11/18/23, 8:23 PM tests.py

backend\api\tests.py

```
from django.test import TestCase
 1
 2
    from django.urls import reverse
    from rest framework import status
 3
 4
    from rest framework.test import APIClient
 5
    from rest framework.authtoken.models import Token
    from .models import User, PreMedicalForm, BlogPost, Comment
 6
 7
    from .serializers import UserSerializer, CommentSerializer
    from .decorators import role required
 8
 9
    from django.http import HttpResponseForbidden
10
11
    class APITests(TestCase):
        def setUp(self):
12
13
            self.client = APIClient()
14
            self.user = User.objects.create user(username='testuser', password='testpassword',
    role='patient')
            self.token = Token.objects.create(user=self.user)
15
            self.pre medical form = PreMedicalForm.objects.create(patient=self.user, symptoms='
16
    Test symptoms')
            self.blog_post = BlogPost.objects.create(title='Test Blog Post', author=self.user,
17
    content='Test content', pre_medical_form=self.pre_medical_form)
            self.comment = Comment.objects.create(content='Test Comment', author=self.user,
18
    blog_post=self.blog_post, commenter_role='patient')
19
20
        def test index(self):
21
            url = reverse('index')
            self.client.credentials(HTTP AUTHORIZATION=f'Token {self.token.key}')
22
            response = self.client.get(url)
23
            self.assertEqual(response.status code, status.HTTP 200 OK)
24
25
        def test register(self):
26
27
            url = reverse('register')
28
            data = {'username': 'newuser', 'password': 'newpassword', 'role': 'patient'}
29
            response = self.client.post(url, data)
30
            self.assertEqual(response.status_code, status.HTTP_201_CREATED)
31
32
        def test_user_delete_view(self):
33
            url = reverse('delete-user', kwargs={'pk': self.user.id})
34
            response = self.client.delete(url)
35
            self.assertEqual(response.status code, status.HTTP 204 NO CONTENT)
            self.assertFalse(User.objects.filter(pk=self.user.id).exists())
36
37
        def test delete view(self):
38
            model = Comment
39
            obj = self.comment
40
41
            url = reverse(f'delete-{model. name .lower()}', kwargs={'pk': obj.id})
42
43
            response = self.client.delete(url)
44
            self.assertEqual(response.status code, status.HTTP 204 NO CONTENT)
            self.assertFalse(model.objects.filter(pk=obj.id).exists())
45
46
47
        def test_comment_delete_view(self):
            self.test delete view()
48
49
50
        def test login view(self):
```

```
51
             url = reverse('login_view')
             data = {'username': 'testuser', 'password': 'testpassword'}
52
53
             response = self.client.post(url, data)
54
             self.assertEqual(response.status code, status.HTTP 200 OK)
55
56
         def test_create_blog_post(self):
57
             url = reverse('create blog post')
             data = {'title': 'New Blog Post', 'content': 'Test content'}
58
             self.client.credentials(HTTP_AUTHORIZATION=f'Token {self.token.key}')
59
             response = self.client.post(url, data, format='json')
60
             self.assertEqual(response.status code, status.HTTP 201 CREATED)
61
62
63
         def test create comment(self):
             url = reverse('create comment')
 64
             data = {'content': 'New Comment', 'blog post': self.blog post.id}
65
 66
             self.client.credentials(HTTP AUTHORIZATION=f'Token {self.token.key}')
             response = self.client.post(url, data, format='json')
67
             self.assertEqual(response.status_code, status.HTTP_201_CREATED)
 68
69
70
         def test_user_details(self):
             url = reverse('user details')
71
72
             self.client.credentials(HTTP AUTHORIZATION=f'Token {self.token.key}')
73
             response = self.client.get(url)
             self.assertEqual(response.status_code, status.HTTP_200_OK)
74
75
             self.assertEqual(response.data['username'], 'testuser')
 76
77
     class SerializersTests(TestCase):
 78
         def setUp(self):
             self.user = User.objects.create user(username='testuser', password='testpassword',
79
     role='patient')
             self.pre medical form = PreMedicalForm.objects.create(patient=self.user, symptoms='
80
    Test symptoms')
             self.blog_post = BlogPost.objects.create(title='Test Blog Post', author=self.user,
81
     content='Test content', pre_medical_form=self.pre_medical_form)
             self.comment = Comment.objects.create(content='Test Comment', author=self.user,
82
    blog_post=self.blog_post, commenter_role='patient')
83
         def test user serializer(self):
84
85
             serializer = UserSerializer(instance=self.user)
             self.assertEqual(serializer.data['username'], 'testuser')
86
87
88
         def test_comment_serializer(self):
89
             serializer = CommentSerializer(instance=self.comment)
             self.assertEqual(serializer.data['content'], 'Test Comment')
90
91
92
93
    class DecoratorsTests(TestCase):
94
         def test_role_required_decorator(self):
             @role required(allowed roles=['admin'])
95
96
             def sample_view(request):
                 return HttpResponseForbidden("Forbidden")
97
98
99
             # Test when the user has the allowed role
             user = User.objects.create_user(username='alloweduser', password='testpassword',
100
     role='admin')
101
             request = self.client.get('/')
102
             request.user = user
```

```
103
             response = sample_view(request)
             self.assertEqual(response.status code, 403)
104
105
             # Test when the user doesn't have the allowed role
106
107
             user = User.objects.create user(username='nonalloweduser', password='testpassword',
     role='patient')
108
             request = self.client.get('/')
109
             request.user = user
             response = sample view(request)
110
111
             self.assertEqual(response.status code, 403)
112
113
     class ModelsTests(TestCase):
114
         def setUp(self):
             self.user = User.objects.create_user(username='testuser', password='testpassword',
115
     role='patient')
             self.pre medical form = PreMedicalForm.objects.create(patient=self.user, symptoms='
116
     Test symptoms')
117
             self.blog post = BlogPost.objects.create(title='Test Blog Post', author=self.user,
     content='Test content', pre_medical_form=self.pre_medical_form)
             self.comment = Comment.objects.create(content='Test Comment', author=self.user,
118
     blog post=self.blog post, commenter role='patient')
119
120
         def test_user_model(self):
121
             self.assertEqual(self.user.username, 'testuser')
122
             self.assertEqual(self.user.role, 'patient')
123
124
         def test_comment_model(self):
125
             self.assertEqual(self.comment.content, 'Test Comment')
126
127
128
```