Homework 1: Databases

You are a consultant at a company called Aggies Forever. The company has decided to create a fitness aware environment in the workplace. They have assigned a team to work specifically on their fitness portal called AggieFit. They want to design a system that will track the fitness activities of the employees and reward them for maintaining their health.

To maintain some anonymity, there is a separate portal in which employees create a unique ID for AggieFit. The map between the unique ID for the fitness portal and employee records are hidden from the AggieFit team. The employees can, however, choose to expose their data. The employees can also choose to add their personal details (e.g., height, weight, age etc) to AggieFit. Employees must choose from a list of fitness goals. They will then report their activities by integrating their fitness devices.

Your task is to assist the AggieFit team in designing their system by providing prototypes and comments. As they design their system, they ask you for help in two aspects of their system as follows:

Task 1: Fitness data on MongoDB

Due: 3/2/2018, 11:59pm CDT

Early bird deadline: 2/23/2018, 11:59pm CDT

1. Provided information

As their resident database expert, they want you to design a database for AggieFit. They have some dummy fitness data for you to play around with. Each employee uses a different device to track their activities, and some employees use multiple devices. The employees may have shared their personal information with AggieFit. Some employees may also have decided to make their employment information available to AggieFit.

2. Installation

They want you to use MongoDB, a document-oriented database, to store the fitness data. First, they want you to install a MongoDB client in your machine and verify that you can connect to their database.

Download and install MongoDB:

1. Download MongoDB from: https://www.mongodb.com/download-center#community
2. Install MongoDB: https://docs.mongodb.com/manual/administration/install-community/
3. Connect the MongoDB client on 34.233.78.56 and use the database `fitness_X`, by running `use fitness_X`, where `X` your UIN. E.g., if your UIN is 111001111, you will

execute `use fitness_111001111`. Verify that you are connected by running the command: `db.fitness_X.find().pretty()`. The query should return some values if you are connected to the database.

4. AggieFit team suggests you to use python to connect to MongoDB and provide you with some helper scripts under MongoDB-Pythons-scripts. You are free to use other tools.
   To use python,
   a. Download and install python 2.7.13 on: https://www.python.org/downloads/
   b. Run python and install pymongo: http://api.mongodb.com/python/current/installation.html
   c. Under eCampus -> Content -> Projects and Homeworks -> HW1, you will find python scripts to connect to MongoDB under task1.zip. A function to connect to a remote MongoDB database is provided in `mongoConnect.py`. You will need to modify `constants.py`. Update `db` and `collection` variables to `fitness_X`, where `X` is your UIN. Run `queries.py` to verify that your installation is completed. The script should print some values if you are connected to the database.

Tip: Watch an example of how to complete steps 3 and 4 (c) in the example video posted on eCampus -> Projects and Homeworks -> HW1. Note that the example is on a different server and database.

3. The tasks

Task 1a: The AggieFit team tells you that their senior management wanted to use a relational database for this application, but they think MongoDB fits better. Now, they need to convince the senior management that their choice is indeed appropriate. They ask you to provide a short write-up to validate their choice of database, specifically, what features of MongoDB makes it more suitable for their application.

Task 1b: AggieFit team has dumped some dummy data into a collection in a MongoDB database. Additionally, they have provided you with more fitness data under eCampus -> Projects and Homeworks -> HW1 under task1.zip. The data is in a file "`dummy-fitness.json`". It contains the fitness data in JSON format. Updates to employees' personal data and new tags for users come in a JSON formatted file. They provide a sample update file for a dummy employee in `user1001-new.json`.

They want you to run queries ordered as follows:

WQ1. Add the data in `dummy-fitness.json` to the MongoDB database

WQ2. Update the database with data from `user1001-new.csv`.

RQ1. Count the number of employees whose data is in the AggieFit database.

RQ2. Retrieve employees who have been tagged as "`active`".

RQ3. Retrieve employees that have a goal step count greater than 5000 steps.

RQ4. Aggregate the total activity duration for each employee. If the employee does not have activity duration in their data, you can report their total activity duration as 0.

Note: If any of the information provided is unclear, please ask for clarification on Piazza.

4. What the AggieFit team wants you to do

You should do the following to complete this task.

Do the following on the database:

Task1b: Execute the 6 queries on your database. You will need to execute the queries in the order specified by your client.

You need to **prepare a report** that contains the following:

1. The time you took to complete this task
2. Task1a: A section on the features of MongoDB that make it suitable for the application. If you disagree with your client, you will have to provide a thorough explanation.
3. Task1b: Include any comments you have about improving the design of the database.

You may use the python interface to connect to MongoDB. In `queries.py`, you will find comments that explain the syntax to perform the basic operations required in this assignment. You can use them to send the commands to the database.

Task 2: Message boards

Due: 3/2/2018, 11:59pm CDT

Early bird deadline: 2/23/2018, 11:59pm CDT

1. Provided information

The AggieFit team wants to add message boards to encourage the employees to interact and exchange information with each other. They intend to categorize the message boards by topics. The employees can view and add messages to boards that they are interested in.

The team took recommendation from experts and were told that they could use two databases based on their requirements: MongoDB and Redis. The experts were not told about the application. The AggieFit team wants to know where the databases fit in their message board application. The basic functionality they desire is as follows:

(i) Select message board: Choose a message board to read and write from. All operations are performed only after the message board is selected.
(ii) Read: Should be able to read all messages on the selected message board.
(iii) Write: Write the message into the selected message board.
(iv) Listen to updates: Waits to get real time updates on the selected message board. If any other user writes on this message board, you should be able to read it immediately. Unlike read, the user does not have to explicitly pull the data. If the user is listening on a board, they should receive updates immediately after the board is updated.

2. The task

The team asks you to propose a system architecture (similar to DropCam and Slack as you saw in the workloads video, but focusing on the databases). They also want you to implement a prototype for the message board application using your proposed architecture.

First, you will need to go through the features of both databases and identify which one is suitable for what part and propose where the databases fit in the architecture. The architecture should clarify what happens when a user selects, reads from, writes to, and listens to the message boards.

Next, you will need to implement a simple prototype that demonstrates basic functionality of the system. For now, a simple command line output is sufficient. Test scenarios will involve multiple users accessing the boards in a particular order. The following operations should be supported:

(i) Select message board: select <board_name>
(ii) Read: read. If no board is selected, output an error that no board was selected.
(iii) Write: write <message>. Output and error if no board is selected.
(iv) Listen to updates: listen. Output and error if no board is selected.
(v) Stop listening to updates: stop or <Ctrl-C>. Output and error if no board is selected and not listening.

Two sample scenarios are attached on eCampus->Project and Homeworks -> HW1.

For the basic case, you can assume that the different users of the system are time synchronized, and therefore see the same ordering of operations. We suggest you code the prototype without any error checking, i.e., assuming that all commands adhere to the specification. You can add error checking incrementally.

MongoDB download and installation instructions are already provided to you in task 1. If you would like to reuse the MongoDB server from task 1, you should name your databases appended with your UIN. Any databases that do not contain UIN suffixes will be deleted on a daily basis.

For Redis, download and install using the instructions in https://redis.io/download. To interface with the programming language of your choice, use https://redis.io/clients. The desired library for python is redis-py. A sample script for a different application is provided along with the lectures.

3. What your client wants you to do

You should do the following to complete this task.

You need to **prepare a report** that contains the following:

1. The time you took to complete this task
2. The task:
   a. A section on your architecture. You may have decided to use the databases for different purposes. You should explain your choice in the report. Also, you will need to explain what happens when a user performs an operation on the system.
   b. A section on your prototype and instructions on how to run your code.

Project submission and grading rubric

You submit a zip file containing two folders called "Task 1" and "Task 2". Within "Task 1", you should include your report, and a file containing your queries for WQ1, WQ2, RQ1, RQ2, RQ3, and RQ4. If you use a client library (e.g., python), you can submit the source code in a single file (e.g., queries.py) along with instructions on how to run the code. In "Task 2", you should include your report and the code.

The submission "button" is located on eCampus -> Projects and Homeworks -> HW1.

Task 1 is worth 40 points and task 2 is worth 60 points. You are required to use github for your code within a folder called HW1. The principles used for grading will be as follows:

- Failure to use github: -100 points
- Task 1:
    - Task 1a: 10 points
    - Task 1b: 30 points
- Task 2:
    - Missing system architecture: -30 points
    - Missing instructions on how to run your code: -30 points
    - Not explaining your design choices: -30 points
    - Prototype fails: -20 points

If you submit by the early bird deadline, you get 10 points extra.

Partial credit will be applied if the answers are not thorough.