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.

Step 1
Whenever we receive an error response from an API call, identify if the fault is a transient fault. A transient fault would be something temporary and recoverable; i.e., not a 4xx-level error such as a badly formatted request or an authentication issue. Transient faults would include things like a 5xx-level response or a request timeout (e.g., due to high volumes of traffic at the API).

Step 2
Implement a retry loop to retry the request, using exponential backoff to increase the time between retries. The maximum number of retry attempts should be pulled from an application configuration parameter.

Step 3
Implement a circuit breaker by keeping track of the total number of times we fail to complete a request because we've run out of retry attempts. Define an application configuration parameter for the maximum number of times this can happen. When we reach that number, open the circuit breaker by stopping all further requests for some configurable length of time.

When that time limit has been reached, allow the next request attempt to be made. If it fails again (after the allowable number of retries), open the circuit breaker and wait again.

During periods when the circuit breaker is open, the application should stop processing input files.

Step 4
We need a way to retry requests that fail because we reached the maximum number of retry attempts. When we are unable to successfully complete a request, write that data record to a new file. All such files can then be re-processed when the API becomes responsive again.