**Score: \_\_\_\_\_**

**WA6 – Back-End: Dynamic Content**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

**Introduction:**

In today’s class we learned about a number of ways in which dynamic content can be generated on the back-end (i.e. by code running on the web server). These included page generation, server pages and API endpoint handlers. You saw a number of examples of server pages using Java Server Pages and an example of an API endpoint handler in JavaScript using the Express framework. Today’s activities give you some additional practice with these ideas and in particular the implementation and use of Express API endpoints.

**Back-End Concepts:**

🔑 1. In a few sentences of your own words, briefly describe the difference between back-end content that is static and back-end content that is dynamic.

🔑 2. In a few sentences of your own words, briefly describe the difference between page generation and server pages as approaches to generating dynamic content on the back-end.

🔑 3. In a few sentences of your own words, briefly describe the ways in which content is made dynamic on the front-end and the ways in which it is made dynamic on the back-end.

**The /sample API Endpoint:**

The server.js file in the api directory gives the JavaScript function that implements the /sample endpoint that we saw in class. This API endpoint returned a JSON structure containing some sample data as a first example of how to create an API endpoint handler on the server side.

🔑 4. The api2.html file in the www directory provides a page that makes a request to the /sample endpoint and displays most of the data that is returned. You can access the rendered version of the api2.html file by clicking on its link in the page at http://localhost:8080/www.

Click the button in the api2.html page to make the API call to the /sample endpoint and render the results. Paste a screenshot of the rendered HTML after the API call is made as your answer to this question.

🔑 5. The server.js file in the api directory in the Comp256Web container contains the source for the code that is run on the server when the /sample endpoint is accessed.

Open the server.js file in a text editor and find source code for the JSON object that is returned from the /sample endpoint.

a. Copy the source code for that object here as your answer to this question.

b. What part of the JSON object from part a is not rendered in the page in question #4?

🔑 6. In question #5 you should have noticed that the second course in classList array in the JSON object was not rendered into the page. Find the api2.html file in the www directory in the Comp256Web container and open it in a text editor.

Study the code in the api2.html file and then add lines to the sampleRequest function so that the data from the second course is also displayed when you click the button. Hint: There are already span elements with id’s defined for the content, so you only need to modify the JavaScript function to use them. Be sure to reload the page and test your changes. Paste just the new lines of JavaScript that you added to the sampleRequest function as your answer here.

🏆 7. Typically courses at a college or universities will have an instructor associated with them.

a. Modify the code that handles the /sample endpoint in the server.js file so that each course also has an instructor. Hint: Add another field and value for the instructor to each object in the classList array. Paste your modified JSON object as your answer here.

b. Now that the back-end includes information about the instructor for each class, the front-end should also display that information. Modify the front end in api2.html so that the instructor information for each class is also displayed. Paste the full contents of your modified api2.html page as your answer here.

**The /customers/bycountry API Endpoint:**

The /customers/bycountry endpoint is also implemented in the server.js file in the api directory. This API endpoint uses our sample database to return a JSON structure containing all of the customers in a country that is specified by a query parameter.

🔑 8. The api3.html file in the www directory provides a page that makes a request to the /customers/bycountry endpoint and displays the names of the customers that are returned. You can access the rendered version of the api3.html file by clicking on its link in the page at http://localhost:8080/www.

Click the button in the api3.html page to make the API call and render the results. Paste a screenshot of the rendered HTML after the API call is made here.

🔑 9. Open the api3.html file in a text editor. This file makes a call to the /customers/bycountry endpoint using Spain as the value in the country query parameter.

a. Give the line of code from the getCustomersFromAPI function in the api3.html file that makes the API request for the customers in Spain.

b. Change the line you found in part a to get all of the customers in the USA.

i. Give the line of code that you modified here.

ii. Give a screen shot of the rendered page showing all of the customers in the USA.

🏆 10. Clearly it isn’t practical to edit the code in api3.html each time you want to get the customers in a different country. A much better solution would be to have a text field where the user could enter the country in which to search for customers. This question has you make that improvement. Modify the api3.html file as described below:

a. Add an input element of type text to the page where the user can enter the country in which they want to search for clients. For example:

Rectangle

Description automatically generated with medium confidence

Be sure that your input element includes and id attribute so that you can work with it in your JavaScript later. Give just the HTML code that you added to your page to add this text field.

b. Add code to the getCustomersFromAPI function in the api3.html file so that the API request that is sent to the /customers/bycountry endpoint includes the country that has been entered in the text field. Hint: Get the value of the text field and use string concatenation to build the URL with a query parameter that includes it as the value of country. Reviewing #9a and #9b may also be helpful.

c. Test your page by entering Italy into the text field. Your result should include three customers. Paste the rendered api3.html page with these results as your answer here.

🏆 11. Optional: Imagine that we would now like to display not just the names of each customer in a country, but also their city. To make this update will require changes to both the back-end and the front end.

a. The back-end will need to include the name of the city in the JSON response that is returned. To do this, the SQL query will need to be modified to include the city. Modify the code in server.js so that the /customers/bycountry endpoint finds both the CompanyName and the City in for each customer in the country specified by the query parameter. Give the modified lines from server.js that build the SQL statement as your answer to this question.

b. Now that the result returned by the back-end contains the city data, it can also be rendered by the front end in api3.htm. Modify the getCustomersFromAPI function in api3.html so that it will include the city in the list item (li) that it adds for each customer. Give the complete getCustomersFromAPI function with your modifications here.

🏆🏆 12. Optional: It is pretty easy to imagine that it would be useful to know all of the orders that a particular customer has placed. In this question you will build that endpoint and a page that provides the results.

a. Add a function to the server.js file for an endpoint /orders/bycustomer that accepts a query parameter named company, for which the value will be the name of a company (e.g. company=Seven Seas Imports). This endpoint will use the value of the company query parameter to construct and SQL statement and ultimately return a JSON object with all of the orders that were placed by that company. Give the code for the function that you added to the server.js file. Hint: This requires a JOIN operation in the SQL statement.

b. An API endpoint is not of much use without a front-end page that calls it. Create a front-end page that allows the user to enter the name of a company in a text field and then displays a list of all of the orders that the customer has placed. The information about an order should include the ID of the order and the date on which the order was placed. Give the full HTML source code for your page. (If you want an extra challenge include the name of the employee who processed the order and the name of the company who shipped it and display it all in an HTML table!)

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.