Copilot

Certainly! Below is an example of how you might set up your `urls.py` and serializers for the sports center booking application:

`urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('facilities/', views.FacilityList.as_view(), name='facility_list'),
    path('bookings/', views.BookingList.as_view(), name='booking_list'),
    path('bookings/create/', views.CreateBooking.as_view(), name='create_booking'),
    path('bookings/<int:pk>/', views.BookingDetail.as_view(), name='booking_detail'),
    path('bookings/<int:pk>/cancel/', views.CancelBooking.as_view(), name='cancel_booking'),
]
```

Serializers:

```
from rest_framework import serializers
from .models import SportsFacility, BookingSlot, Booking

# Serializer for Sports Facility
class SportsFacilitySerializer(serializers.ModelSerializer):
    class Meta:
        model = SportsFacility
        fields = '__all__'

# Serializer for Booking Slot
class BookingSlotSerializer(serializers.ModelSerializer):
    class Meta:
        model = BookingSlot
        fields = '__all__'

# Serializer for Booking
class BookingSerializer(serializers.ModelSerializer):
    slot = BookingSlotSerializer(read_only=True)

    class Meta:
        model = Booking
        fields = ['id', 'user', 'slot', 'booked_on']
```

In the `urls.py` file, you define the URL patterns that correspond to the views for listing facilities, listing bookings, creating a booking, viewing a specific booking, and canceling a booking.

The serializers are used to convert querysets and model instances to JSON so that the data can be rendered into a format that can be easily consumed by your React Native frontend. The `SportsFacilitySerializer` will handle the serialization of the sports facility data, the `BookingSlotSerializer` will handle the serialization of the booking slots, and the `BookingSerializer` will include nested serialization for the booking slots associated with each booking.

Remember to create the corresponding views in your `views.py` file to handle the logic for each endpoint. You'll need to use Django REST Framework's generics or viewsets to define the behavior of each view.

This is a high-level example, and you'll need to customize it to fit the specific requirements of your application. Make sure to test your endpoints thoroughly to ensure they work as expected. Good luck with your project!