Technical Task

NodeJS Developer

Job Post

http://jobsearch.naukri.com/mynaukri/mn_newminnernew.php?f=150823003826

Company: www.ifelsecloud.com

Overview

Do you remember the old Nokia phone. Yeah, that one.
There are 12 keys on this phone as shown below:

1	2 ABC	3 DEF
4 ghi	5 jkl	6 mno
7 pqrs	8 tuv	9 wxyz
* +	0 space	# uppercase

Business as usual

You press a number to use the alphabets associated with it. If you press #2 once, you have **a**. If you press it twice, you get **b** and so on. Just to refresh your memory, if you are one of those kids who don't know the best of all phones! Here is what we are talking about.	

Scenario

We have just migrated to Mars and need to communicate with the remaining people on planet Earth. As you know, the only phone that can survive in such harsh conditions is this phone. However, people on planet Earth use the latest smartphones with qwerty keypad, touch screens and voice commands.

You need to write a program that can help interplanetary communication and save the world!

What do you need to do?

Setup Earth to Mars Communication

- Users on Earth write a text message in plain English.
- This message is then converted into a series of numeric including * and # symbols.
- Users on Mars will then use this numeric series and type on their phones.
- It gets converted into the text message that people on Earth sent them.
- Then people on Mars will type on their phones and create a series of numerics.
- People on Earth will use this series and convert it to plain English.

Task 1

Write a program that can convert a sentence written in English into the corresponding numeric message.

Task 2

Write a program that can convert the series of numerics into the corresponding English characters.

For example:
Message from Earth: houston do you copy
Nokia Translation: 44666887777866666 3666 99966688 2226667999

Message from Mars: 44434446668 444 26 3355566666 666668 44666887777866666
Nokia Translation: idiot i am elon not houston

Important Note:
There is a catch here. You need to define the splitter in numeric to identify what character you want to find. For example, 2.22 is ab because there is a . (splitter) in between.
Without this, 222 will become ambiguous. It can be a combination of a, ab, ba, c etc.

That's it.
If you can do this - congratulations, you have just saved the world. Now put your programming hat on and let's do it!

To help you test

There are several options online to do a quick test. We have found a nice one here to help you test - https://www.thefontworld.com/convert-to-nokia-keypad

Tech Stack on NodeJs

- Use Typescript
- Use Express/Fastify

1. API

1. Write an API with just one Controller and only one POST method
2. Define it's route as host:**port/api/earth-mars-comm/message**
3. The action will just take one parameter as string. It can be either the English text or the Numeric Series.

For example: action(message: string)

2. Middleware

1. You need to create a middleware that will read the request headers and log the sender and receiver in the console.
2. Pass the headers as shown below and log them in the console from the middleware.

x-sender = earth
x-receiver = mars

To understand whether the given message is coming from Earth or Mars, you will need to use these headers, so you can perform the right translation.
For example, if the x-sender is earth and x-receiver is mars, the message has to be converted from English to Numebrs (Task 1).

Log Translation Time
Also, log the time it takes to process the request from the time it enters the program and exits.

3. Translation Method

1. From your controller action, call a method that does the translation.
2. This method will use these header values extracted by the Middleware to perform the translation.
3. Once done, the translation is returned from the action.

4. Interceptor

1. Create an Interceptor that will modify each outgoing message, so our program can understand the communication
2. This interceptor will read the values set by the Middleware (x-sender and x-receiver).
3. If the incoming message was from Earth, the response JSON will show ‘Response From Mars’
4. If the incoming message was from Mars, the response JSON will show ‘Response from Earth’. For example:
   {
   “Response from Mars”: “44434446668 444 26 3355566666 666668 44666887777866666”
   “Nokia Translation”: “idiot i am elon not houston”
   }

   Or
   {
   “Response from Earch”: “Message in english”
   “Nokia Translation”: “13091929992222999 ”
   }

Other details

- This is a fun project and can show us that you know the basics of your Typescript, APIs, and the techstack.
- This should ideally take you 1 hour or 2, if you go Event Driven route (to show off your skills).
- We use these technology stacks and heavily rely upon Kafka, APIs, Microservices with Express, Fastify, NestJS, and in-memory db.
- While the task doesn’t have any real value (there are better ways to communicate on Mars then using a Nokia), it will help us see your programming skills.
- You can use any other library that you feel is appropriate.

To show off your skills

It's not a disadvantage if you don't have the below skills. If your program is well written and documented, you won't be penalized for not knowing these. However, if you do, why don't you tell us that! We heavily use these in our environment.

Containers

There is no need to containerize the application, but you can put a Dockerfile if you know how to containerize it.

NestJS

You can choose any framework you prefer. We use NestJS these days, so if you know it, please write the program using NestJS. This is not a disadvantage thought. A plain ExpressJs program is fine with us.

Use of in-memory DB

There is no need to store the data. However, if you know how to play with in-memory Db like SQLite, you can log the messages.

Make it Event-driven

We can make it event-driven. If you know Event-driven programming, this could be the first solution in your mind to make it a scalable and high-performing application. If you know Kafka or RabbitMQ, you can also convert it to an event-driven program.

Create three subscribers

1. First is for Earth
2. Second is the Translator
3. The third is for Mars

Here is the flow that you can use

1. Your API will send a message to #2 - to the translator.
2. The message will contain the properties of x-header. You can put these either as the property or in the message body itself.
3. The translator will then translate the message
4. Once it's translated, it will forward the message to either #1 or #3 queue, depending upon the send and receiver properties.
5. The #1 and #3 receiver will then receive the message and display it in the console.

How to submit your task?

Please commit it to a Github repository and send us the link in the email over Naukri.com
Note: We do not accept tasks submitted through LinkedIn or direct emails. We can ensure you that more than one member of our team will manually review your work and will provide feedback to each individual submission. Even if it's not up to the mark.
Read the next steps below to understand the interview process.

Next Steps

If you complete this task with good quality, you are guaranteed to be interviewed by our team and we would be super happy to bring you on board. After you submit your task, we take:

1. 1 day to review your task
2. 1 day to schedule an interview - based on your availability
3. 1 day to bring you on board - based on your availability

So, if all goes well, you can be on board with our team within a few days!

If you need more details on what we do and how we use these technologies, please visit the below pages of our website and learn about our product.

Our Application Platform Suite

https://ifelsecloud.com/smart-mes-application-platform

Our Platform Key Features

https://ifelsecloud.com/top-features-of-cloud-mes

How users use these

https://ifelsecloud.com/digital-manufacturing-software-for-modern-teams

Good Luck
@ifelsecloud