```python
from rest_framework.test import APITestCase
from django.urls import reverse
from rest_framework import status
from django.contrib.auth import get_user_model
from rest_framework_simplejwt.tokens import RefreshToken

from users.models import HealthData
from users.models import UserProfile
from llm.views import LlmPromptView
from unittest.mock import patch


User = get_user_model()

class CreatePromptTest(APITestCase):
    def setUp(self):
        # Create a user and associated UserProfile
        self.user = User.objects.create_user(
            username='testuser',
            email='testuser@example.com',
            password='testpassword'
        )
        profile, _ = UserProfile.objects.get_or_create(user=self.user)

        # Use update_or_create, not create, to avoid IntegrityError!!
        self.health_data, created = HealthData.objects.update_or_create(
            profile=profile,
            defaults={
                'dob': '1990-01-01',
                'gender': 'Male',
                'height': 1.75,
                'weight': 75,
                'favourite_workout_type': 'Cardio',
                'workout_experience': 'Intermediate',
                'fitness_goal': 'Build muscle',
                'injuries': 'None',
                'other_considerations': 'None'
            }
        )
        # Confirm data has saved correctly
        self.view = LlmPromptView()

    def test_create_prompt(self):
        # Simulated data
        ser_obj_data = {
            'length': 60,
            'difficulty': 'Easy',
            'workout_type': 'Weights',
            'target_area': 'Chest',
            'equipment_access': 'Full Gym'
        }
        prompt_text = self.view.createPrompt(ser_obj_data)
        # print("[TEST] Final prompt text:\n", prompt_text)
        self.assertIn("length: 60", prompt_text)
        self.assertIn("difficulty: Easy", prompt_text)
        self.assertIn("workout_type: Weights", prompt_text)
        self.assertIn("target_area: Chest", prompt_text)
        self.assertIn("equipment_access: Full Gym", prompt_text)

class CreateWorkoutTest(APITestCase):
    @classmethod
    def setUpTestData(cls):
        cls.user = User.objects.create_user(
            username='testuser',
            email='testuser@example.com',
            password='testpassword',
            first_name='Test',
            last_name='User'
        )

        profile, _ = UserProfile.objects.get_or_create(user=cls.user)
        cls.health_data, created = HealthData.objects.update_or_create(
            profile=profile,
            defaults={
                'dob': '1990-01-01',
                'gender': 'Male',
                'height': 1.75,
                'weight': 75,
                'favourite_workout_type': 'Cardio',
                'workout_experience': 'Intermediate',
                'fitness_goal': 'Build muscle',
                'injuries': 'None',
                'other_considerations': 'None'
            }
        )
        cls.refresh_token = RefreshToken.for_user(cls.user)
        cls.access_token = str(cls.refresh_token.access_token)

    @patch('workout.views.LlmConnection.requestWorkout')
    def test_post_workout(self, mock_requestWorkout):
        # Mock the LLM response
        mock_requestWorkout.return_value = 'Sample LLM Response'

        # Verify healthdata in the test
        health_data =
HealthData.objects.filter(profile__user=self.user).first()
        if health_data:
            # print("[DEBUG] Retrieved HealthData in test_post_workout:")
            for field in HealthData._meta.fields:
                name = field.name
                val = getattr(health_data, name, None)
                # print(f"[DEBUG] {name}: {val}")
        else:
            print("[ERROR] HealthData not found in test_post_workout")

        # Send request
        self.client.credentials(HTTP_AUTHORIZATION='Bearer ' +
self.access_token)
        url = reverse('create-workout')
        data = {
            'length': 60,
            'difficulty': 'Easy',
            'workout_type': 'Weights',
            'target_area': 'Chest',
            'equipment_access': 'Full Gym'
        }
        response = self.client.post(url, data, format='json')
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        mock_requestWorkout.assert_called_once()
```