**ScaleCheck**

Scalecheck is a tool, which helps in testing the APIs at scale and particularly the performance of the Restful APIs under tremendous loads, and also helps in benchmarking the performance of the APIs.

The other use of the tool is for testing the code and performance of the code after each step, which helps in finding the bottleneck in the code, or any new code which is been added in REST API can be analysed, how much impact the new code added to the existing time taken by the REST API.

So, the application helps developer to test the APIs performance at scale by using varying stress testing measures and loads to check whether their APIs pass SCALE CHECK or not.

# Functional Requirements:

1. When the application is launched, the user should be able to create multiple projects to organize the API testing workspaces.
2. In the projects, the user should be able to create multiple API tests based related to a single project.
3. User should be able to test, the API normal GET, POST, PUT, DELETE along with passing headers, body and security tokens.
4. User should be able to test the performance of the API, by generating a suitable type of load from distributed machines, to stress test the performance of the API.
5. User should be able to generate the PDF/XLS report for the performance test, where they will be able to find all the metrics, how the API performed under which type of load and everything.
6. User should be able to select what type of load they want to generate, for how much time and how much requests per second, and based on variety of load generating strategy, the API test results will be made available.
7. There will be a separate library for Java Spring boot applications, which generates logs for the API under test, and those specialized logs when passed as inputs to the application, can give more detailed report of the code lines, and where the bottleneck is present.
8. This application can also monitor the JVM memory during the load test for the application and generate a report for that also showing how much heap memory got used and is there anything which can be done.

# Non-Functional Requirements:

1. The application should be able to connect and ssh into another distributed systems, and should be able to generate the load from that connected machine also.
2. There is no need for failure control, as the mechanism is not that critical for the system. Its just that the multiple connected nodes can be used to generate a distributed traffic to the API with proper inputs.