

Pizza Delivery Project
API
Integration With
Mailgun and Stripe

You can use localhost:3000/3001 (staging) or localhost:5000/5001 (production)

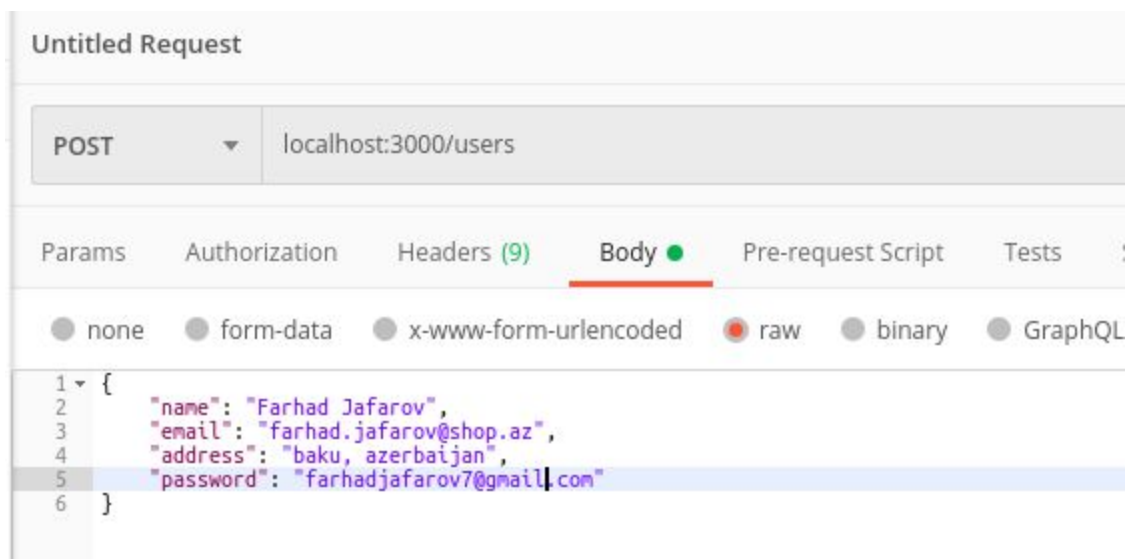
1. /(whatever) will give back 404 not found error if we pass a handler which is not found

2. /users

2.1 **POST method:** for registering a user

Required: name(payload), email(payload), address(payload), password(payload)

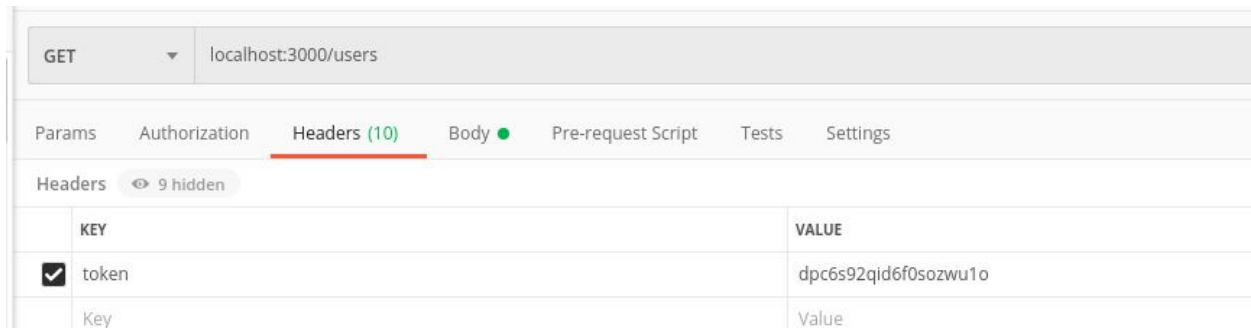
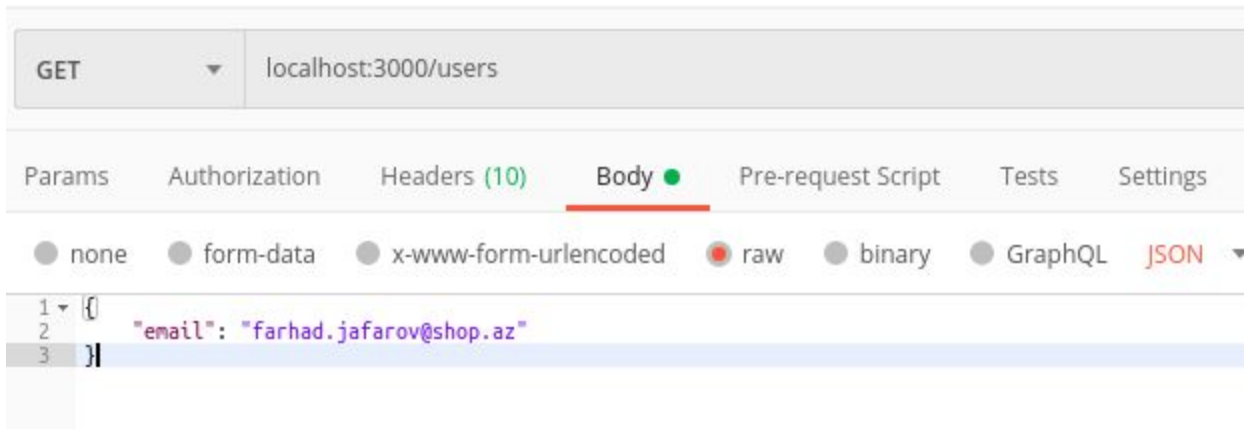
- Name should be string
- Email should be string
- Address should be string
- Password should be string and it should be more than 5 characters



2.2 GET method: for getting user details

Required: email(payload), token(headers)

- Email should be string and it should be for a user which exists
- Token has to be string and exactly 20 characters and it has to be associated with email for the user

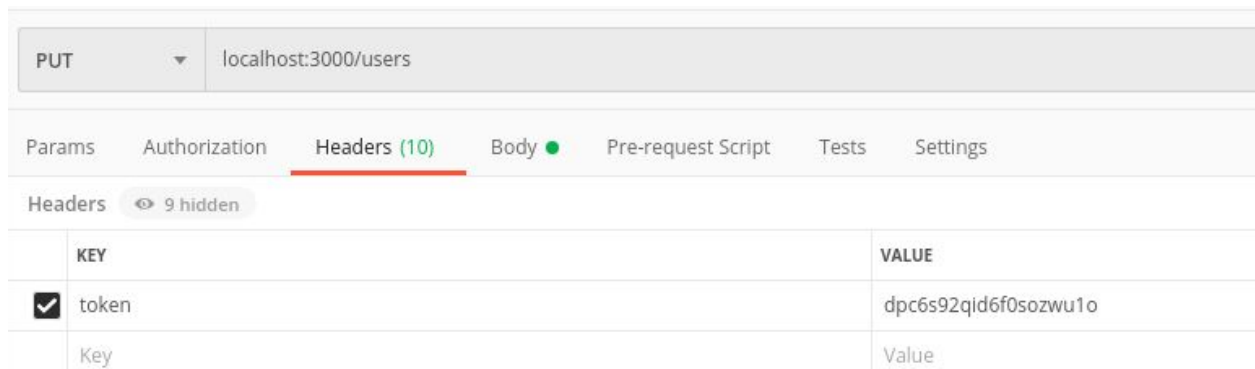
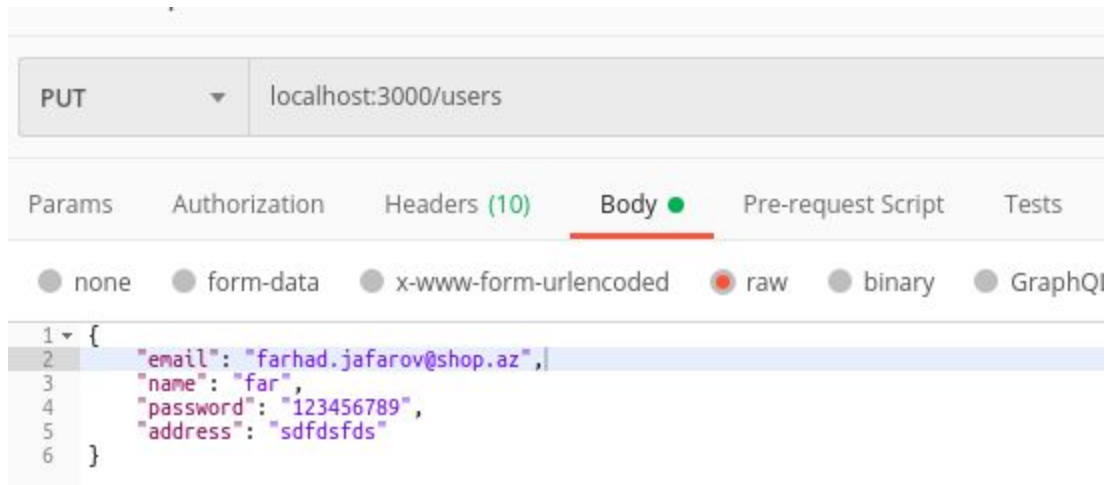


2.3 PUT method: for changing user details

Required: email(payload), token(headers)

Optional: name(payload), address(payload), password(payload)

- Email and token validators are the same like the ones above
- One of the three optionals have to be there or else nothing is going to happen by the result of the request



2.4 DELETE method: for deleting a user

Required: email(payload), token(headers),
password(payload), confirmPassword(payload)

- Email and token and password validations are as stated above
- confirmPassword validation is the same as the password validation
- Password and confirmPassword have to be equal

DELETE

localhost:3000/users

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

```
1 {  
2   "email": "farhad.jafarov@shop.az",  
3   "password": "123456789",  
4   "confirmPassword": "123456789"  
5 }
```

DELETE

localhost:3000/users

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Headers

9 hidden

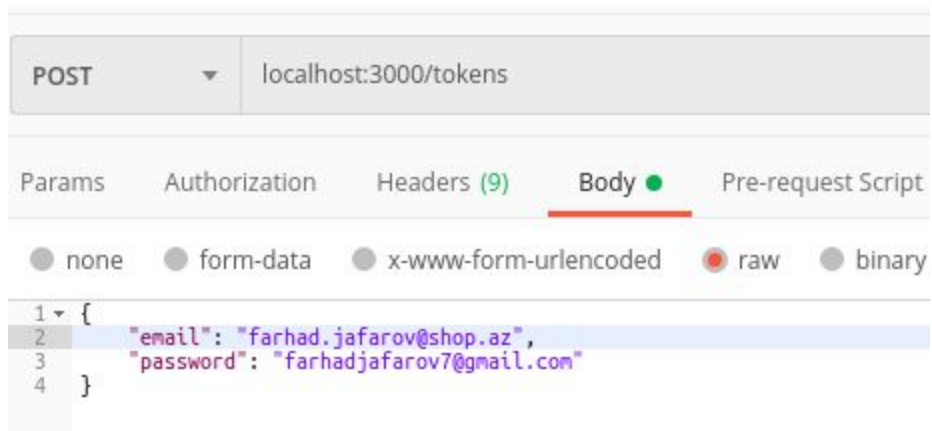
	KEY	VALUE
<input checked="" type="checkbox"/>	token	dpc6s92qid6f0sozwu1o

3. /tokens

3.1 POST method: for a user to login

Required: email(payload) and password(payload)

- Validations are the same
- A token is created with and expiry date of 3+ hours to your time



3.2 DELETE method: for a user to logout

Required: email(payload) and token(payload)

- Validation are the same as stated in previous methods

DELETE

localhost:3000/tokens

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

1 {

2 "email": "farhad.jafarov@shop.az"|

3 }

DELETE

localhost:3000/tokens

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Headers

9 hidden

	KEY	VALUE
<input checked="" type="checkbox"/>	token	svudsdqgnb922fvdh7ba
	Key	Value

4. /menu

4.1 GET method: for checking out the menu

Required: email(payload) and token(headers)

- Token and email validation is the same as in other methods
- Response is the menu.json object which has some pizzas a salad and a desert

```
JS data.js  {} menu.json x  JS handlers.js  JS
.data > menu > {} menu.json > {} 0 > type
1  [
2    {
3      "name": "ceasar",
4      "type": "salad",
5      "price": 5
6    },
7    {
8      "name": "four cheese",
9      "type": "pizza",
10     "price": 10
11   },
12   {
13     "name": "pepperoni",
14     "type": "pizza",
15     "price": 10
16   },
17   {
18     "name": "carbonara",
19     "type": "pizza",
20     "price": 12
21   },
22   {
23     "name": "chocolate",
24     "type": "desert",
25     "price": 8
26   }
27 ]
```

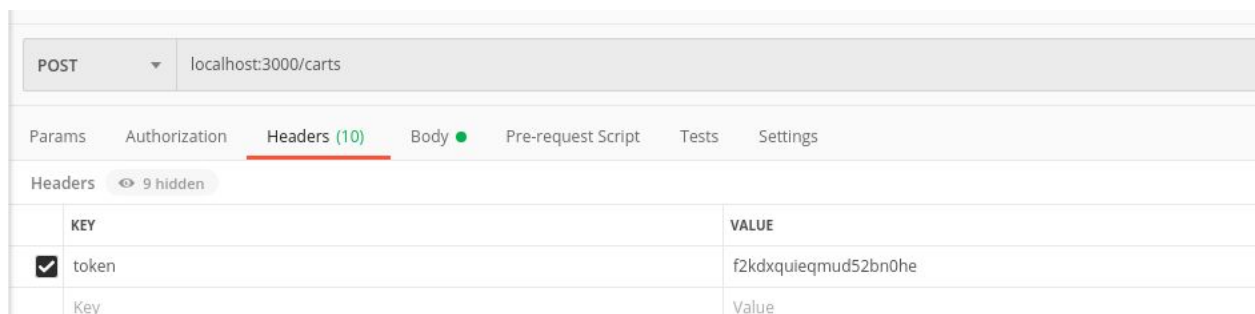
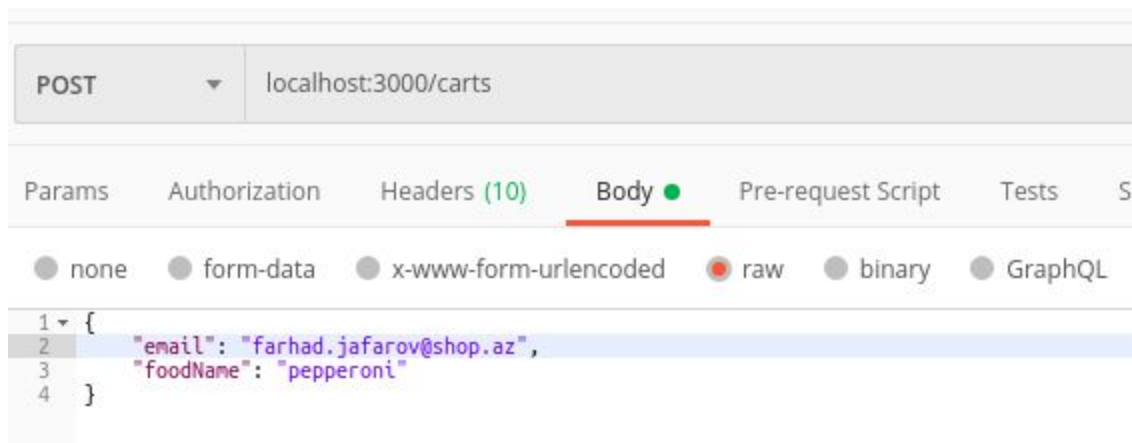
GET	localhost:3000/menu	
Params	Authorization	Headers (8)
Headers 7 hidden		
<input checked="" type="checkbox"/>	token	f2kdxquieqmud52bn0he
	Key	Value

5. /carts

5.1 POST method: to create a shopping cart

Required: email(payload), token(headers),
foodName(payload)

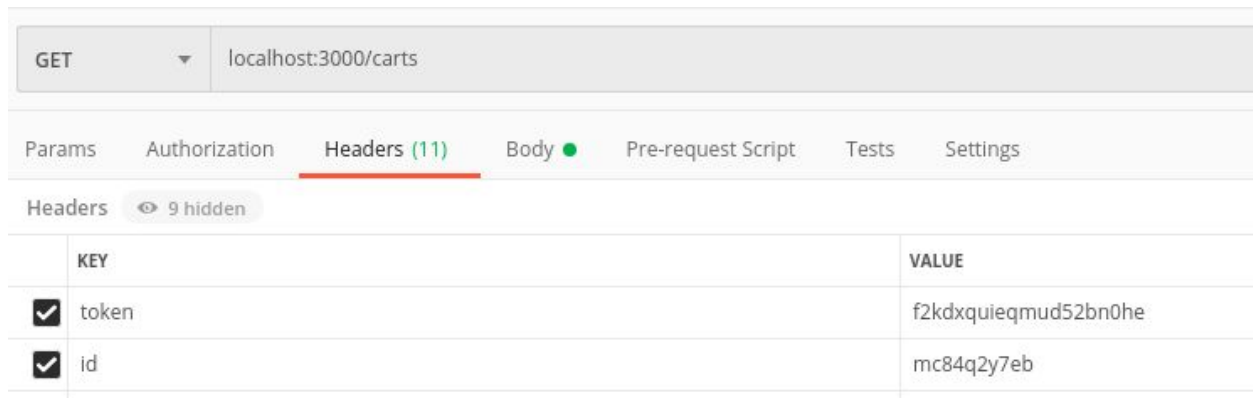
- Email and token validation is the same
- foodName is the name of the food from the menu.json file if the name is wrong you will get back an error that the food is not from the menu
- foodName should be a string



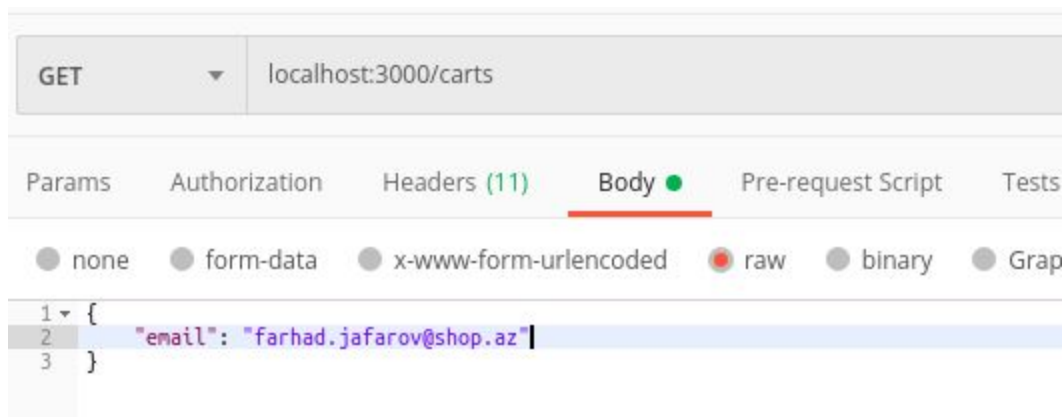
5.2 GET method: to get information on the shopping cart

Required: email(payload), token(headers) and id(headers)

- Email and token validation is the same as in previous methods
- id-is the cart id created during the /carts POST method
- id-should be 10 in length and a string



KEY	VALUE
<input checked="" type="checkbox"/> token	f2kdxquieqmud52bn0he
<input checked="" type="checkbox"/> id	mc84q2y7eb



```
1 {  
2   "email": "farhad.jafarov@shop.az"  
3 }
```

5.3 PUT method: to add something to the shopping cart

Required: email(payload), token(headers), id(headers), foodName(payload)

- Email and token and id and foodName validations are the same as stated above

PUT localhost:3000/carts

Params Authorization Headers (11) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON**

```
1 {  
2   "email": "farhad.jafarov@shop.az",  
3   "foodName": "chocolate"  
4 }
```

PUT localhost:3000/carts

Params Authorization **Headers (11)** Body Pre-request Script Tests Settings

Headers 9 hidden

KEY	VALUE
<input checked="" type="checkbox"/> token	f2kdxquieqmud52bn0he
<input checked="" type="checkbox"/> id	mc84q2y7eb
Key	Value

5.4 DELETE method: to delete item from the shopping cart

Required: email(payload), token(headers), id(headers), foodName(payload)

- Email, token, id, foodName validations are the same as stated above

DELETE localhost:3000/carts

Params Authorization Headers (11) Body Pre-request Script Tests Settings

Headers 9 hidden

KEY	VALUE
<input checked="" type="checkbox"/> token	f2kdxquieqmud52bn0he
<input checked="" type="checkbox"/> id	mc84q2y7eb

DELETE localhost:3000/carts

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "email": "farhad.jafarov@shop.az",  
3   "foodName": "chocolate"  
4 }
```

5.5 UNLINK method: to delete the whole shopping cart

Required: email(payload), token(headers), id(headers)

- Email, token and id validations are the same as stated above

UNLINK localhost:3000/carts

Params Authorization Headers (11) Body Pre-request Script Tests Settings

Headers 9 hidden

KEY	VALUE
<input checked="" type="checkbox"/> token	f2kdxquieqmud52bn0he
<input checked="" type="checkbox"/> id	mc84q2y7eb
Kev	Value

UNLINK localhost:3000/carts

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "email": "farhad.jafarov@shop.az"
3 }
```

6. /orders

6.1 POST method: to make a payment for a shopping cart

Required: email(payload), token(headers), id(headers), cardNum(payload)

- email , token and id validations are the same as stated above
- cardNum is the last four digits of the card the user is making the payment with
- There are only a set of digits acceptable by stripe so if you use anything else you will receive an error
- These last four digits are used for setting the token for payment method that the app sends to stripe

- If payment successful then an email is being sent using mailgun
- The email receiving this email is set in the config as in Mailgun you can only send emails to the email you are registered with in mailgun, in any other case you receive a 400 error

POST localhost:3000/orders

Params Authorization Headers (11) Body ● Pre-request Script Tests Settings

Headers 9 hidden

KEY	VALUE
<input checked="" type="checkbox"/> token	f2kdxquieqmud52bn0he
<input checked="" type="checkbox"/> id	469od0syli
Key	Value

POST localhost:3000/orders

Params Authorization Headers (11) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

```
1 {  
2   "email": "farhad.jafarov@shop.az",  
3   "cardNum": "4242"  
4 }
```

Stripe card numbers that will be successful: (the main thing is the last four digits for this API in the case of American Express Payments it is the last 5 digits)

Card numbers Tokens PaymentMethods

NUMBER	BRAND	CVC	DATE
4242 4242 4242 4242	Visa	Any 3 digits	Any future date
4000 0566 5566 5556	Visa (debit)	Any 3 digits	Any future date
5555 5555 5555 4444	Mastercard	Any 3 digits	Any future date
2223 0031 2200 3222	Mastercard (2-series)	Any 3 digits	Any future date
5200 8282 8282 8210	Mastercard (debit)	Any 3 digits	Any future date
5105 1051 0510 5100	Mastercard (prepaid)	Any 3 digits	Any future date
3782 822463 10005	American Express	Any 4 digits	Any future date
3714 496353 98431	American Express	Any 4 digits	Any future date
6011 1111 1111 1117	Discover	Any 3 digits	Any future date
6011 0009 9013 9424	Discover	Any 3 digits	Any future date
3056 9300 0902 0004	Diners Club	Any 3 digits	Any future date
3622 7206 2716 67	Diners Club (14 digit card)	Any 3 digits	Any future date
3566 0020 2036 0505	JCB	Any 3 digits	Any future date
6200 0000 0000 0005	UnionPay	Any 3 digits	Any future date

Do not forget to change the config.js file for your own configurations of Stripe and Mailgun. VERY IMPORTANT: THE EMAIL WILL BE SENT TO THE MAILGUN EMAIL YOU ARE GONNA WRITE IN YOUR CONFIG.JS FILE.

```
var environments = {};  
  
// staging (default) environment  
environments.staging = {  
  'httpPort': 3000,  
  'httpsPort': 3001,  
  'envName': 'staging',  
  'hashingSecret': 'suckdeep',  
  'stripe': {  
    'accountSid': 'sk_test_i6MEVCLJRP5i7xUZqNxjZHM800xcDFF5jx'  
  },  
  'mailgun': {  
    'domain': 'sandbox20dea98315714933abeldb9819245b42.mailgun.org',  
    'apiKey': '0d5412d4c1bd10e611c179428c2a02de-915161b7-2fdf90e4',  
    'userEmail': 'farhad.jafarov@shop.az'  
  }  
}  
  
// production environment  
environments.production = {  
  'httpPort': 5000,  
  'httpsPort': 5001,  
  'envName': 'production',  
  'hashingSecret': 'suckdeetoo',  
  'stripe': {  
    'accountSid': 'sk_test_i6MEVCLJRP5i7xUZqNxjZHM800xcDFF5jx'  
  },  
  'mailgun': {  
    'domain': 'sandbox34e3b01f00fa44ba98948a5c18f36b1f.mailgun.org',  
    'apiKey': '0d5412d4c1bd10e611c179428c2a02de-915161b7-2fdf90e4',  
    'userEmail': 'farhad.jafarov@shop.az'  
  }  
}  
  
// determine which environment was passed as a command-line argument  
var currentEnvironment = typeof(process.env.NODE_ENV) == 'string' ? process.env.NODE_ENV.toLowerCase() : '';  
  
// check if environment exists in config file, if not default to staging  
var environmentToExport = typeof(environments[currentEnvironment]) == 'object' ? environments[currentEnvironment] : environments.staging;  
  
// export the module  
module.exports = environmentToExport;
```

For checking expiry of the token, token has expiry date of 3hours+ plus from the date it was created, there is background worker working in a short interval to check the expiry and if it has expired the token gets deleted and you see it in the console.log, if it hasn't expired you still get a message in the console.log that it hasn't expired.