Assumptions and considerations:

We assume the server is highly available. Therefore, we did not consider using load balancing in the current implementation. We also assume that the database is highly available.

There are several improvements that can be made for future work. For stronger security protection, we can add an authorization layer on top of REST authentication. To further improve performance, we would also like to reduce the number of round-trips to DB by creating additional classes to serialize complicated objects. Additionally, we can load data on-demand.

Our good wish for the project is for high availability and distributed cache as the following:

* + High Availability: introducing redundancies to remove single point of failure. Example: load balancing between machines.
  + Distributed Cache: independent cache server to improve DB performance with concurrent read and concurrent write.