Copilot

Certainly! For your cricket net booking system, you'll need models to represent the nets, the slots, and the bookings. Here's a simplified example of how you might structure your Django models and the logic for handling bookings:

```
from django.db import models
from django.contrib.auth.models import User
from datetime import datetime, timedelta

# Model for the Cricket Net
class CricketNet(models.Model):
    net_number = models.IntegerField(unique=True)
    is_available = models.BooleanField(default=True)

    def __str__(self):
        return f"Cricket Net {self.net_number}"

# Model for the Booking Slot
class BookingSlot(models.Model):
    start_time = models.DateTimeField()
    end_time = models.DateTimeField()
    cricket_net = models.ForeignKey(CricketNet, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.cricket_net} | {self.start_time} - {self.end_time}"

    # Check if the slot is within the booking time range
    @property
    def is_active(self):
        return self.start_time <= datetime.now() <= self.end_time

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

    def __str__(self):
        return f"Booking by {self.user} for {self.slot}"

    # Logic to empty the slot if it's booked
    def save(self, *args, **kwargs):
        if self.slot.is_active:
            self.slot.cricket_net.is_available = False
            self.slot.cricket_net.save()
        super().save(*args, **kwargs)

    # Logic to release the slot after the end time
    def release_slot(self):
        if not self.slot.is_active:
            self.slot.cricket_net.is_available = True
            self.slot.cricket_net.save()
```

For the endpoints, you can create views that handle the following actions:

- **List Available Nets**: An endpoint to list all cricket nets and their availability.
- **Create Booking**: An endpoint to create a new booking for a specific net and time slot.
- **List Bookings**: An endpoint to list all bookings for a user.
- **Cancel Booking**: An endpoint to cancel a booking and release the net.

Here's an example of how you might define your API endpoints in Django:

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

urlpatterns = [
    path('nets/', views.list_nets, name='list_nets'),
    path('book/', views.create_booking, name='create_booking'),
    path('my_bookings/', views.list_my_bookings, name='list_my_bookings'),
    path('cancel_booking/<int:booking_id>/', views.cancel_booking, name='cancel_booking'),
]
```

In your views, you'll need to implement the logic to handle these requests, such as checking for available slots, creating bookings, and cancelling them. Remember to include authentication and permissions to ensure that only authorized users can create and manage bookings.

This is a high-level overview, and you'll need to adapt it to fit the specific requirements of your application. Good luck with your project!