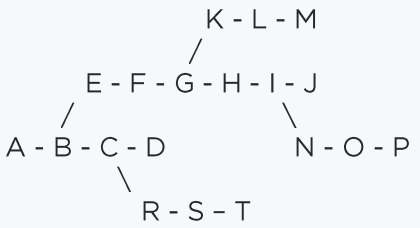
CHALLENGE - BACKEND

Imagine a stadium full of audio devices connected between them. Each one has a name composed by 4 chars (e.g. ABCD) and is always connected to another device.

Example of a network of these devices (for simplicity, the devices name use only one char):



This network is represented by a file with the following format DEVICE1->DEVICE2 XXX where DEVICE1 and DEVICE2 are the names of the devices, -> means that DEVICE1 is connected to DEVICE2 and XXX is a number that represents the latency, in milliseconds, between DEVICE1 and DEVICE2. An example of such file would be:

```
A->B 45
B->C 456
C->D 67
B->E 23
E->F 6
F->G 64
G->H 564
...
G->K 23
K->L 12
...
```

The network you will need to use for this challenge has been attached to the email you received. The name of the file is a unique UUID. This UUID is the answer of the first question.

Consider that the file:

- Only has one starting node

- There are no loops

- The nodes are NOT alphabetically ordered in the network

- The nodes are randomly organized in the file

OBJECTIVES

You need to develop a REST API with the following Routes:

THE ROUTES

PUT /network

Used to upload a file that describes the network. This file has the same syntax described above:

```
A->B 45
B->C 456
C->D 67
B->E 23
E->F 6
F->G 64
G->H 564
...
G->K 23
K->L 12
...
```

Uploading a new file will overwrite the previous uploaded network.

GET /starting-node

Gets the name of the starting node of the network. If you take into consideration the example above. the starting node would be A.

GET /paths/{node}

Returns a JSON with list of paths beginning on the passed {node}. Consider that if no {node} is passed, the {node} is the starting node of the network (the value returned by GET /starting-node)

If you take into consideration the example above, the response expected for GET /paths/G would be:

```
{
  starting_node: "G",
  paths: [
    {
      latency: 3427,
      path: ["G", "K", "L","M"]
    },
    {
      latency: 5645342,
      path: ["G", "H", "I","J"]
    },
    {
      latency: 34235,
      path: ["G", "H", "I","N","O","P"]
    }
  ]
}
```

Fake latency numbers where used in the example above.

GET /highest-time/{node}

Gets the highest latency from the starting node {node} until the end of the path. Remember that starting in {node}, several paths might occur. Consider that if no {node} is passed, the {node} is the starting node of the network (the value returned by GET /starting-node).

If you take into consideration the example network above, given the file:

```
...
C->D 34
C->R 67
R->S 85
S->T 876
...
```

The response expected for GET /highest-time/C would be:

```
{
  latency: 1028,
  path: ["C","R","S","T"]
}
```

GET /lowest-time/{node}

Gets the lowest latency from the starting node {node} until the end of the path. Remember that starting in {node}, several paths might occur. Consider that if no {node} is passed, the {node} is the starting node of the network (the value returned by GET /starting-node)

If you take into consideration the example network above, given the file:

```
...
C->D 34
C->R 67
R->S 85
S->T 876
...
```

The response expected for GET /highest-time/C would be:

```
{
  latency: 34,
  path: ["C","D"]
}
```

QUESTIONS

Your final objective is to answer the following questions:

1. What was the token used to generate the network file?
2. What is the name of starting node?
3. What are the paths in the network and their respective travel time?
4. What is the path with more hops?
5. What is the path with the biggest travel time?
6. What is the path with the shortest travel time?
7. How will you API perform if the file that describes the network has 1GB. And 10GB?
8. What documentation did you consult to accomplish your assignment?
9. How much time did you spent developing your assignment?
10. If you had unlimited time, what would you do differently?

For questions 3 to 6, assume the starting node as the origin of all paths

WHAT TO SUBMIT

You should submit a zip file or provide access to a code repository that contains all the code, documentation to run and understand your system and the answers to the above questions.

You can use any programming language you want to complete this challenge.