, 2022) or use the API LAS year list option.
* Clicking on an Area name uses the React Router to navigate the user to a Details view as outlined in Task 4 without the page reloading

**WebAPI Controller:** PartBController

**Controller Actions:**

****

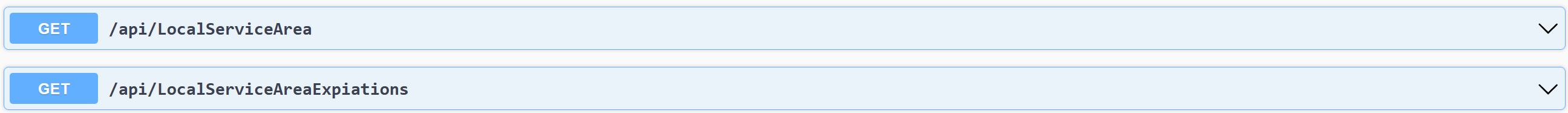
### Task 4 Local Service Area Expiations Detail

This is the equivalent of the Expiation Codes Detail View except you are displaying the number of expiations for each Expiation Code in the selected Area and year. The page must utilise bootstrap styles and load showing:

* Consist of at least 1 ReactJS component
* The Selected Area information
* The Selected Year
* The total number of expiations
* A list showing each Expiation Code, Description and total number of expiations for that code
* Order the results by Expiation Code.
* A button to navigate back to the original search page

**WebAPI Controller:** PartBController

**Controller Actions:**



## 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 WebAPI Interface**

1. Run the SAExpiationsA2 WebAPI Application to view the Swagger interface. It will list the actions in the available controllers  
   Background pattern

   Description automatically generated
2. Expand the desired action to view a list of the available parameters, their requirements and an example JSON schema produced:  
   Graphical user interface, application

   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 structure of the URL required to trigger the WepAPI action based on the parameters passed. Check each Parameter for a red astrix (\*) indicating if it is a non nullable [required] parameter. The **Server Response** property will show the JSON data returned:  
   Graphical user interface, text, application

   Description automatically generated