JumpCloud Interview Assignment

Prerequisites

- You will need a computer that meets the basic system requirements for the language/framework of your choice
- Basic Git knowledge
- A GitHub.com account

Evaluation Criteria

The purpose of this assignment is to give us insight into your aptitude in some of the areas where a JumpCloud engineer works on a daily basis. Additionally, if you move forward in the interview process this submission will be used as a building block for us to understand how you think about more complex concepts.

Your submission will be evaluated on the following criteria:

- *Does the submission meet the requirements?*
- *How well is the submission documented?* While great code can sometimes be self documenting, we expect code to have documentation including how to use it, intended uses, tricky spots, future considerations, etc.
- *What is the test quality?* We expect all production code to be well-tested. How do you test your code?
- *Code organization*- Is the code cleanly organized?

Requirements

This assignment may be completed in Java, Go, NodeJS or Python. Be sure to provide clear instructions on how to build and test your code. Please don't make any assumptions about the environment that your code will be compiled/run in without explicitly stating those assumptions. Please try to limit the setup complexity by avoiding frameworks or libraries that are far from standard or require any advanced setup-- the simpler the better. If you have questions, please reach out.

To submit your work please push to a public GitHub project and be sure to document any configuration or run instructions. We're looking for a solution to the problem as well as attention to detail and code craftsmanship. Good luck and have fun!

The assignment is to write a small library class that can perform the following operations:

1. Add Action

```
addAction (string) returning error
```

This function accepts a json serialized string of the form below and maintains an average time for each action. 3 sample inputs:

```
1) {"action":"jump", "time":100}
2) {"action":"run", "time":75}
3) {"action":"jump", "time":200}
```

Assume that an end user will be making concurrent calls into this function.

2. Statistics

```
getStats () returning string
```

Write a second function that accepts no input and returns a serialized json array of the average time for each action that has been provided to the addAction function. Output after the 3 sample calls above would be:

```
[
    {"action":"jump", "avg":150},
    {"action":"run", "avg":75}
]
```

Assume that an end user will be making concurrent calls into all functions.