**CAAS Tech Challenge**

Dear Candidate,

The text that follows describes a small tech challenge that we wish you to tackle within the CAAS Software Engineering recruitment process.

The purpose of the challenge is to allow you to show us your hands-on experience. You don’t need to execute the challenge from scratch with witnesses. Instead, you’re expected to spend few hours preparing on your own. Usually, we leave a full week for the candidate to get ready.

Imagine that you’re doing a demo for your fellow squad team members: you will need to be able to clearly explain the code you have written but also the steps that you’re following to create, test and run the code.

This will be done through a walkthrough. The walkthrough should last for half an hour, maximum, which will include some questions. Half an hour goes fast.

We don’t expect to see a finished product: we do expect to see the first iteration into a future, more mature, solution, as well as clear explanations of what has been created, how and why.

After the walkthrough, we may ask some questions. Also use the chance to ask for questions yourself.

We invite you to do the solution presentation face to face at CAAS. This way you will know where we work and we have an opportunity to meet with your future team.

**The Flight Plan project**

**Problem statement:** Flights file a flight plan that specify a route that the pilot wishes to take between its departure and destination airports. The route is made up of waypoints and airways. We want to create a software that interrogates a few APIs, select a flight and display the associated flight route on a global map.

An additional task would be to find and propose alternate flight routes when a button is pressed and to display it accordingly. This additional task is an optional task and candidates need not complete this task.

**1. The required functionality: introducing the Flight Plan, Airways and Waypoints APIs**

CAAS has developed the above APIs that you can query. The API is accessible through these endpoints

* <http://118.189.146.180:9080/flight-manager/displayAll>
* <http://118.189.146.180:9080/geopoints/list/airways>
* <http://118.189.146.180:9080/geopoints/list/fixes>

Your temporary key to access the API isin the email that was sent along with this document.

And here you have a couple of examples of query using curl that you can use to check your key:

curl -v -H 'apikey: xxxxxx' 118.189.146.180:9080/flight-manager/displayAll

Your next steps to familiarise yourself with the API could be the following:

1. Please check that the key is working correctly first as it is periodically rotated. If you have any issue with the key, just reply to this email.
2. The link to the API documentation for the two services can be found at the following link
   * [https://app.swaggerhub.com/apis-docs/CAASFSDATMSE/flight-object-manager/1.0.0](https://app.swaggerhub.com/apis-docs/CAASFSDATMSE/flight-object-manager/1.0.0" \t "_blank)
   * [https://app.swaggerhub.com/apis-docs/CAASFSDATMSE/aeronautical-data-service/1.0.0](https://app.swaggerhub.com/apis-docs/CAASFSDATMSE/aeronautical-data-service/1.0.0" \t "_blank)

It is written in conformance with Open API 3.0.2 standards. Have a read through prior to using the APIs.

1. Figure out what Flight Plans, Airways and Waypoints are:
   * <https://en.wikipedia.org/wiki/Flight>\_plan
   * [https://en.wikipedia.org/wiki/Airway\_(aviation)](https://eur02.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FStandard_terminal_arrival_route&data=04%7C01%7CGabriel.mesquida.e%40thalesdigital.io%7C6c62bdc79fa94b07644308d8fb3cb67d%7C737c6905f1864bcfafb343e349ee23a3%7C0%7C0%7C637535585496659366%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=62tp7x86NETrpWGjEwxNedVE53kRF%2F188GIabsG5d7A%3D&reserved=0)
   * <https://skybrary.aero/articles/waypoint>
2. Look at the capability to get a list of flight plans with a REST client:
   * <http://118.189.146.180:9080/flight-manager/displayAll>
   * Flight plans are represented in a JSON Flight Object Model
3. For the selected flight plan, retrieve and display graphically the flight route in the flight plan using the Airways and waypoint recorded in the flight plan. This list of the Airways and Waypoints can be found in the following APIs respectively in JSON:
   * [*http://118.189.146.180:9080/geopoints/list/airways*](http://118.189.146.180:9080/geopoints/list/airways)
   * [*http://118.189.146.180:9080/geopoints/list/fixes*](http://118.189.146.180:9080/geopoints/list/fixes)
4. Create a simple UI for the flight plan display. Ensure the following functions are possible:
   * Listing of all air-routes
   * Simple search function of flight plan based on Callsign
5. As an additional optional task, candidates may wish to use the airways and waypoints to compute an alternate route from the departure airport to the destination.

**2. What do we expect in a tech challenge?**

1. You will write some code that acquires and exploits some information, as described above in the functionality section.
2. We expect to see an end-to-end *CI/CD* process, from code, to testing and deployment of an iteration of working software.
3. The cycle will start from some repository or source code management tool (*GitHub* & *GitLab* are recommended).
4. There is no restrictions on the programming languages that you use. We do have a preference for Node.JS orTypescript, but you may use any language that you are comfortable with.
5. You should split back end and front end to add value to your challenge.
6. The back end will connect to an API, capture the required information and graphically display the flight path (see following section).
7. The build environment shall be Linux (not *Windows*).
8. The binary artifact will be properly containerised (*Docker* or equivalent).
9. To integrate and deploy, you will use some kind of automation (e.g. *GitHub* actions), kindly show us your *DevOps* capabilities.
10. The deployment will be in a digital infrastructure (*Azure*, *Google*, *AWS* recommended, running locally is a possibility too).
11. There will be a front-end, that will allow the selection of a flight and to display the flight path of the selected flight on screen (no graphical design is required, using a framework is recommended).
12. A *README.md* file explaining the main points of what has been built, maybe with a component diagram, to support the walkthrough.

**3. The walkthrough:**

You only have half an hour to present your work and demonstrate that it works.

Imagine that you are presenting this to your fellow squad members: clear explanations make a difference. Don’t get lost going into every detail: the walkthrough must have proper flow.

We suggest that in the repository, you create a *README.me* file with a clear diagram of the components of your solution and also with the steps that are required to build it. (No need for a Power Point.) You can use the *README.md* to assist you on your presentation. You are not expected to code live or to memorise the commands you are going to use.

For the walkthrough session, we recommend you spend some minutes to walk us through the code you have written. We would suggest topics such as explaining its structure and the key concepts used. We also would like you to show us the code that supports the building, test, and deployment.

Finally, demonstrate the interaction with front-end of the solution. Select a flight and show the flight path graphically and the other functionality as stated above. If implemented show also the alternate flight path. Show us that it works!

Other topics and ideas that you might like allocate some minutes to discuss:

* Your feedback about the tech challenge and what you have learnt from it (please do at least one element which is new to you, and explain it to us)
* How would you work differently if this was not a test but a two-week sprint.
* Your own suggestions on how to improve the code for production quality (how to increase maturity)

We are also happy to provide additional clarifications if required. Just reply to this email with your questions. And if you want to propose a different challenge based on an exciting topic (ML?) we are open to your suggestions.

Our suggested timeframe is to do the presentation one week after receiving this email. When you’re ready, please confirm your availability to David Leow and we will set a slot for the walkthrough.

We’re looking forward to reviewing the outcome of your tech challenge.

Thank you very much, and best of luck.

***Options for the CI/CD process (choose what suits you best):***

We think that you should package your code in a container and push it to a repo as an intermediate step before running it. This cannot be done manually but instead by using a script/ action to build and deploy.

Here you have some options for you to consider for your deployment:

* Using *GitHub* actions for the script or *yaml* scripts in *Gitlab*
* Running your code as container instance in a cloud supplier (*Cloud run, Fargate, ACI*)
* Running on *Kubernetes* (for instance *AKS, GKE* or *EKS*)
* Run it in your laptop using *minikube* or a locally installed *Kubernetes* cluster (if you have a powerful laptop)

Additional CI/CD to support extra functionality (not required)

* Use a *helm chart* to deploy the *Kafka* broker / *zookeeper* and/or your code once packaged