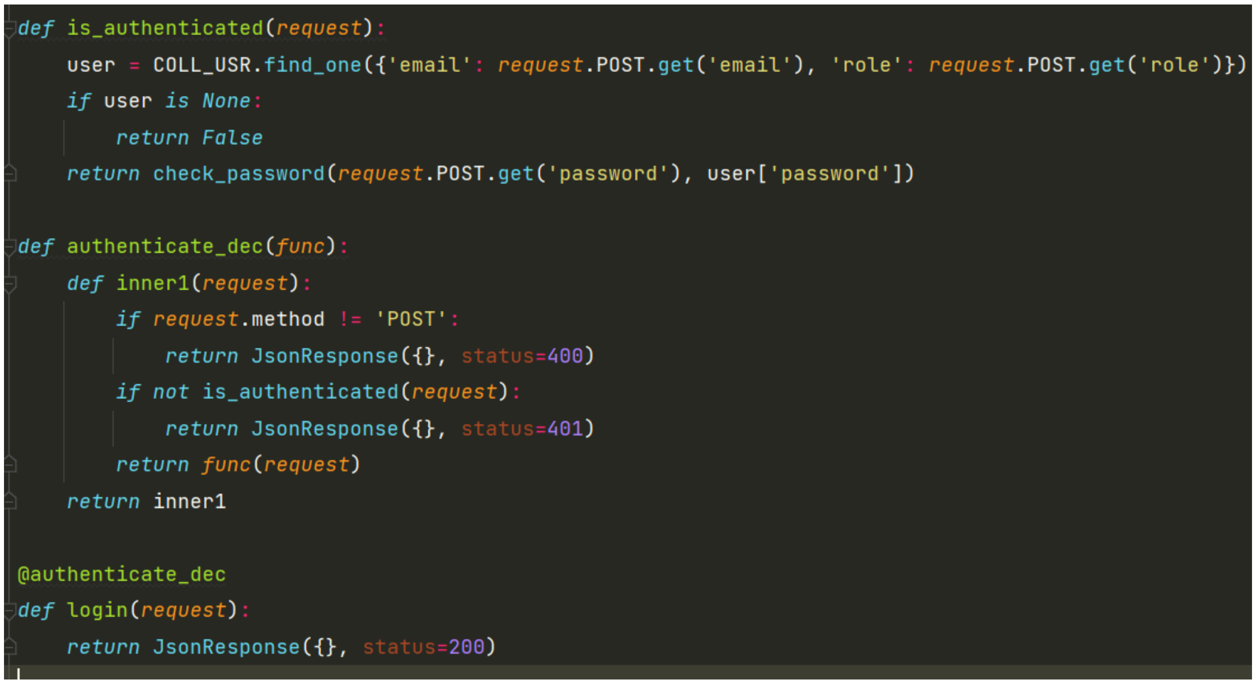
IT314 - Software Engineering
Lab-8

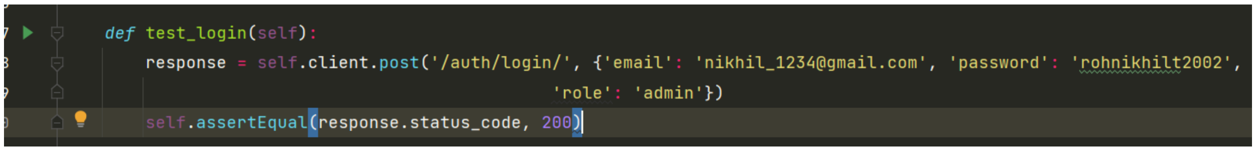
Group 4 Members:

1. 202001002 - Suyash Vyas
2. 202001005 - Mann Joshi
3. 202001015 - Om Vaghani
4. 202001017 - Mihir Patel
5. 202001019 - Ruchi Detroja
6. 202001022 - Meet Shrimali
7. 202001025 - Nilav Shah
8. 202001030 - Ritik Mahyavanshi
9. 202001048 - Dakshveer Singh Chauhan

Test case #1:

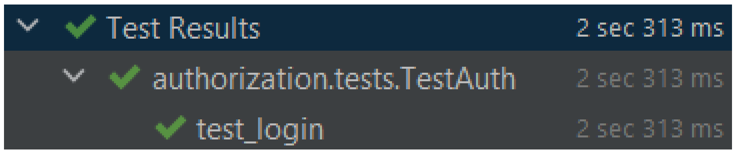


```python
def is_authenticated(request):
    user = COLL_USR.find_one({'email': request.POST.get('email'), 'role': request.POST.get('role')})
    if user is None:
        return False
    return check_password(request.POST.get('password'), user['password'])

def authenticate_dec(func):
    def inner1(request):
        if request.method != 'POST':
            return JsonResponse({}, status=400)
        if not is_authenticated(request):
            return JsonResponse({}, status=401)
        return func(request)
    return inner1

@authenticate_dec
def login(request):
    return JsonResponse({}, status=200)
```



```
    def test_login(self):
        response = self.client.post('/auth/login/', {'email': 'nikhil_1234@gmail.com', 'password': 'rohnikhilt2002',
                                                      'role': 'admin'})
        self.assertEqual(response.status_code, 200)
```

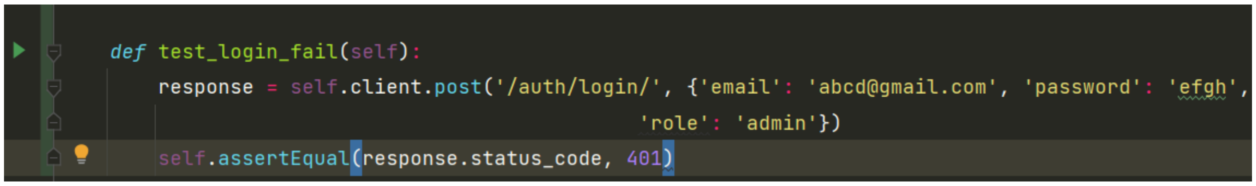
This test case is testing the view “login”. When the user writes their mail, password and role, this view is run by backend and test_login is the function to test this code.
The test will send the post request to the client and get the response and checks its response code which would run perfectly fine as the parameters in the payload are sent correctly.
Output on running the test:



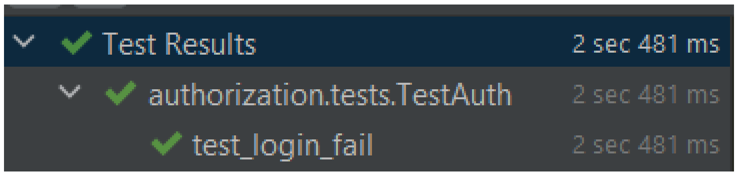
```
✓ Test Results                        2 sec 313 ms
  ✓ authorization.tests.TestAuth      2 sec 313 ms
    ✓ test_login                      2 sec 313 ms
```

Testcase #2:

This test case will be testing the same previous view but with wrong parameters in the payload which would return unauthorized as response.

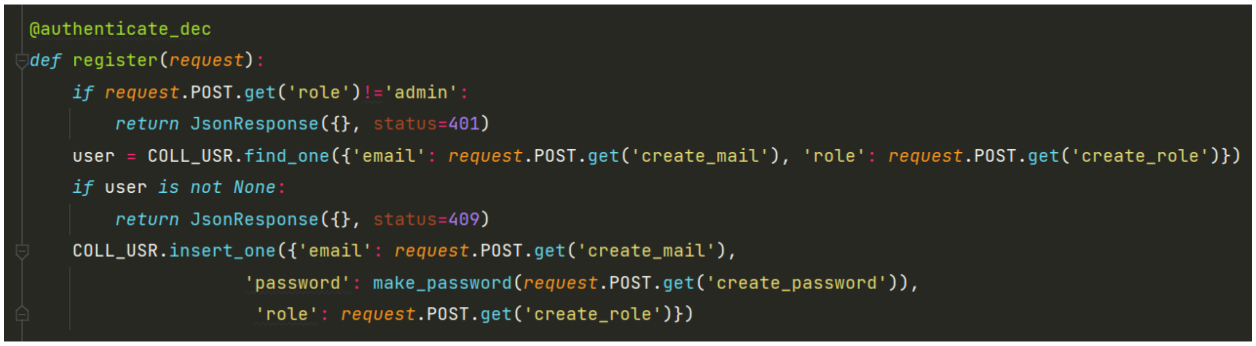


```
    def test_login_fail(self):
        response = self.client.post('/auth/login/', {'email': 'abcd@gmail.com', 'password': 'efgh',
                                                      'role': 'admin'})
        self.assertEqual(response.status_code, 401)
```

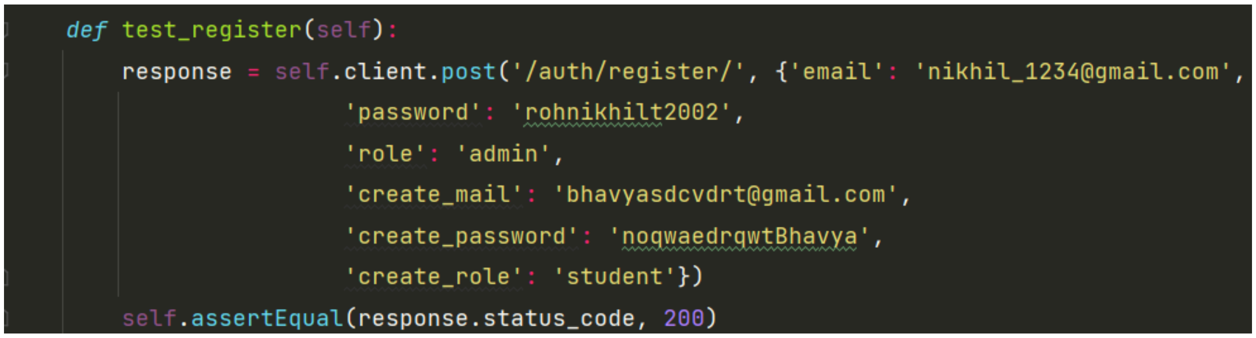


```
✓ Test Results                        2 sec 481 ms
  ✓ authorization.tests.TestAuth      2 sec 481 ms
    ✓ test_login_fail                 2 sec 481 ms
```

Testcase #3:

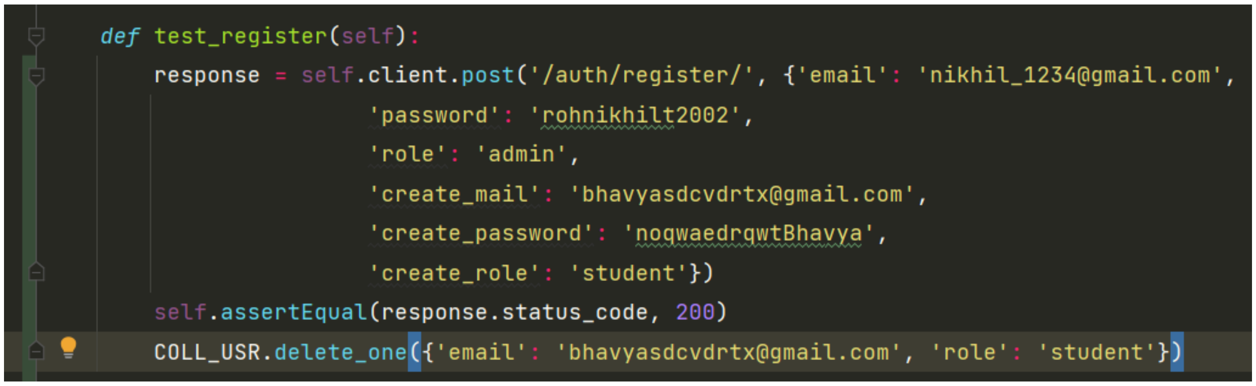


```
@authenticate_dec
def register(request):
    if request.POST.get('role')!='admin':
        return JsonResponse({}, status=401)
    user = COLL_USR.find_one({'email': request.POST.get('create_mail'), 'role': request.POST.get('create_role')})
    if user is not None:
        return JsonResponse({}, status=409)
    COLL_USR.insert_one({'email': request.POST.get('create_mail'),
                'password': make_password(request.POST.get('create_password')),
                 'role': request.POST.get('create_role')})
```



```
    def test_register(self):
        response = self.client.post('/auth/register/', {'email': 'nikhil_1234@gmail.com',
                            'password': 'rohnikhilt2002',
                            'role': 'admin',
                            'create_mail': 'bhavyasdcvdrt@gmail.com',
                            'create_password': 'noqwaedrqwtBhavya',
                            'create_role': 'student'})
        self.assertEqual(response.status_code, 200)
```

This test correctly returns the 200 code but on executing the same test twice the test will return the code 409 saying that the data already exists in the code. The problem here is that we can't run this test twice and to resolve the error we must delete the data that was inserted by calling the register function

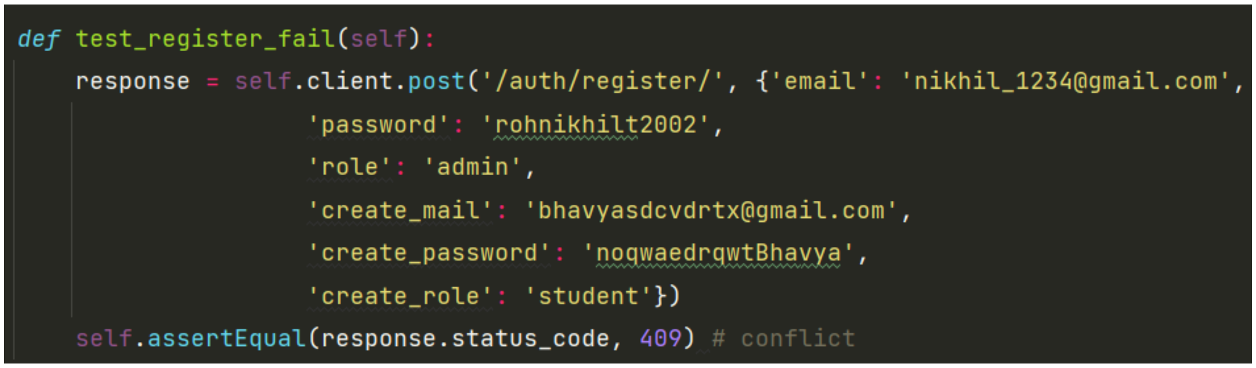


```
    def test_register(self):
        response = self.client.post('/auth/register/', {'email': 'nikhil_1234@gmail.com',
                            'password': 'rohnikhilt2002',
                            'role': 'admin',
                            'create_mail': 'bhavyasdcvdrtx@gmail.com',
                            'create_password': 'noqwaedrqwtBhavya',
                            'create_role': 'student'})
        self.assertEqual(response.status_code, 200)
        COLL_USR.delete_one({'email': 'bhavyasdcvdrtx@gmail.com', 'role': 'student'})
```

The above function is the correct implementation of the test function of register view which deletes the data that was inserted in the database by calling the post method.

This resolves the error and now we can call the same test function multiple times.

Testcase #4



```python
def test_register_fail(self):
    response = self.client.post('/auth/register/', {'email': 'nikhil_1234@gmail.com',
                    'password': 'rohnikhilt2002',
                    'role': 'admin',
                    'create_mail': 'bhavyasdcvdrtx@gmail.com',
                    'create_password': 'noqwaedrqwtBhavya',
                    'create_role': 'student'})
    self.assertEqual(response.status_code, 409) # conflict
```

Above test function tests for the failed registration as the user already exists in the database and view returns the status code 409 which refers to conflict in the database.

Conclusion:-

The test functions send the post request to one of the views, record the response, and do checks if the response is as intended or not.

We have to make sure that we are selecting the right test data to ensure that the code is tested under different scenarios and conditions. This will help identify any errors or issues that may not have been apparent with a limited set of test data.

Testing the failing test cases is equally important as testing for successful scenarios otherwise the code might give errors