Carsten Hood
UIN: 922009787
CSCE 489-599 Cloud Computing
Early deadline: 2017/9/25

Task 1: P1-T1 – Fitness data on MongoDB

1. *The time you took to complete this task.*

6 hours (including familiarization, browsing online, watching videos, answering questions etc)

2. P1-T1-A: *The features of MongoDB that make it suitable for the application.*

MongoDB's features will allow for a quicker development process and a more robust application. MongoDB's document model is a more intuitive and flexible means of containing data such as the application's user fitness information. For example, data can be structured in familiar JSON files rather than in unwieldy relational structures. Also, since users will provide disparate fitness data, we can take advantage of MongoDB's flexible data model. MongoDB's expressive query language then allows for efficient access of this data. Furthermore, the quality of strong consistency will facilitate application development, since developers won't have to account for obsolete or unsynchronized data. Another desirable feature of MongoDB is the health of its community and developers; documentation and support is easily accessible and unlikely to diminish.

3. P1-T1-B: *Include any comments you have about improving the design of the database.*

No comments on design improvement. See *queries.txt* for query solutions.

4. P2-T1-C:

i. *Your comments and opinions about what could go wrong with placing all their data in one server.*

The central risk of keeping all data on a single server is the risk of losing data or data availability should the sole server or its network connection be damaged or otherwise fail. The risk of data loss can be partially mitigated through frequent backups, but backups may still be obsolete or incomplete and don't address the risk of losing data access temporarily. With a single server even basic maintenance and development tasks such as software updates may require temporarily taking the application offline. For these reasons it would be advisable to use two or more servers.

ii. *Your comments and opinions on how data should be spread across multiple servers.*

I recommend maintaining multiple duplicate servers such that each server contains copies of the same data. Due to the relatively small amount of data and the small workload involved, this would be relatively simple and inexpensive. The servers do not necessarily have to constitute separate dedicated hardware systems and may be virtual servers operating alongside other processes. One option is to use one or two in-house servers along with a third-party-hosted virtual server (in a separate location). This would provide a level of robustness in case of failure or compromise of one of the servers or one of the servers' network connections. It would also ensure that the application could continue to function normally during local server maintenance.