This assessment aims to test your understanding and problem-solving approach when asked to implement code that performs calculations specifically related to the maritime industry.

You need to create a software service that does the following:

- **Reads** input data from a **CSV file**. The information contained in the file concerns metrics that were collected by two different vessels over a specific period of time.
- **Processes** the **metrics** found in the file (more on that below).
- **Provides** a Rest **API** interface so that a user can request more information and statistics related to the data found in the original file, by calling a list of endpoints that need to be implemented.

The metrics found in the input file must be processed in the following ways:

- **Invalid data filtering**: The basic rules to identify whether a metric's value is considered invalid are if it is below zero, if it is completely missing, or if it can be considered an outlier. You can also add other checks if you want.
- **New metric calculation**: Simple calculations can be used for creating new metrics based on the provided ones.
- **Statistics calculation**: Both raw and calculated metrics can be utilized in the code, to gain insight on each vessel's performance.
- **Data merging**: Both raw and calculated metrics must be available to the user, for each waypoint.

The Rest API contained in the final deliverable, must be able to provide the user with the following information:

1. **[*New metric calculation*]** For the **vessel** requested by the user, how much was the **speed difference** between the vessel's actual speed over ground and the speed over ground that was proposed by the system, at each waypoint (a waypoint being defined as a specific pair of coordinates).
2. **[*Invalid data filtering*]** For the **vessel** requested by the user, what **problems** were identified in the original data (based on the previously mentioned filtering), sorted by frequency of occurrence, descending.
3. **[*Statistics calculation*]** Which of the two vessels was **more compliant** with the system's suggestions. The compliance percentage per waypoint can be calculated based on how far from the proposed speed the vessel's actual speed was.
4. **[*Data merging*]** For a specified period of **time**, and for a specific **vessel**, retrieve all values for both raw and calculated metrics.
5. **[*Statistics calculation - Bonus*]** For a specific **vessel**, identify groups of consecutive waypoints with problematic data and sort those groups by the number of problems found, descending. The user must be able to indicate a specific type of problem for which to retrieve information (e.g. outliers, missing values), if they want.

Keep the following things in mind while implementing your solution:

- You must include error handling in your code, to account for edge cases.
- You must ensure that all errors are properly logged.
- You must use Java 17/21 and SpringBoot 3.x, as well as any other library that facilitates an efficient implementation (up to 6 external libraries total).
- Directly retrieving the information from the CSV file, using an in-memory cache or storing it in a database is up to you to decide.

While the following are not hard requirements for this exercise, we will appreciate:

- Usage of object-oriented code and constructs.
- Taking into account concerns such as performance and maintainability. This also applies to SQL, if used.

You should deliver the following:

1. The **code** itself, in the form of a **private** GitHub/BitBucket/GitLab repository, to which we must be provided with access.
2. **Documentation** that describes how to setup and use the tool, what the dependencies are, and any assumptions taken into account during implementation.
3. **Documentation** in the code itself, with javadocs and comments where necessary.
4. **Unit tests** that are descriptive of how your code should work.