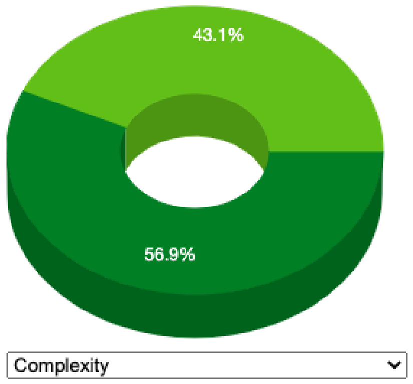
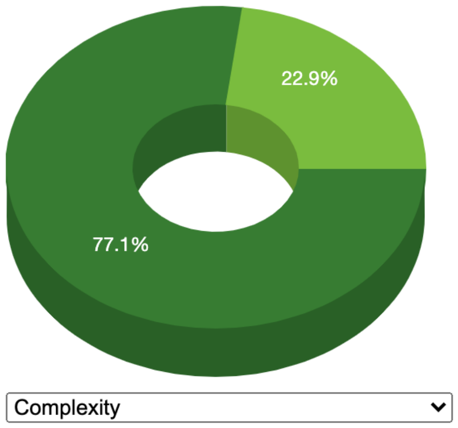
CodeMR

Below is the CodeMR report for the project.



Sprint 2 Report



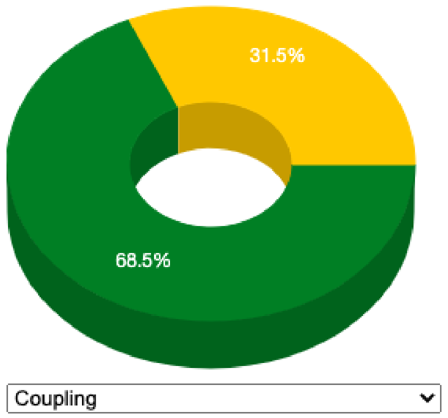
Sprint 3 Report

Complexity

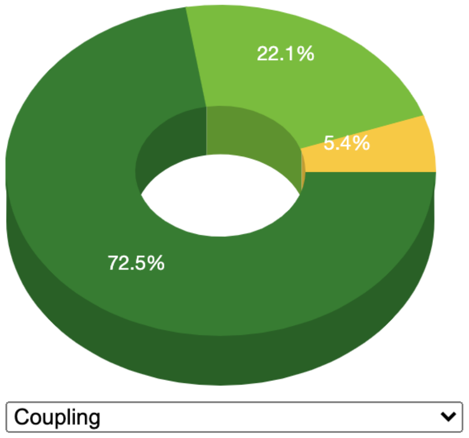
Sprint 2:
Complexity is a measurement of the relationships between the classes. The more difficult it is to understand the design of the code, the harder it is to test or maintain the code. The depiction of the Complexity graph does not present an area of concern.

Sprint 3:
The complexity decreased significantly. Refactoring was done on the JsonFileParser and SummaryEntry.

To decrease the complexity, separate processor classes were created. For example, an ActivityProcessor was created to parse out json objects that were an activity such as cycling or walking.



Sprint 2 Report



Sprint 3 Report

Coupling

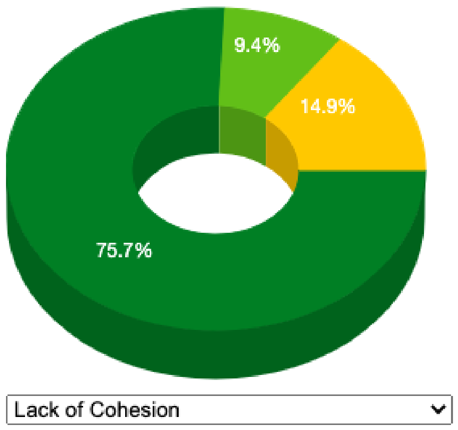
Coupling is a measurement of the dependency between classes such as a class that refers to a method in another class. The coupling graph depicts some areas of concern and high levels of coupling reduces reusability and maintainability of the code. This problem can be contained by adding more interfaces to a collection of similar classes to follow more closely the “program to an interface not an implementation” principle.

<u>Sprint 2:</u>
The JsonFileParser instantiates objects to create entries and add other objects. We suspect because the List for IActivity and ISegment as well as the added private helper methods might be causing this coupling. We anticipate the coupling to increase when we start to parse out additional attributes.

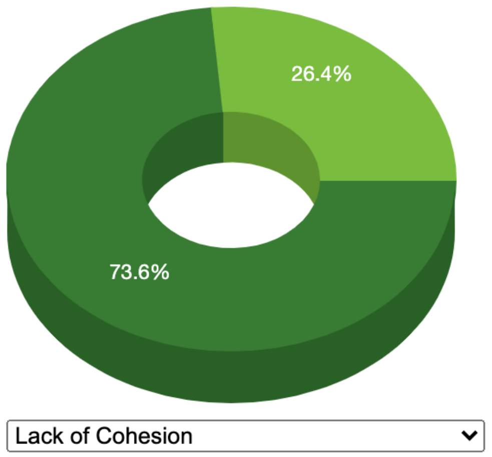
<u>Sprint 3:</u>
For the refactoring, to decrease the coupling in JsonFileParser, we revisited the necessity to have additional classes to represent an activity. We came to the conclusion it was not necessary because the activity was a field, where the type of activity was designated.

Moreover, to decrease the coupling, we utilized a little of the Builder Pattern and FactoryPattern. The idea of the Builder Pattern was used in that when the json object was parsed, it would build the Activity model or the Place model.

Lastly, the idea of FactoryPattern was used in that we created processor classes where Segment, Place, Location, and Activity were built or created, added to a collection, and then inserted into the database.



Sprint 2 Report



Sprint 3 Report

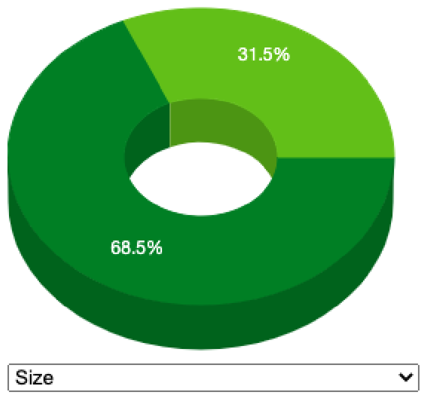
Lack of Cohesion

<u>Sprint 2:</u>
Lack of cohesion is a measurement of how well the roles and responsibilities are defined for classes such that a healthy cohesive class performs 1 function. The lack of cohesion graph depicts an area of concern which can result in confusion of where in code to implement or correct certain features or capabilities. This problem can be reduced by splitting some of the classes into smaller classes.

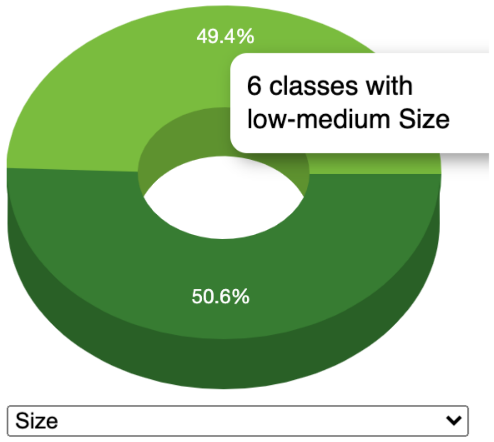
<u>Sprint 3:</u>
We saw a decrease in the lack of cohesion, which was not expected, but a pleasant result of refactoring.

Per the professor, lack of cohesion is of no concern.

Class showing lack of cohesion:
Activity, Entry



Sprint 2 Report



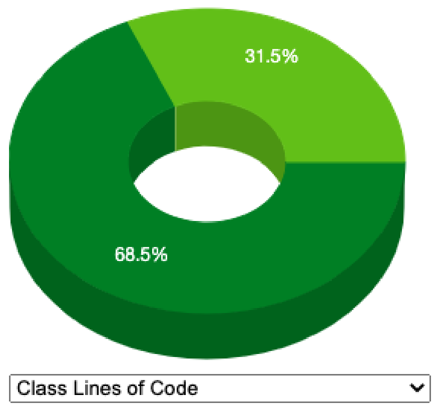
Sprint 3 Report

Size

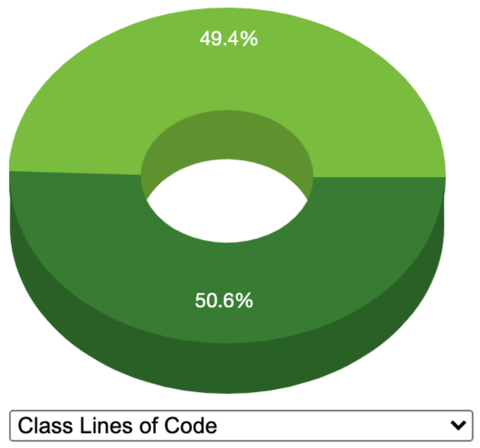
<u>Sprint 2:</u>
Size is a measurement that depends on the amount of methods or lines in the code. The current size graph depicts that the current length of the classes is reasonable and not a concern.

Class showing low-medium size:
JsonFileParser

<u>Sprint 3:</u>
It was expected that the size of the project would grow as we added more implementations, especially RESTful api and aggregations.



Sprint 2 Report

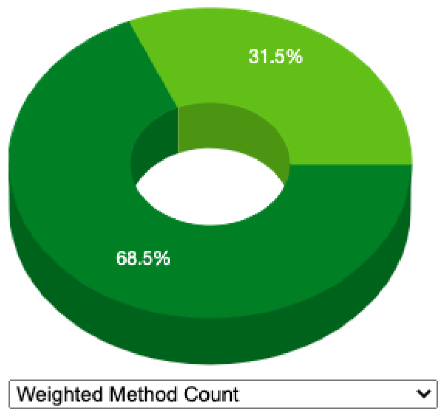
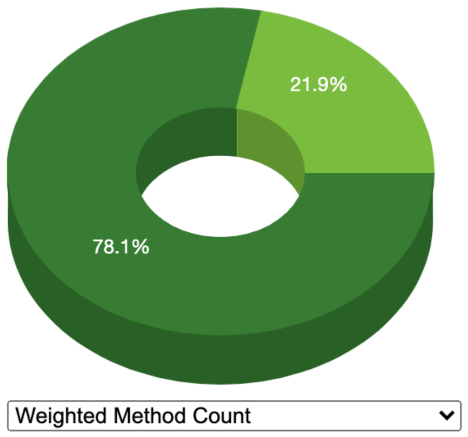


Sprint 3 Report

Class Lines of Code

<u>Spring 2:</u>
Clean lines of code is similar to the size measurement except it does not include commented and empty lines of code in its measurement. The class lines of the code graph does not depict areas of concern.

<u>Sprint 3:</u>
The increase in the class lines of code was to be expected as we started to merge everyone’s code. This is similar to the size.

 Sprint 2 Report  Sprint 3 Report	**Weighted Method Count** <u>Sprint 2:</u> The weighted method count is a measurement similar to the size and complexity metrics. It is the number of methods and the complexity of the code. The weighted method count graph does not currently show an area of concern. <u>Sprint 3:</u> Surprisingly we saw a decrease in the weighted method count. Weighted method count reveals the complexity of the code. It appears that creating classes with that were smaller and specific this lowered the complexity.