Student Developer Test

This test has the purpose of testing how good you are at a wide array of fields that may be useful working for Corti. There will be 2 tasks:

- Python integration with AWS
- Web development with NodeJS

Things that are important to us

- Clean and orderly code
- Testing
- Documentation

Rules

- You can use whatever you would use doing your possible future job at Corti
- You will have 5 days to finish the test. You decide how much or how little you want to work. You need to send a link to a github repo with your solution no later than the time specified in the e-mail from us.
- You can ask as many questions as you want during your completion of the tasks. Just hit reply on the e-mail from us. However, note that the questions you ask will be a part of our final assessment.

Task nr. 1: Python integration with AWS

It's your job to design a simple messaging system using AWS SQS (https://aws.amazon.com/sqs/).
You need to write a Python3 program that can:

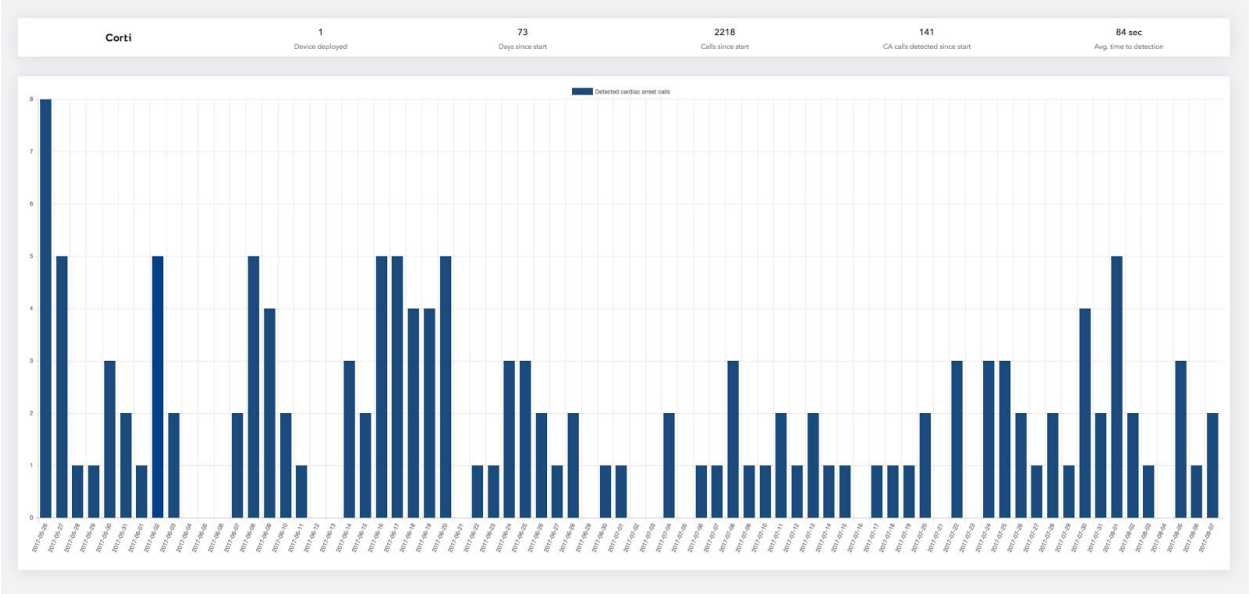
- Read a line from a file
- Write it to the SQS
- Read it back from the SQS
- Write it back to a file

Imagine it as 2 asynchronous processes exchanging information.

At the interview you will be asked to demo your solution on your own computer.

Task nr. 2: Web development with NodeJS

You're tasked with implementing a dashboard that looks like this:

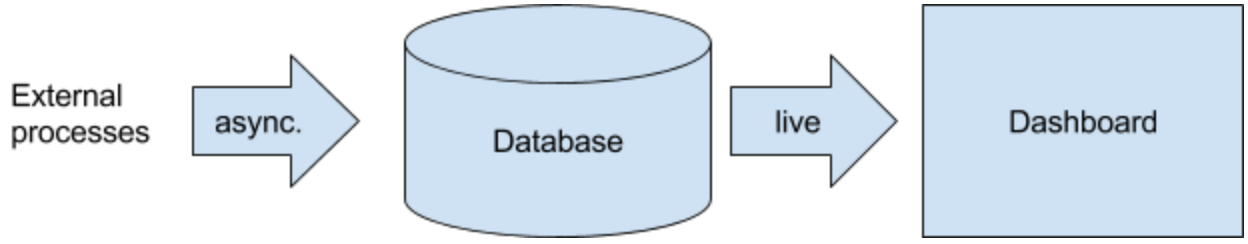


High res:
https://www.dropbox.com/s/xdex30g5idhqbam/Screenshot%202017-09-07%2016.25.17.png?dl=0

The dashboard should show various stats collected from our live devices. The bar chart shows the number of cardiac arrest calls detected on a given day.

You should implement both the backend and the frontend. You have to use NodeJS as backend and VueJS as frontend. (If you're in time trouble it is ok to pick a different frontend framework if you're not familiar with VueJS).

The system you're working with looks like this.



External processes (i.e. the devices), that you don't have access to, asynchronously push data to a database with the following collections and schema:

```
Call:
      deviceId: <unique id>,
      createdAt: <unix timestamp>
      prediction: {
```

```
        predictionTime: <int> (time in seconds)
        ca: <boolean>
    }
```

The dashboard needs to be updated “live” meaning that as fast as possible after the database receives a new Call object should the dashboard be updated.
Here’s how to calculate the individual numbers:

- Device deployed: the `deviceId`’s in a `Call` is unique
- Use createdAt to link all data to a specific time
- Use prediction.ca to tell if a call is a CA (cardiac arrest call)
- Use prediction.predictionTime to calculate avg time to detection

Frontend stuff:

- You have to use VueJS
- Use font: Avenir
- Grab Corti logo from here: https://www.dropbox.com/s/h19r9j6fvvpwp9g/corti_logo.svg?dl=1

Backend stuff

- You have to use NodeJS as backend
- You should create your own dummy data
- You can pick which database you want to ue
- You can use any npm’s you want

You can assume that the dashboard will be used for internal use to begin with so you don’t have to take scale too much into consideration, but you should be prepared to (among many other things) answer questions around how your solution will scale.

At the interview you will be asked to demo your solution on your own computer.