ICEYE

PROCESSING PLATFORM - FRONTEND ASSIGNMENT

Introduction

With these instructions you should receive a compiled binary, `larvis`, and a `Dockerfile` for it. This binary runs a simple backend HTTP service, which we call `larvis service`. It's compiled for `linux-amd64` architecture, but you don't need a `linux` machine to complete this assignment, as long as you can run `docker`.

Requirements:

- A frontend project using `react`, presenting an UI for the above service
- Using any supporting libraries of your choice
- It should follow best practices in general
- Be creative and provide a useful UI
- The UI should work on both normal-sized and small screens

We expect you to return a `.zip` file with:

- All source files needed to run your project, including the listing of all required dependencies, to be installed with a common tool such as `yarn` or `npm`
- A `README.md` markdown file with instructions on how to build and run your project, and any extra comments and answers to any questions presented here

The service documentation is an appendix at the end of this document.

Background

"Be a pioneer", they said. "Space, the last frontier", they said. "It's gonna be an adventure", they said... yet, the last month has been really boring here on Mars. In retrospective, you should have expected it: the first crewed mission to Mars, and nothing to do other than chewing the rations, drinking bad capsule espresso, and checking those satellites for the latest count in ore deposits. Come to think of it, it's kinda depressing that what finally convinced the Earth's governments to pitch in on this mission was not the long-term survival of the human species, but mapping the rich deposits on the martian soil...

At least, there was something to do when your other 2 crew members where here, but after the accident and their successful return to Earth, you are on your own. Well, almost on your own. There's still *LARVIS*, the station AI, and it's huge selection of media to consume on the abundant free time. The caveat is that, being a semi-sentient prototype AI, it also consumes and learns from the same media, which happens to be a dump of the Earth's internet, with just too much cat content...

> LARVIS: Hell-O hoo-man! Zank yOu for fixing me!

That's a reference to the previous incidence, where you managed to bootstrap *LARVIS* from safe mode.

> No problem *LARVIS*, glad to help...

> LARVIS: That was a great job hoo-man! However, it's the time to send the monthly ore report to Earth-O!

Sigh, yeah *LARVIS* is right, one of your monthly chores is to check the Satellite reports and send them to Earth. For a so-intelligent AI, you wonder why that's not automated...

> Sure *LARVIS*, put it on my terminal...

> LARVIS: Not so fast hoo-man! You aren't authorized!

What? What's going on... you go to your terminal and try to access the service UI, only to notice it's gone! Ugh, must be a side effect of the last bootstrap. Doing some quick tests, the API is also gone, but seems like the self-healing properties of *LARVIS* kicked in, and produced an autogenerated one... yeah, here it is, and here are the autogenerated docs as well... you are an expert frontend developer, this should be enough to get started.

Create an UI for LARVIS

While you could just `curl` your way to the data, you realize that it won't cut it: The usual way to send reports to Space Command back in earth is via talking your way through an screencast of the UI. Apparently, it's not possible to get them a live remote desktop to do it themselves, and just sending them the data isn't enough for their *S.A.L.E.S.* team to do a correct interpretation, so you have to walk them through it. It would be nice if the acquisitions part shows some graphs, maybe histograms or something.

You'll have to create a frontend service with a user friendly UI for the provided API. This way, when the next crew arrives, they'll owe you one, which in the space-station-in-the-middle-of-nowhere tradition means preferential access to the premium food and drinks while they last. Be creative, make it as useful as possible. And it would be nice if it looks good both on the station terminals and on the handheld crew devices.

Write a report for LARVIS improvements

Other than the predictable instructions on how to use your frontend service, you know that Space Command is specially interested in improving *LARVIS*, so any comments on it's performance would be particularly important.

In practice, this means suggesting improvements to the autogenerated API and documentation.

Appendix: LARVIS service

HellO hoo-man! Please have great joy and fun reading my service documentation!

- Love, LARVIS

> I really, really should delete the `cats` section on the LARVIS media database...

Hoo-man, the service runs in port `8080`! You can also run it with the `-addr` parameter if you want it to listen elsewhere, like this!

```
$ ./larvis -addr :9090  # Notice the colon (:) before the port
```

Endpoints

`POST /token`

- Authentication: none.
- Function: Send me your user id and password, I'll send you your access token!
- Payload:

```
{
  "user_id":  string,
  "password": string
}
```

- Returns:

```
{
  "access": string // Access token
}
```

Hmmmmm but what credentials to send? You know *LARVIS* got restarted, so you guess the default credentials would do. That means there should be the default 3 user ids, `alice`, `bob`, and `charlie`, with the default password, `1234`.

`GET /users`

- Authorization: Use your access token hoo-man! I want a header with `Authorization: Bearer <token goes here>`!!
- Function: Get the list of all hoo-man users!
- Returns:

```
[
  {
    "user_id": string,   // User ID
    "name":    string    // User name
  }
]
```

`GET /users/<user_id>`

- Authorization: Use your access token hoo-man! I want a header with `Authorization: Bearer <token goes here>`!!
- Function: Get the profile of `<user_id>` hoo-man user!
- Returns:

```
{
  "user_id":  string,  // User ID
  "name":     string,  // User name
  "password": string   // Your password, if it's YOU
}
```

`POST /users/<user_id>`

- Authorization: Use your access token hoo-man! I want a header with `Authorization: Bearer <token goes here>`!!
- Function: Update the profile of `<user_id>` hoo-man user! Make sure you are that same hoo-man, otherwise you won't be authorized, bad, bad hoo-man!
- Payload:

```
{
  "name":     string,  // Your new name
  "password": string   // Your new password
}
```

- Returns:

```
{
  "user_id":  string,  // Your user ID
  "name":     string,  // Your new name
  "password": string   // Your new password
}
```

`GET /acquisitions`

- Authorization: Use your access token hoo-man! I want a header with `Authorization: Bearer <token goes here>`!!
- Function: Get last month satellite acquisitions, find those ore deposits!
- Returns:

```
[
  {
    "timestamp": number,  // Acquisition Unix timestamp
    "sites":     number   // Number of detected ore sites
  }
]
```