General Overview

You are required to choose one of the two following challenges and complete it. Please, return the results to us ASAP.

Additional Notes for the Candidate

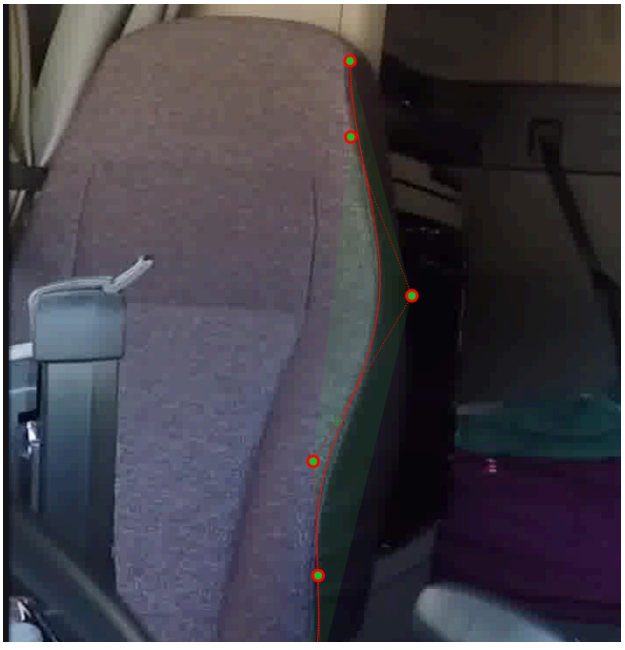
- Document your work and results as well.
- Your work will not be used for commercial purposes outside of the interview process.
- We are interested in the both the thought process and approach.
- Feel free to communicate us if you have any questions.

Challenge 1: Spline Visualizing Service

First of All, you need to understand what is spline. Splines may have different control points and can be indicated using three parameters t, c, k. You can also try drawing splines by signing in dlabel.org and creating an item of spline type. Afterwards, we want you to implement a service to take an image and tck parameters as inputs and visualize the spline on the image just like the following image and return it as response.

Question

- Implement the mentioned service in Sanic or Flask framework.



Challenge 2: User-based request prioritizing

Suppose we have a CPU/GPU bound service or function (limited_f) that takes relatively long (from seconds to potentially minutes) to complete. This service can process one request at a time. We want to develop an intermediary service that takes (user_id, x) as input and prioritizes the requests in order to fairly allocate the limited service to users and call limited_f(x). The problem is that user A might send thousands of requests in a row. Therefore, user B can't access the service until the server finishes processing all user A's requests. Users should be able to submit a job (it should be fast to submit jobs). They should be able to check if their job is finished or not and get their result in case it is.

Question

- Implement the mentioned intermediary service in Sanic or Flask framework**.**

Please note that we are **NOT** interested in the following implementations:

- Processing jobs and queues (or calling limited_f directly) when some user submits a job.
- Using request limiters like Flask-Limiter (this is not the problem we are trying to solve).
- Using task queues like Celery or RQ.
- Please do NOT use docker.
- We do not want you to spend more than a few hours on this challenge. If it seems too complicated, please look at Challenge 2.

We are interested in using asyncio and multiprocessing.

As a bonus point, you could also add the ability to have users with different weights, meaning the ones with higher weights will receive more service than the ones with lower weight.

Another bonus point would be the ability to monitor the queue (how many jobs are in queue), limited service (which user is currently using it, or how fast is it) and removing a user from the queue.