

backend\api\tests.py

```
1 from django.test import TestCase
2 from django.urls import reverse
3 from rest_framework import status
4 from rest_framework.test import APIClient
5 from rest_framework.authtoken.models import Token
6 from .models import User, PreMedicalForm, BlogPost, Comment
7 from .serializers import UserSerializer, CommentSerializer
8 from .decorators import role_required
9 from django.http import HttpResponseForbidden
10
11 class APITests(TestCase):
12     def setUp(self):
13         self.client = APIClient()
14         self.user = User.objects.create_user(username='testuser', password='testpassword',
15 role='patient')
16         self.token = Token.objects.create(user=self.user)
17         self.pre_medical_form = PreMedicalForm.objects.create(patient=self.user, symptoms='
Test symptoms')
18         self.blog_post = BlogPost.objects.create(title='Test Blog Post', author=self.user,
19 content='Test content', pre_medical_form=self.pre_medical_form)
20         self.comment = Comment.objects.create(content='Test Comment', author=self.user,
21 blog_post=self.blog_post, commenter_role='patient')
22
23     def test_index(self):
24         url = reverse('index')
25         self.client.credentials(HTTP_AUTHORIZATION=f'Token {self.token.key}')
26         response = self.client.get(url)
27         self.assertEqual(response.status_code, status.HTTP_200_OK)
28
29     def test_register(self):
30         url = reverse('register')
31         data = {'username': 'newuser', 'password': 'newpassword', 'role': 'patient'}
32         response = self.client.post(url, data)
33         self.assertEqual(response.status_code, status.HTTP_201_CREATED)
34
35     def test_user_delete_view(self):
36         url = reverse('delete-user', kwargs={'pk': self.user.id})
37         response = self.client.delete(url)
38         self.assertEqual(response.status_code, status.HTTP_204_NO_CONTENT)
39         self.assertFalse(User.objects.filter(pk=self.user.id).exists())
40
41     def test_delete_view(self):
42         model = Comment
43         obj = self.comment
44
45         url = reverse(f'delete-{model.__name__.lower()}', kwargs={'pk': obj.id})
46         response = self.client.delete(url)
47         self.assertEqual(response.status_code, status.HTTP_204_NO_CONTENT)
48         self.assertFalse(model.objects.filter(pk=obj.id).exists())
49
50     def test_comment_delete_view(self):
51         self.test_delete_view()
52
53     def test_login_view(self):
```

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

```
103         response = sample_view(request)
104         self.assertEqual(response.status_code, 403)
105
106         # Test when the user doesn't have the allowed role
107         user = User.objects.create_user(username='nonalloweduser', password='testpassword',
role='patient')
108         request = self.client.get('/')
109         request.user = user
110         response = sample_view(request)
111         self.assertEqual(response.status_code, 403)
112
113     class ModelsTests(TestCase):
114         def setUp(self):
115             self.user = User.objects.create_user(username='testuser', password='testpassword',
role='patient')
116             self.pre_medical_form = PreMedicalForm.objects.create(patient=self.user, symptoms='
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)
118             self.comment = Comment.objects.create(content='Test Comment', author=self.user,
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
```