**Challenge 1**

**Part 1**  
Develop a node app that exposes the following end-points  
  
/token

* Prompts the user for a username and password. The username and password can be hard-coded in the backend
* Upon entering the correct credentials, the node app should return a token object of the following format:   
  *{    
      "token":{  
          "accessToken": "XXXXXXXXX",  
      }  
  }*
* Use structo as the jwt secret

*/about*  
Returns a static "Hello World" when requested with a correct jwt token. The token returned from /token should be used. Returns a 401 status code otherwise.  
Bonus: Suggest how this app can be tested (No need to code this part)

**- can use Advance REST Client Chrome extension to test it (I was using it to test out the code with mongodb setup).**

**- can create REACT Client to test it.**

**- can subscribe Postman API platform to test it.**

**Part 2**  
The access token can be expired. Design a new API to allow the client to securely and automatically regenerate a new token.  
Explain how to improve on token-based authentication.

**- when renew or generate token, can send out email or sms with unique code for second layer verification.**

**- if there is custom app in mobile device, can prompt option for second layer verification.**

You are encouraged to use any third-party libraries. Share your code via a public GitHub repository

<https://github.com/hunsberg/ngaikying.git>

**Challenge 2**

Note: You do not need to code anything. You can describe and explain in words.

* How would you implement a pagination api that needs to retrieve records from a high volume database?

To simplify user experience, I will opt for seek pagination. Seek pagination can solve the issue of drift issue of offset pagination when new item is added.Seek pagination is efficient on high volume database.

* Discuss at least two pagination schemes and their pros/cons

1. Offset pagination is easy to implement but inefficient on large data query because the pagination will scan through the list to the offset location.
2. Cursor pagination is has no data “drift” issue that offset pagination encountered, but more complex to implement.
3. Seek pagination provide simple user experience but is more complex to implement compare to offset pagination.

* Give examples of client request messages and server response messages in each scheme

1. **Offset Pagination**

- request: **GET /items?limit=20&offset=20**

-response: **{ “paging”:{ “total”:295, “page”:3, “pages”: 15} }**

1. **Cursors Pagination**

- request: **GET /items?limit=20&after\_id=40**

- response: **{ “after\_id”: “”}**

1. **Seek Pagination**

- request: **GET /items?limit=20$after\_id=28&sort\_by=email**

- response:

Mention any third-party library/tools you would use.

- can consider Moesif if pricing acceptable.