Developer Hiring Test (2019)

Context

We are looking for all rounders (Vue.js, Node.js, MongoDB, C#, Git) that can work independently on vertical tasks (as part of a team).

- **Input**: This document.
- **Output**: A zipped project (or cloud hosted Git project) where the solution is provided (code, any DB init scripts or sample DB zipped).
- **Time**: We expect this task to take around 3 days, but if you work part-time on this you can ask for more time.

We are looking for people that can work in various positions. Even if you think you cannot handle all of the requirements in given time, you can still apply if you can handle some of these.

Sample Problem Space

Our users want to create **Tasks** and combine them in a **Pipeline**. Tasks can be created and combined by different users, but only users that create a Task / Pipeline them can edit / delete it.

A task contains two domain properties a **Name** and the **AverageTime** it takes to run.

Users combine Tasks in a Pipeline which has a user given Name. You can assume that pipeline has no branches (only one branch), and if it helps assume also no repetitions of tasks. An example pipeline is shown next:

T1 -> T2 -> T3 -> T4

[Calculate Average Time] [Run Pipeline]

Once the pipeline is created, users wants to do two things:

1. Calculate the average time it takes, using average time of tasks.
2. Run the pipeline and store its run time.

Task Requirements

Design a simple Vue.js program with a Node.js in JavaScript (ideally using Express) backend that uses MongoDB, which does the following:

1. Allows user to create a Task and save it in MongoDB
    a. No edit is needed for Tasks
    b. Optional, user can delete a Task
2. Allows user to create a Pipeline and save it in MongoDB
    a. We do not need a fancy UI, a list where all tasks are added / removed will do
    b. No need to implement, but explain, how you will implement pagination for Tasks list when they are selected for inclusion in Pipeline
3. Allow user to calculate total average time of a Pipeline using Node.js to read Task.AverageTime data from MongoDB and show the result in UI
    a. Users can recalculate total average time if Pipeline changes
    b. You may choose to trigger re-calculate explicitly, or as pipeline changes
        i. How would handle an automatic update of this time in UI?
4. Allow user to run the pipeline
    a. Running the pipeline should trigger an external program from Node.js.
        i. The time Pipeline run took should be stored in DB (PipelineRunTime)
    b. The external program is written in C#
        i. Simplest implementation, program does nothing (Thread.Sleep(random))
        ii. You may just read the task average time from DB in C# add them as return that results in output, to be printed in UI.
        iii. Use async / await.
    c. Show “Pipeline (Name) finished” in UI, when pipeline is done.

Out of scope:

- You are not required to implement user creation and login. You can assume you have a list of users in DB and select the current acting user from a list in UI.
- No need to use Vuex, but it is nice if you can use Vue Router, thought it is not required to do so.
- No need to write unit tests, but you will be asked how they can be implemented for UI and backends.
- No TypeScript, but if you like TypeScript you may mention pros and cons during the interview.
- If you like, you can use Twitter Bootstrap to style your app, we do not need very fancy CSS. What UI/CSS framework would you suggest?

Bonus Task (Data Processing)

You are not required to implement this task, but you may think about it, or even provide code if you like (especially, if you want to work in our data processing part):

Assume, you have in DB a collection of how long each pipeline run took (**PipelineRunTime**).

We need to know the **median** of all times.

How would you solve this task in these two cases:

1. If it is feasible to fetch all DB collection data in memory in Node.js/C#
2. If not feasible to load all DB collection data in memory

Extra bonus: How would you show a UI graph made of the median of each week? What technologies would you use?

Other Questions (you may be asked)

No need to write code for these:

1. What happens if pipeline has branches
    a. What if tasks are repeated?
2. How would you model Tasks dependencies?
    a. What would be your time forecast for this task?
3. How would you do the unit tests?
    a. Integration tests?
        i. CI & CD
4. If you were to use a message-broker for backend, what would you choose, how it may work?
5. What could be a good usage of Docker in our example task?

EOF