**Part 2: Technical Specification**

**The implementation you’ve done here is an MVP. While testing your application, you find that the external system where your requests are sent is very unstable! We are required to guarantee that the contents of the external system’s database match the data files that your application ingests, thus your application needs to account for intermittent request failures. Write a brief, maximum one page document that outlines your approach to meeting this requirement.**

So I think that the best way to catch errors on any HTTP requests would be to use some sort of promise-based HTTP request package. My favorite is Axios, but I believe something like a fetch() would suffice as well. What promise-based means is that it your Axios instance will attempt to post, get, update, delete, etc to the base URL you have defined in your request. That instance will return a promise, which will ‘resolve’ or ‘reject’. If there are any errors with the request, they will be caught and passed into the reject function native to promises. You can then use ‘.catch’ and pass in an error handling function. On the other hand, if the request succeeds, then the response will be passed into a ‘resolve’ function, and you can then view that with the .then() method, passing in your resolved function to handle that response. These functions will allow you to customize what you want to see as a response from either a successful or failed http request.

Upon failure, you will get a header/status code similar to when one succeeds. Depending on what that header is, it can tell you a lot about *why* the request failed. Various failures might include that the username and password being used to access the external system (API) are not valid. Another might be not receiving a response or connection from the external system at all. Though there may be limited resources to use as a developer to fix a problem such as the former, this is all important information. If this is an automated process, a good way to document this would be to create a log or text file to document failures (or even all responses) to requests to this external system with a date and timestamp. This would allow a developer to see *when* requests started failing, and for how long. Not only does this help in the troubleshooting process, but it can verify which data successfully was posted, and which data still needs to be added to the system once the system is indeed back up.

A step further on this project (really, the next logical step) to go might be to validate the data being posted, based on the parameters and types that we received in the CSV schemas here. This would help to prevent errors in attempting to post non-sanitized data to a database (say, attempting to post a string as the minimum score here, or a number for the required Boolean value).