**Modern Asynchronous Programming: Assignment**

**Instructions**

To complete this assignment, you will be building two functions that together will allow for "bulk updating" of an API endpoint in the Constellations Server.

This assignment will make use of the Constellations Server you've used in previous practice exercises as well as the axios library.

See below for more information on the project setup as well as the requirements for each function.

**Starter code**

To start, make sure you've forked and cloned the constellations server, linked below. Follow the instructions in that repository to get the project up and running.

* [GitHub: Constellations Server](https://github.com/Thinkful-Ed/starter-constellations-server)

**Project Setup**

The src/main.js file includes an axios variable and a BASE\_URL variable which points to the correct URL for the constellations server. Please do not change either of these lines.

It also includes a isValid() function which can be used to determine whether or not a constellation is valid. As you can see, this check is relatively primitive and only checks for truthy values for specific properties.

All of the functions you will need to fill in are included in this file.

**Functions**

**update()**

| **Arguments** | **Return value** |
| --- | --- |
| A valid constellation. | A promise. |

This function should take in a constellation object that includes an id. It should then return a PUT request with axios to update that resource with the new contents.

const constellation = {

id: "HwLvy2S",

name: "Ursa Minor",

meaning: "Little Bear",

starsWithPlanets: 6,

quadrant: "NQ3",

};

*//> Makes a PUT request to http://localhost:5000/constellations/HwLvy2S*

await update(constellation); *//> { data: { id: "HwLvy2S", ... }, ... }*

If that request fails, it should return a rejected promise with the following result. Assume id is the value of the constellation's id property.

{

error: `Updating constellation (id: ${id}) failed.`;

}

**bulkImport()**

| **Arguments** | **Return value** |
| --- | --- |
| An array of constellations. | A promise. |

This is the main function you will be working on.

First, this function should take an array of objects. If an array is *not provided*, the function should return a rejected promise with an object. That object should have an error key that includes a relevant message.

bulkImport({}).catch((error) => {

console.log(error);

*//> { error: "Inputted argument must be an array." }*

});

**Note:** You can use Array.isArray() to determine whether or not the inputted argument is an array.

Before making the PUT request, you must first validate that *all of the constellations* are valid, according to the isValid() function. If any constellation is not, the function should return a rejected promise with an object and an error message, like above.

const constellations = [{ starsWithPlanets: 4 }];

bulkImport(constellations).catch((error) => {

console.log(error);

*//> { error: "All constellations must include relevant fields." }*

});

If all of the constellations are valid, the function will create multiple requests for each of the given constellations and store those requests inside of an array. Use the update() function to create each of the PUT requests.

Finally, use Promise.allSettled() to return a promise that evaluates all of the requests.

**Tips**

* You may complete this challenge on your own machine before uploading it to Qualified.
* Reference the related checkpoint for help on completing this Practice problem.
* If you need help, contact your mentor or speak with your peers in Slack.