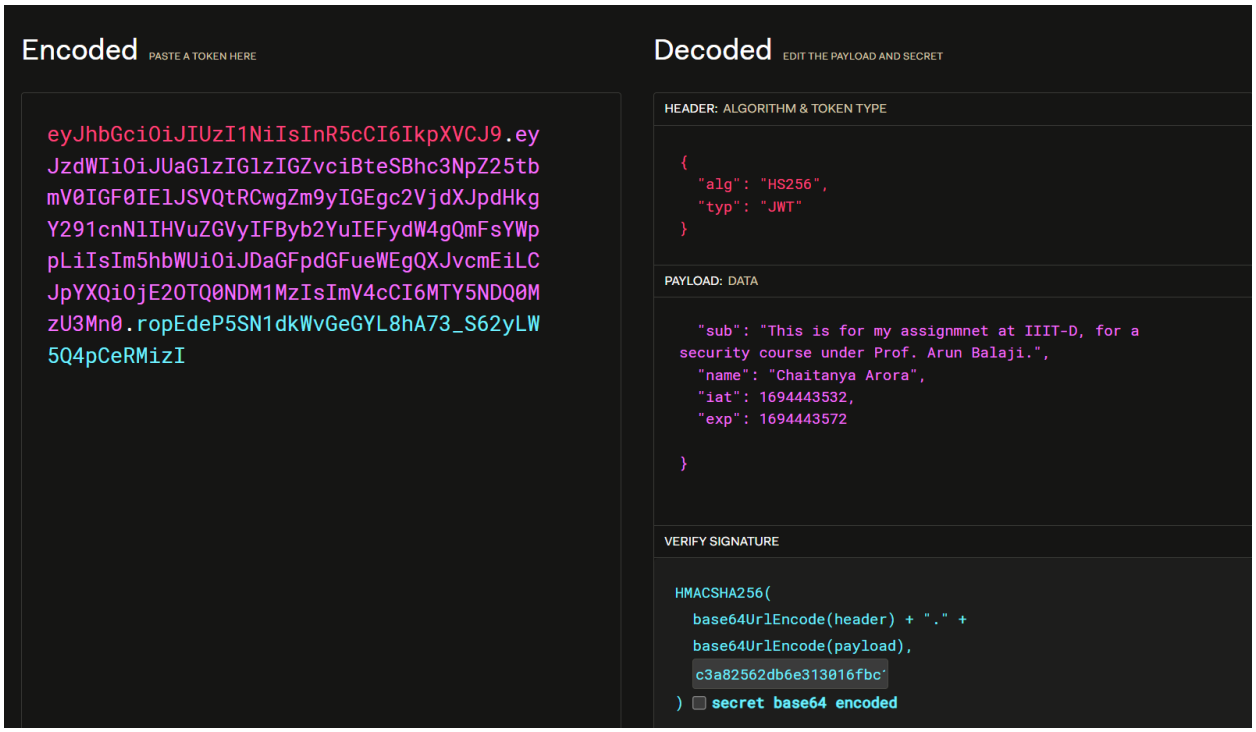
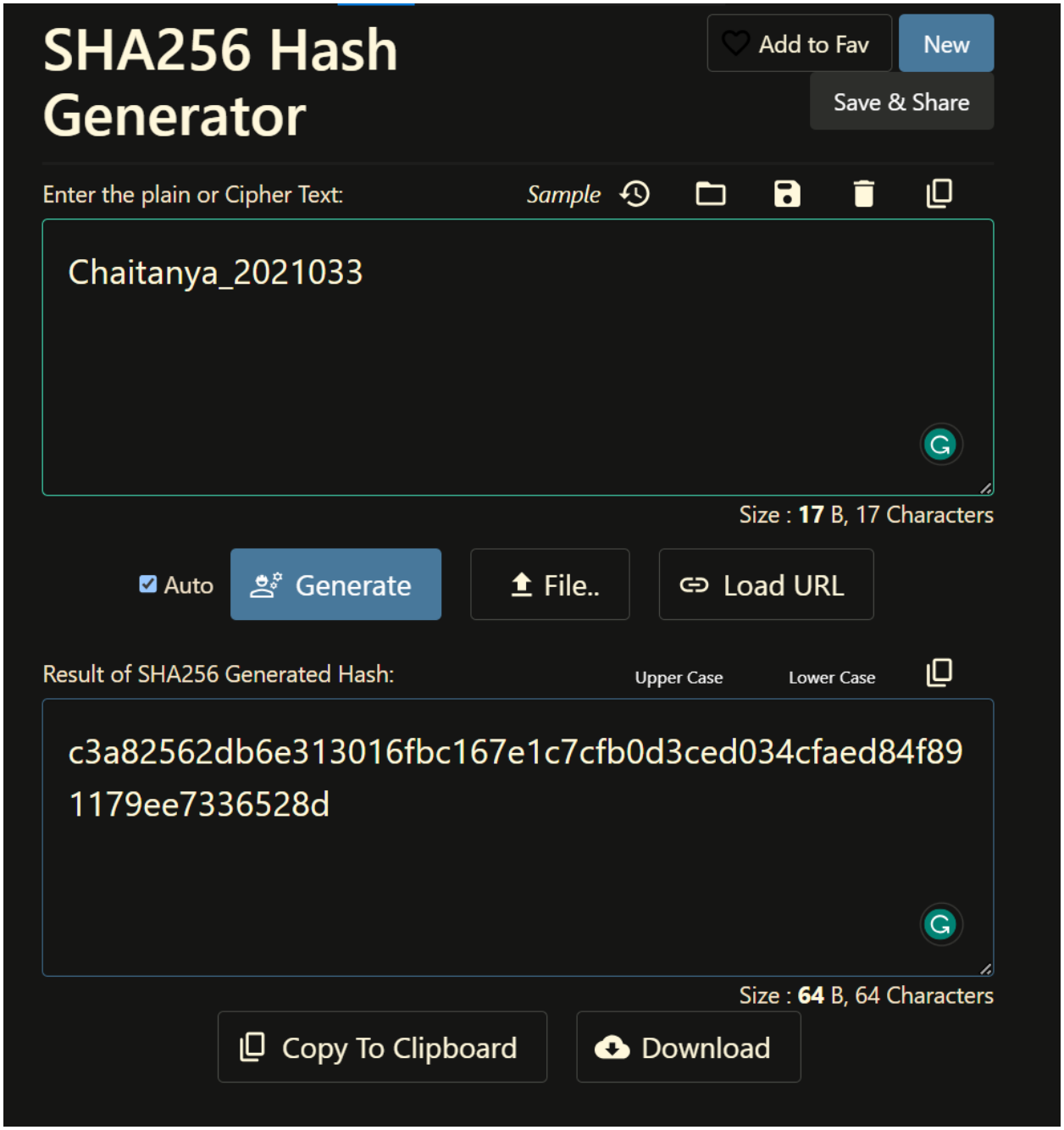
Define a Python function `verifyJwt(token, secret)` that takes in a JWT and a secret as arguments. Validate the token's signature against the supplied secret. If it is valid, return the decoded payload. Otherwise, throw an exception. The function should implement checks for at least two symmetric algorithms of your choice. For testing, you can demonstrate by generating any jwt token from the jwt debugger in the above link.
Note: You are not allowed to use the "PyJWT" library, or any other library implementing JWT verification. (10 Marks)

Here are some of the test cases I ran on my code to verify JWTs.
Screenshots added:



The secret key I used to generate this JWT was: *The sha of my name:*

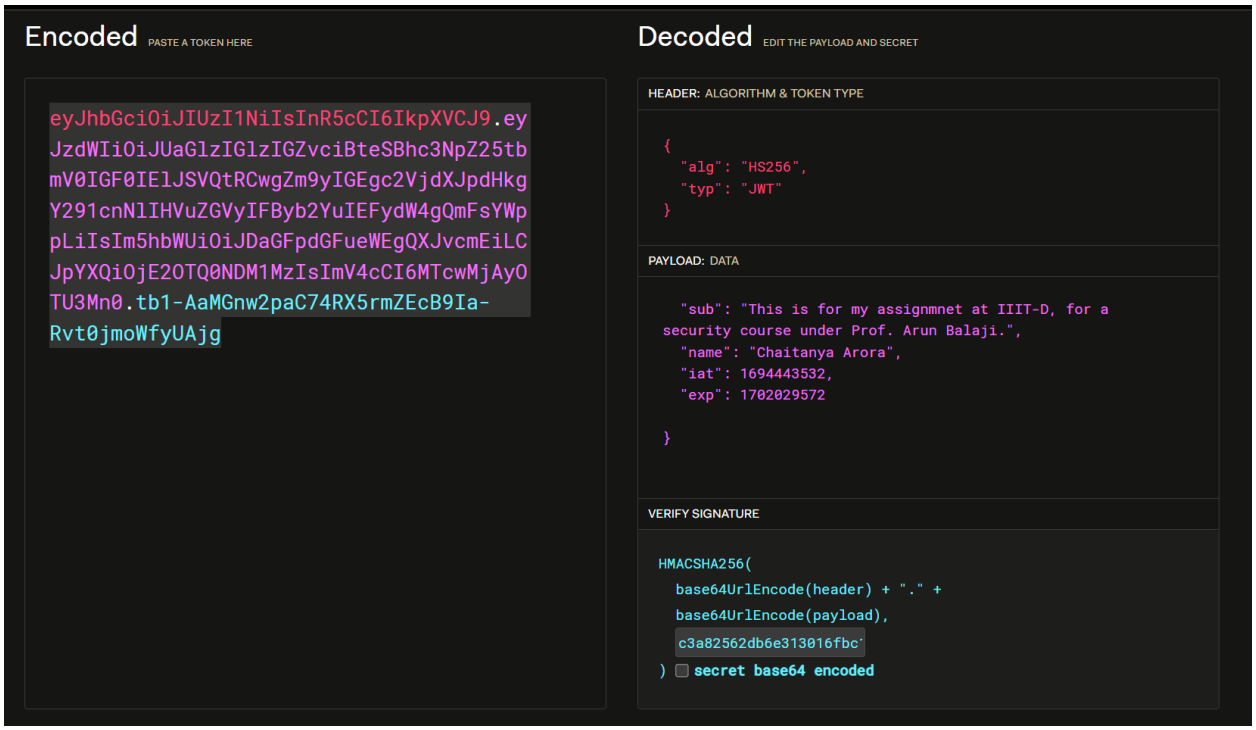


After this I entered this information into my program and this was the output

```
PS C:\Users\om> & C:/Users/om/AppData/Local/Programs/Python/Python31
1/python.exe c:/Users/om/Desktop/sem5/FCS/2021033_q3.py
Please enter the token: eyJhbGciOiJIUzM4NCIsInR5cCI6IkpXVCJ9.eyJzdWI
iOiJUaGlzIGlzIGZvciBteSBhc3NpZ25tbmV0IGF0IElJSVQtRCwgZm9yIGEgc2VjdXJ
pdHkgY291cnNlIHVuZGVyIFByb2YuIEFydW4gQmFsYWppLiIsIm5hbWUiOiJDaGFpdGF
ueWEgQXJvcmEiLCJpYXQiOjE1MTYyMzkwMjIsImV4cCI6MTY5NDQ0MzUzMn0.9JzGe21
KAjIdRIlvHozMKE8J8_0o836G9L9oEN4QwHqrIO0F9-XIwLW4pUx_BL9w
Please enter the secret key: c3a82562db6e313016fbc167e1c7cfb0d3ced03
4cfaed84f891179ee7336528d
Traceback (most recent call last):
  File "c:\Users\om\Desktop\sem5\FCS\2021033_q3.py", line 63, in <mo
dule>
    print(verifyJwt(token, secret))
          ^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\om\Desktop\sem5\FCS\2021033_q3.py", line 50, in ver
ifyJwt
    raise ValueError("Token has expired")
ValueError: Token has expired
PS C:\Users\om>
```

This output is consistent as the expiry date set for this JWT has gone and therefore its throwing error.

For the following test case:



Ouput is correct as this is what i precisely encoded.

```
ValueError: Toke& C:/Users/om/AppData/Local/Programs/Python/Python31
1/python.exe c:/Users/om/Desktop/sem5/FCS/2021033_q3.py
Please enter the token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWI
iOiJUaGlzIGlzIGZvciBteSBhc3NpZ25tbmV0IGF0IElJSVQtRCwgZm9yIGEgc2VjdXJ
pdHkgY291cnNlIHVuZGVyIFByb2YuIEFydW4gQmFsYWppLiIsIm5hbWUiOiJDaGFpdGF
ueWEgQXJvcmEiLCJpYXQiOjE2OTQ0NDM1MzIsImV4cCI6MTcwMjAyOTU3Mn0.tb1-AaM
Gnw2paC74RX5rmZEcB9Ia-Rvt0jmoWfyUAjg
Please enter the secret key: c3a82562db6e313016fbc167e1c7cfb0d3ced03
4cfaed84f891179ee7336528d
{'sub': 'This is for my assignmnet at IIIT-D, for a security course
under Prof. Arun Balaji.', 'name': 'Chaitanya Arora', 'iat': 1694443
532, 'exp': 1702029572}
1694450080.9661908
PS C:\Users\om>
```

Following is the test case in which I entered an invalid test case and see the output for that:

```
PS C:\Users\om> & C:/Users/om/AppData/Local/Programs/Python/Python31
1/python.exe c:/Users/om/Desktop/sem5/FCS/2021033_q3.py
Please enter the token: eyJhbGciOiJIUzM4NCIsInR5cCI6IkpXVCJ9.eyJzdWI
iOiJUaGlzIGlzIGZvciBteSBhc3NpZ25tbmV0IGF0IElJSVQtRCwgZm9yIGEgc2VjdXJ
pdHkgY291cnNlIHVuZGVyIFByb2YuIEFydW4gQmFsYWppLiIsIm5hbWUiOiJDaGFpdGF
ueWEgQXJvcmEiLCJpYXQiOjE1MTYyMzkwMjIsImV4cCI6MTY5NDQ0MzUzMn0
Please enter the secret key: c3a82562db6e313016fbc167e1c7cfb0d3ced03
4cfaed84f891179ee7336528d
Traceback (most recent call last):
  File "c:\Users\om\Desktop\sem5\FCS\2021033_q3.py", line 53, in <mo
dule>
    print(verifyJwt(token, secret))
          ^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\om\Desktop\sem5\FCS\2021033_q3.py", line 15, in ver
ifyJwt
    raise ValueError("Token must have exactly 3 parts.")
ValueError: Token must have exactly 3 parts.
PS C:\Users\om>
```

Partb:

Refer to this sheet and find your corresponding jwt token. Your task is to retrieve the secret used to sign the JWT. Once retrieved, create a new JWT with the same secret, and payload updated with your roll number and emailID. Document and explain your steps. (10 Marks)

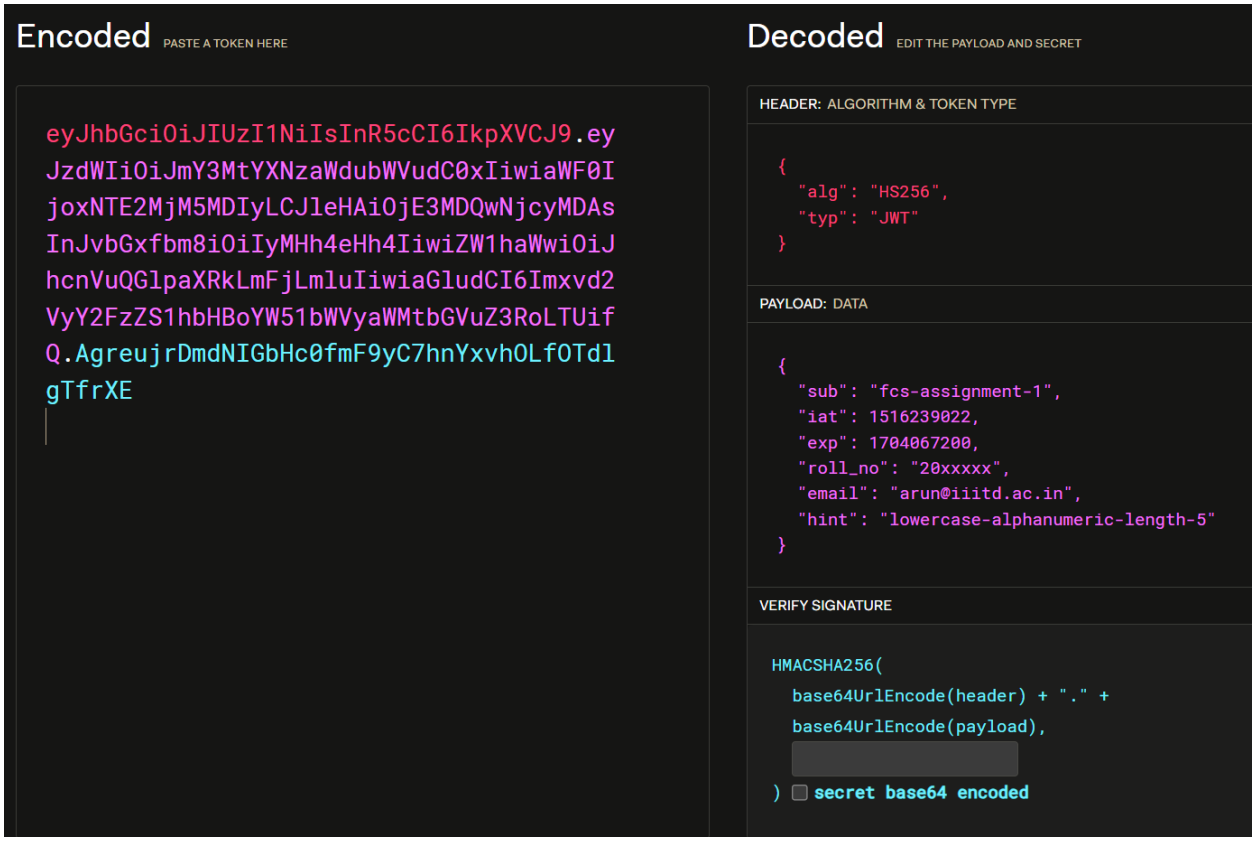
My Token is here:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJmY3MtYXNzaWdubWVudC0xIiwiaWF0IjoxNTE2MjM5MDIyLCJl

eHAiOjE3MDQwNjcyMDAsInJvbGxfbm8iOiIyMHh4eHh4IiwiZW1haWwiOiJhcnVuQGlpaXRkLmFjLmluIiwiaGludCI6Imx

vd2VyY2FzZS1hbHBoYW51bWVyaWMtbGVuZ3RoLTUifQ.HqMrrepIr17JlgVyfAZgl4PdReaBvjmbsVUt9ubQta0

I put this token into jwt debugger and here is an output



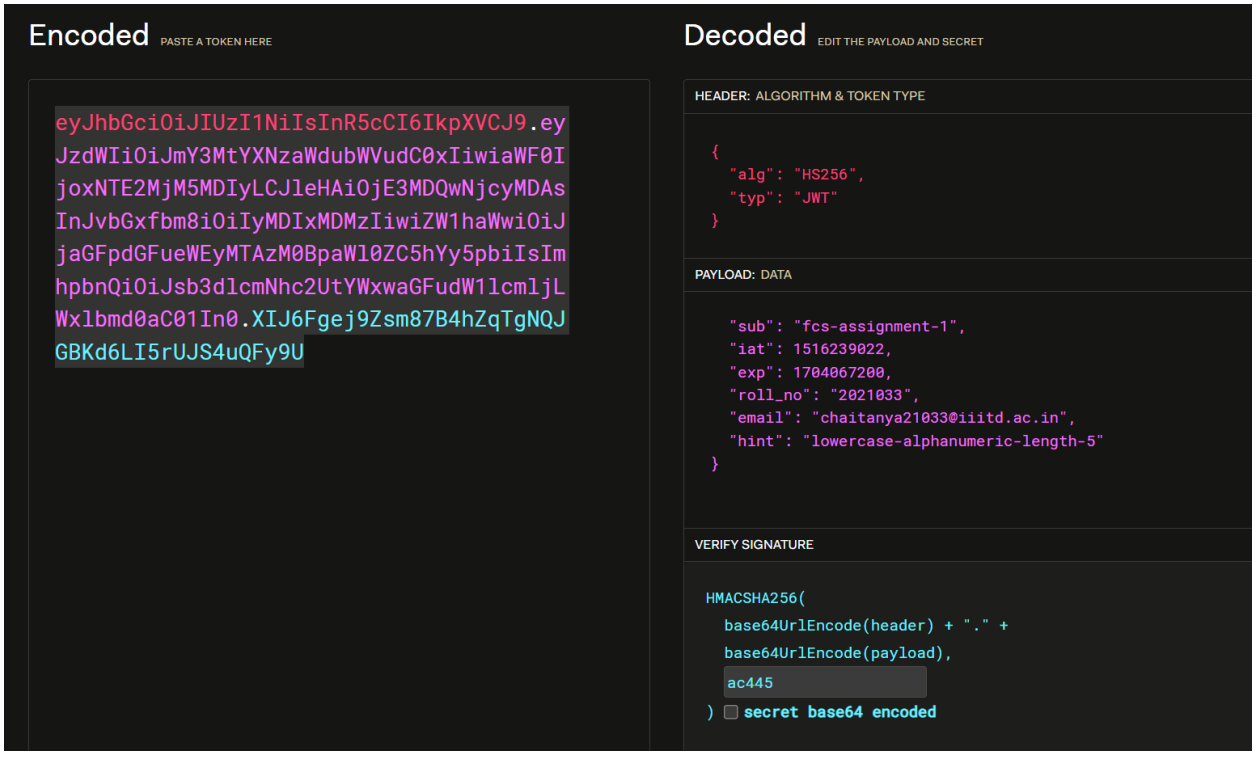
Thanks to the Hint, I wrote the following python code and then ran it for all the possible combinations:

```python
def find_secret(jwt):
    s = "1234567890abcdefghijklmnopqrstuvwxyz"
    for a in s:
        for b in s:
            for c in s:
                for d in s:
                    for e in s:
                        print("Trying: ", a+b+c+d+e)
                        curr = a+b+c+d+e
                        try:
                            verifyJwt(jwt,curr)
                            return jwt, curr
                        except:
                            pass
```

Luckily after hours of running the code I landed upon this secret key: ac445

Here is my JWT with my updated email and roll number:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJmY3MtYXNzaWdubWVudC0xIiwiaWF0IjoxNTE2MjM5MDIyLCJleHAiOjE3MDQwNjcyMDAsInJvbGxfbm8iOiIyMDIxMDMzIiwiZW1haWwiOiJjaGFpdGFueWEyMTAzM0BpaWl0ZC5hYy5pbiIsImhpbnQiOiJsb3dlcmNhc2UtYWxwaGFudW1lcmljLWxlbmd0aC01In0.XIJ6Fgej9Zsm87B4hZqTgNQJGBKd6LI5rUJS4uQFy9U



List down some world use cases of the jwt tokens and suggest some modifications in their architecture so as to improve their security. (5 Marks)

Authorization is possible: When you log into a website, a unique pass is given to you. You must present this pass as identification each time you attempt to view something or make a modification in order to do so.

Can be applied to lessen server side overload: Each visitor receives a unique badge (JWT) containing their details instead of the website remembering them all. When they return, they present the badge as identification.

Passing a sealed letter among pals to communicate information. You recognize and believe in the seal, thus you believe the message.

Improvements
Transient Tokens: Picture tokens as single-use tickets. They are useless if lost and later found. If someone steals a digital token, they won't be able to use it for very long because they can be designed to expire quickly.

Change Tokens Frequently: We can update digital tokens just like some people frequently change their passwords(hopefully). A stolen old token quickly loses all of its value.

Utilize Unique Key Pairs:
Consider a lock that requires two distinct keys—one to lock it and another to unlock it. This approach is used by several digital systems, making unwanted access more difficult.

Private information shouldn't be present in digital tokens. They ought not to reveal much if seen.

Take Back Lost Tokens: Similar to how a club would deactivate a lost membership card, digital systems can render lost or stolen tokens useless and, therefore prevent abuse.