Copilot

For your sports center booking application that includes swimming, badminton, and tennis, you can extend the Django models to accommodate these activities. Here's an example of how you might structure your models to handle multiple sports:

```python
from django.db import models
from django.contrib.auth.models import User

# Base model for a Sports Facility
class SportsFacility(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
    open_time = models.TimeField()
    close_time = models.TimeField()

    class Meta:
        abstract = True

# Model for Swimming Pool with specific attributes
class SwimmingPool(SportsFacility):
    lane_count = models.IntegerField()

# Model for Badminton Court
class BadmintonCourt(SportsFacility):
    court_number = models.IntegerField(unique=True)

# Model for Tennis Court
class TennisCourt(SportsFacility):
    surface_type = models.CharField(max_length=50)

# Model for Booking Slot
class BookingSlot(models.Model):
    start_time = models.DateTimeField()
    end_time = models.DateTimeField()
    facility = models.ForeignKey(SportsFacility, on_delete=models.CASCADE, related_name='booking_slots')

# Model for Booking
class Booking(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    slot = models.ForeignKey(BookingSlot, on_delete=models.CASCADE)
    booked_on = models.DateTimeField(auto_now_add=True)

    def save(self, *args, **kwargs):
        # Additional logic to handle booking conflicts
        super().save(*args, **kwargs)
```

In this structure, `SportsFacility` is an abstract base class that provides common fields for all types of sports facilities. Then, you have specific models for each type of facility (e.g., `SwimmingPool`, `BadmintonCourt`, `TennisCourt`) that inherit from `SportsFacility` and add any specific attributes they need.

The `BookingSlot` model is linked to the `SportsFacility` through a foreign key, which allows you to create slots for any type of facility. The `Booking` model is then used to link a user to a specific `BookingSlot`.

For the endpoints, you can create views that handle actions such as listing available facilities, creating bookings for specific facilities, and managing user bookings. You can use Django REST Framework's viewsets and routers to simplify the creation of these endpoints.

Remember to handle authentication and permissions to ensure that only authorized users can access the booking system. Also, consider adding logic to prevent double bookings and to handle cancellations and changes to bookings.

This is a simplified example, and you’ll need to adapt it to fit the specific requirements of your application. Good luck with your sports center booking app!