# CHAPTER 5
# IMPLEMENTATION AND TESTING

## 5.1 Implementation

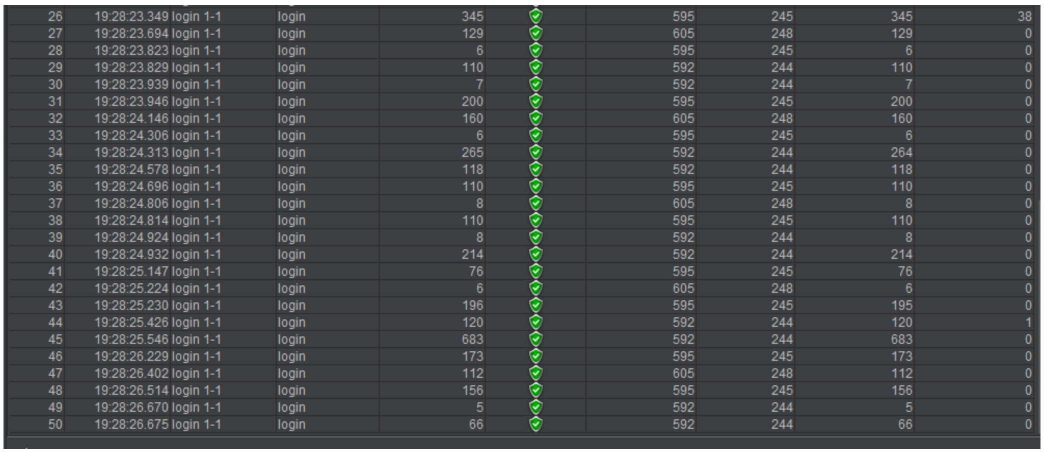
In this chapter describes the implementation, explanation of the code and also the testing carried out in this project

```
1.    func  CreateToken(w  http.ResponseWriter,  r  *http.Request)
{returns varchar(100)
2.    var account structs.Account
3.    err := json.NewDecoder(r.Body).Decode(&account)
4.    if err != nil {
5.    respondWithError(w, http.StatusBadRequest, err.Error())}
6.    token      :=      jwt.NewWithClaims(jwt.SigningMethodHS256,
jwt.MapClaims{
7.    "Username": account.Email,
8.    "Role": account.Role,
9.    "exp":       time.Now().Add(time.Hour * 168).Unix(),})
10.   tokenString, err2 := token.SignedString(mySigningKey)
11.   if err2 != nil {
12.   respondWithError(w, http.StatusBadRequest, err2.Error())}
13.   respondWithJson(w,  http.StatusOK,  structs.JWTToken{Token:
tokenString})
```

In line 1 is a command to create the CreateToken function with http respond and request for the parameters. Second row is the call of struct or classes that contain the structure and form of the account table into variable. Lines 3 to 5 contain checking the body of the http request and the structure of the account variable, whether the body of request matches the structure of the account or not. If it doesn't match, the programs will execute the 5th line which is a function to show the results if an error occurs in the request. Line 6 through 9 contain a command to make a token, if all parameters are in accordance by calling a function from the JWT directory to generate a token using HS256 method. The contents of the token that will be made are the username, role, and expiration of the token are in lines 7 to 9. Line 10 functions to input the token that has been signed into a string variable. Lines 11 through 12 function to indicate if there is an error signing the token to the string. Line 13 returns token in the form of an http response.

## 5.2 Testing

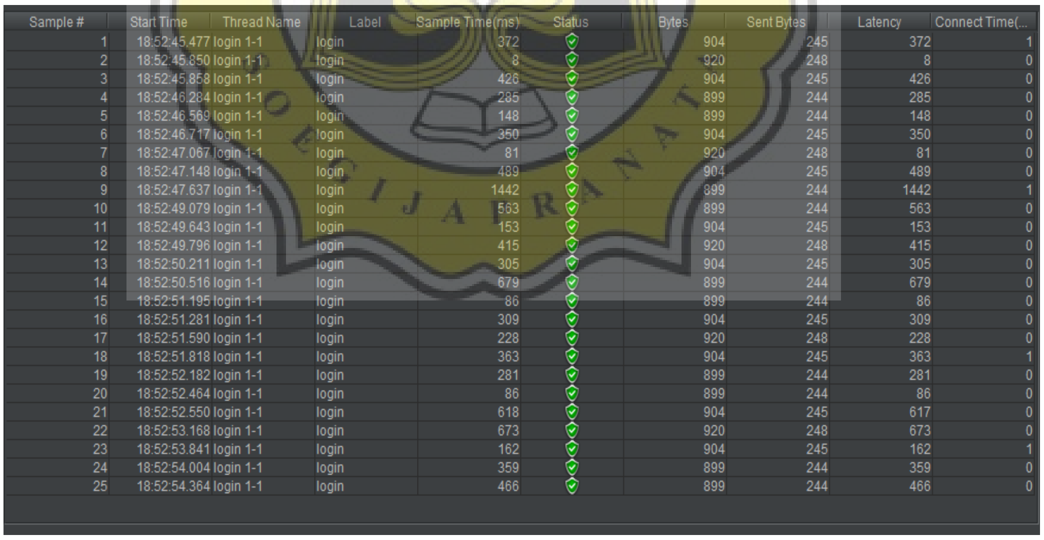
The results of testing JSON Web Token on web services.



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 26 | 19:28:23.349 | login 1-1 | login | 345 | ✓ | 595 | 245 | 345 | 38 |
| 27 | 19:28:23.694 | login 1-1 | login | 129 | ✓ | 605 | 248 | 129 | 0 |
| 28 | 19:28:23.823 | login 1-1 | login | 6 | ✓ | 595 | 245 | 6 | 0 |
| 29 | 19:28:23.829 | login 1-1 | login | 110 | ✓ | 592 | 244 | 110 | 0 |
| 30 | 19:28:23.939 | login 1-1 | login | 7 | ✓ | 592 | 244 | 7 | 0 |
| 31 | 19:28:23.946 | login 1-1 | login | 200 | ✓ | 595 | 245 | 200 | 0 |
| 32 | 19:28:24.146 | login 1-1 | login | 160 | ✓ | 605 | 248 | 160 | 0 |
| 33 | 19:28:24.306 | login 1-1 | login | 6 | ✓ | 595 | 245 | 6 | 0 |
| 34 | 19:28:24.313 | login 1-1 | login | 265 | ✓ | 592 | 244 | 264 | 0 |
| 35 | 19:28:24.578 | login 1-1 | login | 118 | ✓ | 592 | 244 | 118 | 0 |
| 36 | 19:28:24.696 | login 1-1 | login | 110 | ✓ | 595 | 245 | 110 | 0 |
| 37 | 19:28:24.806 | login 1-1 | login | 8 | ✓ | 605 | 248 | 8 | 0 |
| 38 | 19:28:24.814 | login 1-1 | login | 110 | ✓ | 595 | 245 | 110 | 0 |
| 39 | 19:28:24.924 | login 1-1 | login | 8 | ✓ | 592 | 244 | 8 | 0 |
| 40 | 19:28:24.932 | login 1-1 | login | 214 | ✓ | 592 | 244 | 214 | 0 |
| 41 | 19:28:25.147 | login 1-1 | login | 76 | ✓ | 595 | 245 | 76 | 0 |
| 42 | 19:28:25.224 | login 1-1 | login | 6 | ✓ | 605 | 248 | 6 | 0 |
| 43 | 19:28:25.230 | login 1-1 | login | 196 | ✓ | 595 | 245 | 195 | 0 |
| 44 | 19:28:25.426 | login 1-1 | login | 120 | ✓ | 592 | 244 | 120 | 1 |
| 45 | 19:28:25.546 | login 1-1 | login | 683 | ✓ | 592 | 244 | 683 | 0 |
| 46 | 19:28:26.229 | login 1-1 | login | 173 | ✓ | 595 | 245 | 173 | 0 |
| 47 | 19:28:26.402 | login 1-1 | login | 112 | ✓ | 605 | 248 | 112 | 0 |
| 48 | 19:28:26.514 | login 1-1 | login | 156 | ✓ | 595 | 245 | 156 | 0 |
| 49 | 19:28:26.670 | login 1-1 | login | 5 | ✓ | 592 | 244 | 5 | 0 |
| 50 | 19:28:26.675 | login 1-1 | login | 66 | ✓ | 592 | 244 | 66 | 0 |

Illustration 5.1: The Results of Users Logging in without Encrypted Data Simultaneously 5 Times

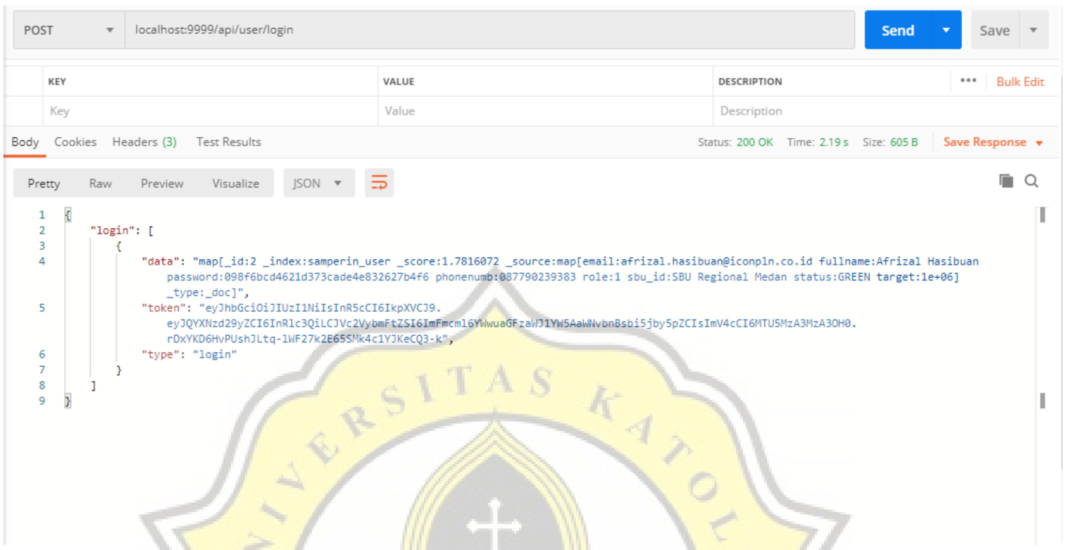
From the illustration above it can be seen that the time taken to login and request authentication takes an average of 135.56ms. The fastest time is 5ms and the longest time is 683ms



| Sample # | Start Time | Thread Name | Label | Sample Time(ms) | Status | Bytes | Sent Bytes | Latency | Connect Time(... |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 18:52:45.477 | login 1-1 | login | 372 | ✓ | 904 | 245 | 372 | 1 |
| 2 | 18:52:45.850 | login 1-1 | login | 8 | ✓ | 920 | 248 | 8 | 0 |
| 3 | 18:52:45.858 | login 1-1 | login | 426 | ✓ | 904 | 245 | 426 | 0 |
| 4 | 18:52:46.284 | login 1-1 | login | 285 | ✓ | 899 | 244 | 285 | 0 |
| 5 | 18:52:46.569 | login 1-1 | login | 148 | ✓ | 899 | 244 | 148 | 0 |
| 6 | 18:52:46.717 | login 1-1 | login | 350 | ✓ | 904 | 245 | 350 | 0 |
| 7 | 18:52:47.067 | login 1-1 | login | 81 | ✓ | 920 | 248 | 81 | 0 |
| 8 | 18:52:47.148 | login 1-1 | login | 489 | ✓ | 904 | 245 | 489 | 0 |
| 9 | 18:52:47.637 | login 1-1 | login | 1442 | ✓ | 899 | 244 | 1442 | 1 |
| 10 | 18:52:49.079 | login 1-1 | login | 563 | ✓ | 899 | 244 | 563 | 0 |
| 11 | 18:52:49.643 | login 1-1 | login | 153 | ✓ | 904 | 245 | 153 | 0 |
| 12 | 18:52:49.796 | login 1-1 | login | 415 | ✓ | 920 | 248 | 415 | 0 |
| 13 | 18:52:50.211 | login 1-1 | login | 305 | ✓ | 904 | 245 | 305 | 0 |
| 14 | 18:52:50.516 | login 1-1 | login | 679 | ✓ | 899 | 244 | 679 | 0 |
| 15 | 18:52:51.195 | login 1-1 | login | 86 | ✓ | 899 | 244 | 86 | 0 |
| 16 | 18:52:51.281 | login 1-1 | login | 309 | ✓ | 904 | 245 | 309 | 0 |
| 17 | 18:52:51.590 | login 1-1 | login | 228 | ✓ | 920 | 248 | 228 | 0 |
| 18 | 18:52:51.818 | login 1-1 | login | 363 | ✓ | 904 | 245 | 363 | 1 |
| 19 | 18:52:52.182 | login 1-1 | login | 281 | ✓ | 899 | 244 | 281 | 0 |
| 20 | 18:52:52.464 | login 1-1 | login | 86 | ✓ | 899 | 244 | 86 | 0 |
| 21 | 18:52:52.550 | login 1-1 | login | 618 | ✓ | 904 | 245 | 617 | 0 |
| 22 | 18:52:53.168 | login 1-1 | login | 673 | ✓ | 920 | 248 | 673 | 0 |
| 23 | 18:52:53.841 | login 1-1 | login | 162 | ✓ | 904 | 245 | 162 | 1 |
| 24 | 18:52:54.004 | login 1-1 | login | 359 | ✓ | 899 | 244 | 359 | 0 |
| 25 | 18:52:54.364 | login 1-1 | login | 466 | ✓ | 899 | 244 | 466 | 0 |

Illustration 5.2: The Results of Users Logging in with Encrypted Data Simultaneously 5 Times

From the illustration above it can be seen that the average time required when logging in with encrypted data tends to be longer, which is 373.88ms. The fastest time is 8ms and the longest time is 1442ms.

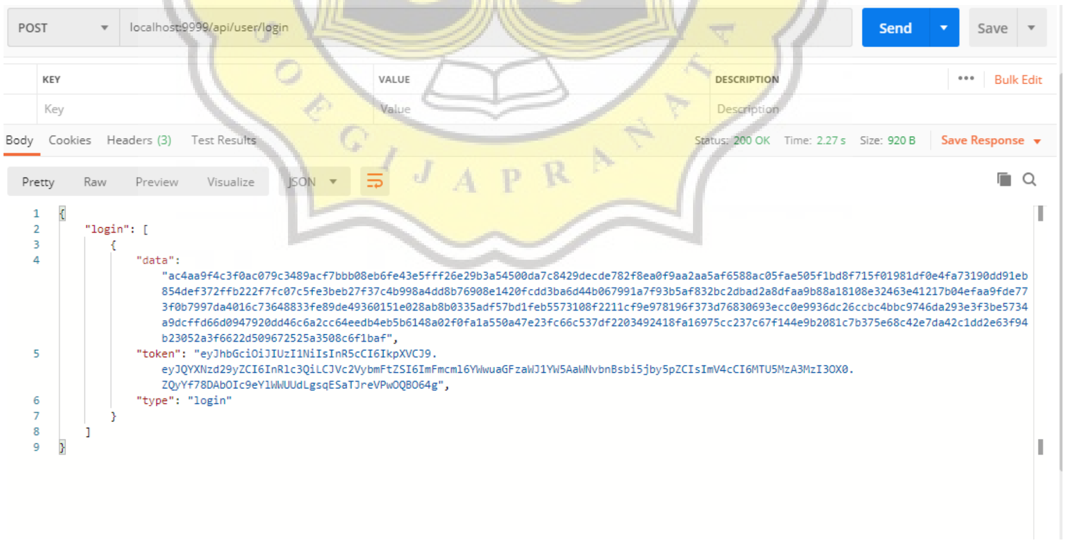


POST localhost:9999/api/user/login

Status: 200 OK Time: 2.19 s Size: 605 B

```
{
    "login": [
        {
            "data": "map[_id:2 _index:samperin_user _score:1.7816072 _source:map[email:afrizal.hasibuan@iconpln.co.id fullname:Afrizal Hasibuan
                password:098f6bcd4621d373cade4e832627b4f6 phonenumb:087790239383 role:1 sbu_id:SBU Regional Medan status:GREEN target:1e+06]
                _type:_doc]",
            "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
                eyJQYXNzd29yZCI6InRlc3QiLCJVc2VybmFtZSI6ImFmcml6YWwuaGFzaWJ1YW5AaWNvbnBsbi5jby5pZCIsImV4cCI6MTU5MzA3MzA3OH0.
                rDxYKD6HvPUshJLtq-lWF27k2E65SMk4c1YJKeCQ3-k",
            "type": "login"
        }
    ]
}
```

Illustration 5.3: Data at the Time User Login is not Encrypted



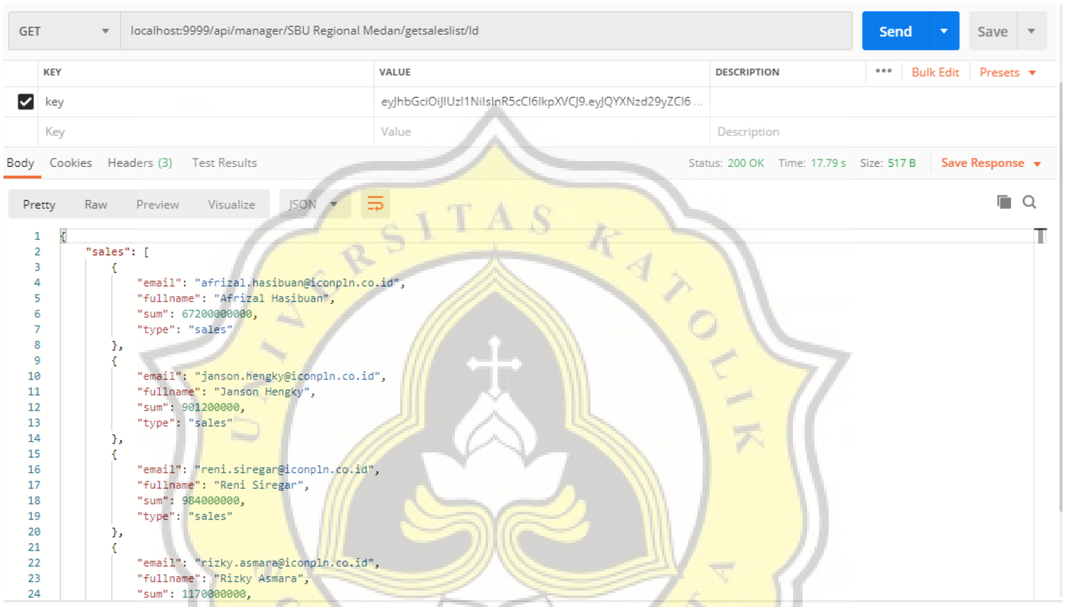
POST localhost:9999/api/user/login

Status: 200 OK Time: 2.27 s Size: 920 B

```
{
    "login": [
        {
            "data":
                "ac4aa9f4c3f0ac079c3489acf7bbb08eb6fe43e5fff26e29b3a54500da7c8429decde782f8ea0f9aa2aa5af6588ac05fae505f1bd8f715f01981df0e4fa73190dd91eb
                854def372ffb222f7fc07c5fe3beb27f37c4b998a4dd8b76908e1420fcdd3ba6d44b067991a7f93b5af832bc2dbad2a8dfaa9b88a18108e32463e41217b04efaa9fde77
                3f0b7997da4016c73648833fe89de49360151e028ab8b0335adf57bd1feb5573108f2211cf9e978196f373d76830693ecc0e9936dc26ccbc4bbc9746da293e3f3be5734
                a9dcffd66d0947920dd46c6a2cc64eedb4eb5b6148a02f0fa1a550a47e23fc66c537df2203492418fa16975cc237c67f144e9b2081c7b375e68c42e7da42c1dd2e63f94
                b23052a3f6622d509672525a3508c6f1baf",
            "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
                eyJQYXNzd29yZCI6InRlc3QiLCJVc2VybmFtZSI6ImFmcml6YWwuaGFzaWJ1YW5AaWNvbnBsbi5jby5pZCIsImV4cCI6MTU5MzA3MzI3OX0.
                ZQyYf78DAbOIc9eY1WWUUdLgsqESaTJreVPwOQBO64g",
            "type": "login"
        }
    ]
}
```

Illustration 5.4: Data When the User Login is Encrypted

Based on the results of testing (illustration 5.3), it can be seen that user data at login can be seen clearly and the data directly sent from the database. There are important data such as user passwords which even though already encrypted with the MD5 method. It is different in illustration 5.4. The data that sent by the server when the user is logged in is encrypted with the AES method. Although it can be seen, but the data will be difficult to decrypt as long as the key used is unknown to any party.



localhost:9999/api/manager/SBU Regional Medan/getsaleslist/Id

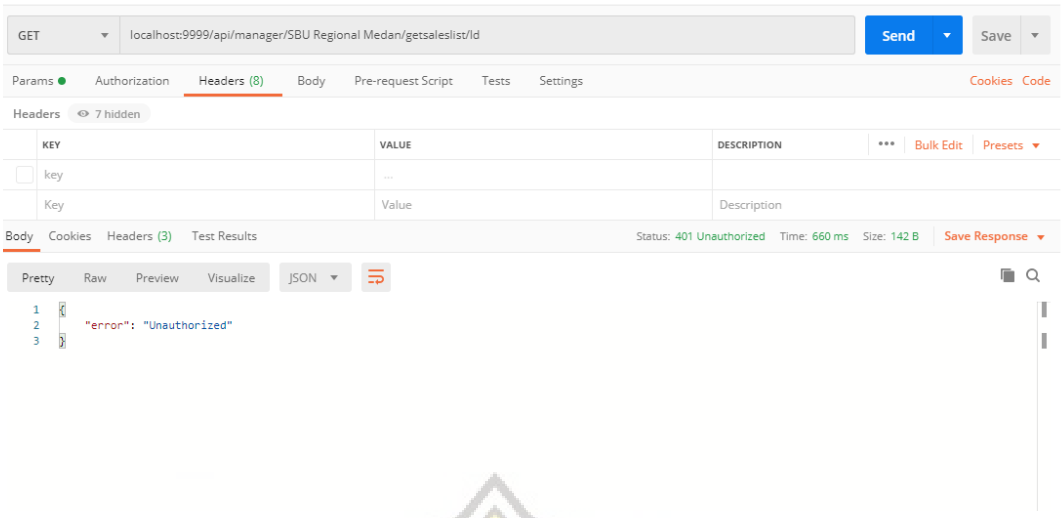
| | KEY | VALUE | DESCRIPTION |
|---|---|---|---|
| ✓ | key | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJQYXNzd29yZCI6 ... | |
| | Key | Value | Description |

Status: 200 OK Time: 17.79 s Size: 517 B

```
{
    "sales": [
        {
            "email": "afrizal.hasibuan@iconpln.co.id",
            "fullname": "Afrizal Hasibuan",
            "sum": 67200000000,
            "type": "sales"
        },
        {
            "email": "janson.hengky@iconpln.co.id",
            "fullname": "Janson Hengky",
            "sum": 901200000,
            "type": "sales"
        },
        {
            "email": "reni.siregar@iconpln.co.id",
            "fullname": "Reni Siregar",
            "sum": 984000000,
            "type": "sales"
        },
        {
            "email": "rizky.asmara@iconpln.co.id",
            "fullname": "Rizky Asmara",
            "sum": 1170000000,
```

Illustration 5.5: Getting Leads Status with Tokens

The illustration above shows the results of the output on GetSales service. Illustration 5.5 shows that with tokens that have been authenticated and have not expired, the output will appear according to user requests but the output is not encrypted.



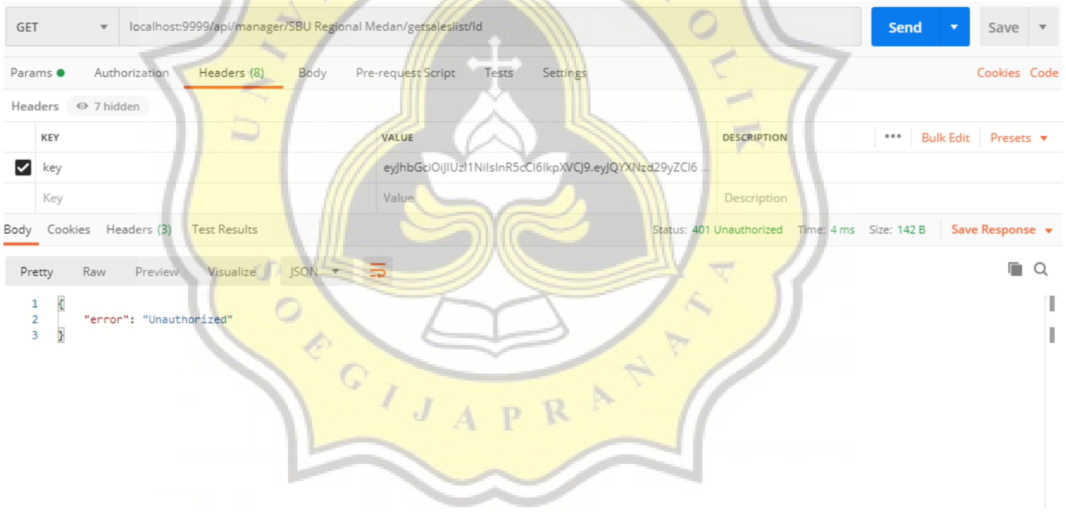
localhost:9999/api/manager/SBU Regional Medan/getsaleslist/Id

| KEY | VALUE | DESCRIPTION |
|---|---|---|
| key | ... | |
| Key | Value | Description |

Status: 401 Unauthorized Time: 660 ms Size: 142 B

```
{
    "error": "Unauthorized"
}
```

Illustration 5.6: Getting Leads Status without Tokens



localhost:9999/api/manager/SBU Regional Medan/getsaleslist/Id

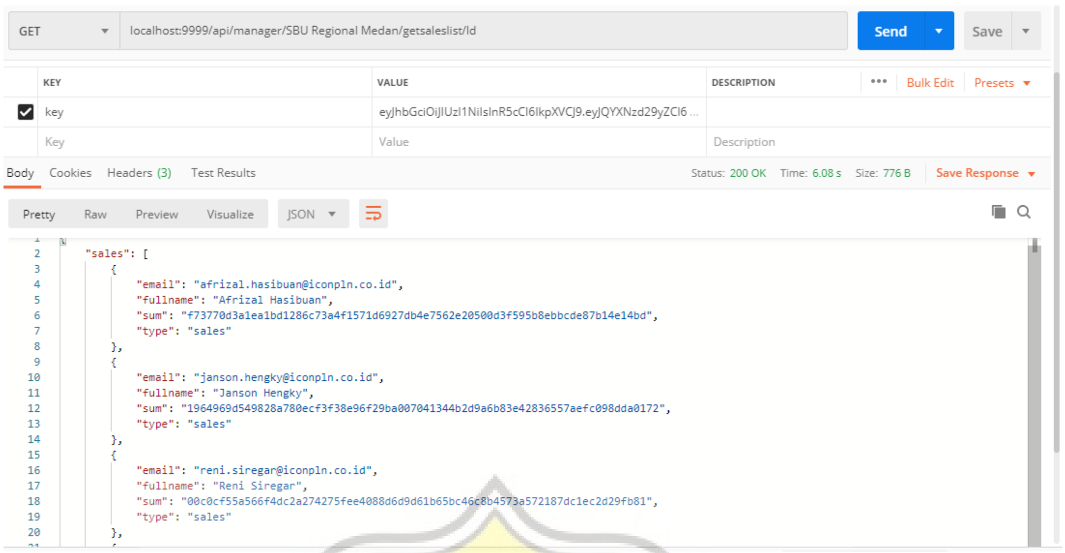
| KEY | VALUE | DESCRIPTION |
|---|---|---|
| ☑ key | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJQYXNzd29yZCI6 ... | |
| Key | Value | Description |

Status: 401 Unauthorized Time: 4 ms Size: 142 B

```
{
    "error": "Unauthorized"
}
```

Illustration 5.7: Getting Leads Status with Expired Tokens

Unlike the case in illustration 5.6, there is no token is sent in the request header. Then the output will give the output that there is no authorization. And in illustration 5.7, there is a token in the request but the resulting output is the same as if there were no tokens. That is because the token sent is expired even though the contents of the token are in accordance with the requirements to access the service.



localhost:9999/api/manager/SBU Regional Medan/getsaleslist/Id

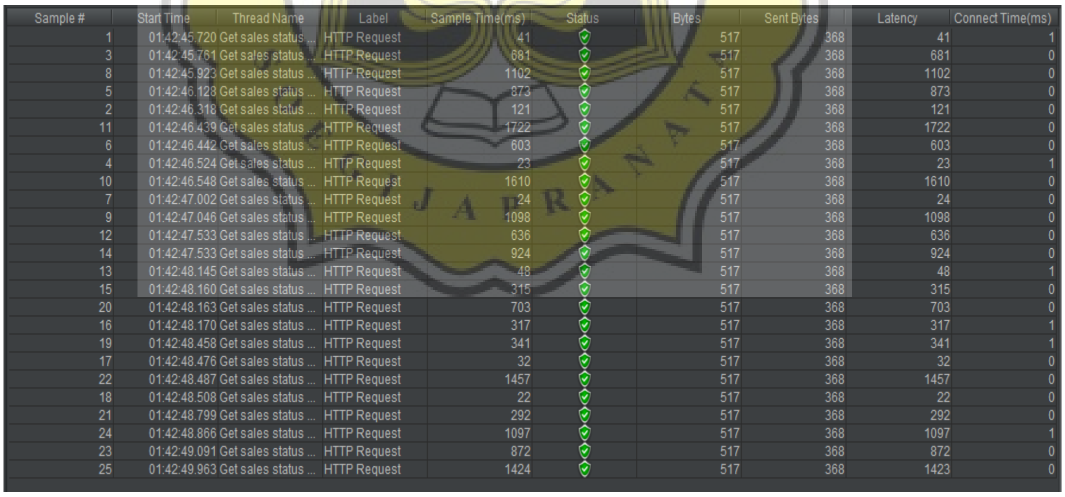
| | KEY | VALUE | DESCRIPTION |
|---|---|---|---|
| ✓ | key | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJQYXNzd29yZCI6 ... | |
| | Key | Value | Description |

Status: 200 OK Time: 6.08 s Size: 776 B

```
    "sales": [
        {
            "email": "afrizal.hasibuan@iconpln.co.id",
            "fullname": "Afrizal Hasibuan",
            "sum": "f73770d3a1ea1bd1286c73a4f1571d6927db4e7562e20500d3f595b8ebbcde87b14e14bd",
            "type": "sales"
        },
        {
            "email": "janson.hengky@iconpln.co.id",
            "fullname": "Janson Hengky",
            "sum": "1964969d549828a780ecf3f38e96f29ba007041344b2d9a6b83e42836557aefc098dda0172",
            "type": "sales"
        },
        {
            "email": "reni.siregar@iconpln.co.id",
            "fullname": "Reni Siregar",
            "sum": "00c0cf55a566f4dc2a274275fee4088d6d9d61b65bc46c8b4573a572187dc1ec2d29fb81",
            "type": "sales"
        },
```

Illustration 5.8: Getting Leads Status with Correct Tokens and Encrypted Data

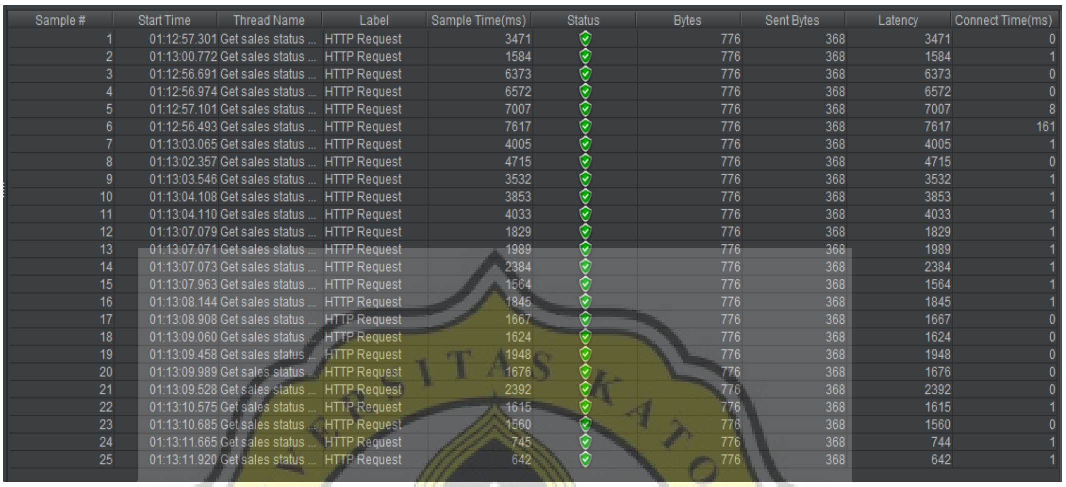
According to the illustration above, the token sent in the header is already authenticated and has received a response with encrypted data.



| Sample # | Start Time | Thread Name | Label | Sample Time(ms) | Status | Bytes | Sent Bytes | Latency | Connect Time(ms) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 01:42:45.720 | Get sales status ... | HTTP Request | 41 | ✓ | 517 | 368 | 41 | 1 |
| 3 | 01:42:45.761 | Get sales status ... | HTTP Request | 681 | ✓ | 517 | 368 | 681 | 0 |
| 8 | 01:42:45.923 | Get sales status ... | HTTP Request | 1102 | ✓ | 517 | 368 | 1102 | 0 |
| 5 | 01:42:46.128 | Get sales status ... | HTTP Request | 873 | ✓ | 517 | 368 | 873 | 0 |
| 2 | 01:42:46.318 | Get sales status ... | HTTP Request | 121 | ✓ | 517 | 368 | 121 | 0 |
| 11 | 01:42:46.439 | Get sales status ... | HTTP Request | 1722 | ✓ | 517 | 368 | 1722 | 0 |
| 6 | 01:42:46.442 | Get sales status ... | HTTP Request | 603 | ✓ | 517 | 368 | 603 | 0 |
| 4 | 01:42:46.524 | Get sales status ... | HTTP Request | 23 | ✓ | 517 | 368 | 23 | 1 |
| 10 | 01:42:46.548 | Get sales status ... | HTTP Request | 1610 | ✓ | 517 | 368 | 1610 | 0 |
| 7 | 01:42:47.002 | Get sales status ... | HTTP Request | 24 | ✓ | 517 | 368 | 24 | 0 |
| 9 | 01:42:47.046 | Get sales status ... | HTTP Request | 1098 | ✓ | 517 | 368 | 1098 | 0 |
| 12 | 01:42:47.533 | Get sales status ... | HTTP Request | 636 | ✓ | 517 | 368 | 636 | 0 |
| 14 | 01:42:47.533 | Get sales status ... | HTTP Request | 924 | ✓ | 517 | 368 | 924 | 0 |
| 13 | 01:42:48.145 | Get sales status ... | HTTP Request | 48 | ✓ | 517 | 368 | 48 | 1 |
| 15 | 01:42:48.160 | Get sales status ... | HTTP Request | 315 | ✓ | 517 | 368 | 315 | 0 |
| 20 | 01:42:48.163 | Get sales status ... | HTTP Request | 703 | ✓ | 517 | 368 | 703 | 0 |
| 16 | 01:42:48.170 | Get sales status ... | HTTP Request | 317 | ✓ | 517 | 368 | 317 | 1 |
| 19 | 01:42:48.458 | Get sales status ... | HTTP Request | 341 | ✓ | 517 | 368 | 341 | 1 |
| 17 | 01:42:48.476 | Get sales status ... | HTTP Request | 32 | ✓ | 517 | 368 | 32 | 0 |
| 22 | 01:42:48.487 | Get sales status ... | HTTP Request | 1457 | ✓ | 517 | 368 | 1457 | 0 |
| 18 | 01:42:48.508 | Get sales status ... | HTTP Request | 22 | ✓ | 517 | 368 | 22 | 0 |
| 21 | 01:42:48.799 | Get sales status ... | HTTP Request | 292 | ✓ | 517 | 368 | 292 | 0 |
| 24 | 01:42:48.866 | Get sales status ... | HTTP Request | 1097 | ✓ | 517 | 368 | 1097 | 1 |
| 23 | 01:42:49.091 | Get sales status ... | HTTP Request | 872 | ✓ | 517 | 368 | 872 | 0 |
| 25 | 01:42:49.963 | Get sales status ... | HTTP Request | 1424 | ✓ | 517 | 368 | 1423 | 0 |

Illustration 5.9: 5 Users (threads) Accessing GetSales without Encrypted Data Simultaneously 5 Times

The illustration above shows the time taken by 5 users (threads) when accessing GetSales service within 1 second 5 times. The average time needed is 655.12ms, the fastest time is 22ms, and the longest time is 1722ms.



| Sample # | Start Time | Thread Name | Label | Sample Time(ms) | Status | Bytes | Sent Bytes | Latency | Connect Time(ms) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 01:12:57.301 | Get sales status ... | HTTP Request | 3471 | ✓ | 776 | 368 | 3471 | 0 |
| 2 | 01:13:00.772 | Get sales status ... | HTTP Request | 1584 | ✓ | 776 | 368 | 1584 | 1 |
| 3 | 01:12:56.691 | Get sales status ... | HTTP Request | 6373 | ✓ | 776 | 368 | 6373 | 0 |
| 4 | 01:12:56.974 | Get sales status ... | HTTP Request | 6572 | ✓ | 776 | 368 | 6572 | 0 |
| 5 | 01:12:57.101 | Get sales status ... | HTTP Request | 7007 | ✓ | 776 | 368 | 7007 | 8 |
| 6 | 01:12:56.493 | Get sales status ... | HTTP Request | 7617 | ✓ | 776 | 368 | 7617 | 161 |
| 7 | 01:13:03.065 | Get sales status ... | HTTP Request | 4005 | ✓ | 776 | 368 | 4005 | 1 |
| 8 | 01:13:02.357 | Get sales status ... | HTTP Request | 4715 | ✓ | 776 | 368 | 4715 | 0 |
| 9 | 01:13:03.546 | Get sales status ... | HTTP Request | 3532 | ✓ | 776 | 368 | 3532 | 1 |
| 10 | 01:13:04.108 | Get sales status ... | HTTP Request | 3853 | ✓ | 776 | 368 | 3853 | 1 |
| 11 | 01:13:04.110 | Get sales status ... | HTTP Request | 4033 | ✓ | 776 | 368 | 4033 | 1 |
| 12 | 01:13:07.079 | Get sales status ... | HTTP Request | 1829 | ✓ | 776 | 368 | 1829 | 1 |
| 13 | 01:13:07.071 | Get sales status ... | HTTP Request | 1989 | ✓ | 776 | 368 | 1989 | 1 |
| 14 | 01:13:07.073 | Get sales status ... | HTTP Request | 2384 | ✓ | 776 | 368 | 2384 | 1 |
| 15 | 01:13:07.963 | Get sales status ... | HTTP Request | 1564 | ✓ | 776 | 368 | 1564 | 1 |
| 16 | 01:13:08.144 | Get sales status ... | HTTP Request | 1845 | ✓ | 776 | 368 | 1845 | 1 |
| 17 | 01:13:08.908 | Get sales status ... | HTTP Request | 1667 | ✓ | 776 | 368 | 1667 | 0 |
| 18 | 01:13:09.060 | Get sales status ... | HTTP Request | 1624 | ✓ | 776 | 368 | 1624 | 0 |
| 19 | 01:13:09.458 | Get sales status ... | HTTP Request | 1948 | ✓ | 776 | 368 | 1948 | 0 |
| 20 | 01:13:09.989 | Get sales status ... | HTTP Request | 1676 | ✓ | 776 | 368 | 1676 | 0 |
| 21 | 01:13:09.528 | Get sales status ... | HTTP Request | 2392 | ✓ | 776 | 368 | 2392 | 0 |
| 22 | 01:13:10.575 | Get sales status ... | HTTP Request | 1615 | ✓ | 776 | 368 | 1615 | 1 |
| 23 | 01:13:10.685 | Get sales status ... | HTTP Request | 1560 | ✓ | 776 | 368 | 1560 | 0 |
| 24 | 01:13:11.665 | Get sales status ... | HTTP Request | 745 | ✓ | 776 | 368 | 744 | 1 |
| 25 | 01:13:11.920 | Get sales status ... | HTTP Request | 642 | ✓ | 776 | 368 | 642 | 1 |

Illustration 5.10: 5 Users (threads) Accessing GetSales with Encrypted Data Simultaneously 5 Times

Based on the illustration above, it can be seen that the average time required to respond to 5 users simultaneously is greater than if it did not use the encryption. The average time is 3049.68ms which is almost 5 times longer than the unencrypted. The fastest time is 642ms and the longest time is 7617ms.

| No | Encrypted and authorized | Not Authorized | Expired token |
|---|---|---|---|
| 1 | Sample Start:2020-06-19 01:13:11 ICT<br><br>Load time:642<br><br>Connect Time:1<br><br>Latency:642<br><br>Size in bytes:776 | Sample Start:2020-06-19 01:14:33 ICT<br><br>Load time:122<br><br>Connect Time:2<br><br>Latency:122<br><br>Size in bytes:142 | Sample Start:2020-06-19 01:22:13 ICT<br><br>Load time:3<br><br>Connect Time:2<br><br>Latency:3<br><br>Size in bytes:142 |

| | | | |
|---|---|---|---|
| | Sent bytes:368<br><br>Headers size in bytes:109<br><br>Body size in bytes:667<br><br>Sample Count:1<br><br>Error Count:0<br><br>Data type ("text"\|"bin"\|""):text<br><br>Response code:200<br><br>Response message:OK | Sent bytes:173<br><br>Headers size in bytes:118<br><br>Body size in bytes:24<br><br>Sample Count:1<br><br>Error Count:1<br><br>Data type ("text"\|"bin"\|""):text<br><br>Response code:401<br><br>Response message:Unauthorized | Sent bytes:368<br><br>Headers size in bytes:118<br><br>Body size in bytes:24<br><br>Sample Count:1<br><br>Error Count:1<br><br>Data type ("text"\|"bin"\|""):text<br><br>Response code:401<br><br>Response message:Unauthorized |

Table 5.9: Comparison of the Three Outputs with Different Tokens Conditions

It can be seen that the output of each condition is different. In conditions with no tokens used, the size of sent bytes is smaller than the size when the correct token sent. And for expired tokens, the size of sent bytes is the same as the non-expired token. But the size in bytes and the body size are smaller than the correct token.