let's start by setting up the Django project and creating the necessary apps for vendor management, purchase order tracking, and metrics calculation.¶

First, make sure you have Django installed. If not, you can install it using pip:

In [ ]:
```
pip install django
```

Then, create a new Django project:

In [ ]:
```
django-admin startproject vendor_management_system
```

Next, navigate into the project directory:

In [ ]:
```
cd vendor_management_system
```

Now, let's create the Django apps for vendor management, purchase order tracking, and metrics calculation:

In [ ]:
```
python manage.py startapp vendors
python manage.py startapp purchase_orders
python manage.py startapp metrics
```

After creating the apps, let's define the models and API endpoints as per the assignment requirements.

In 'vendors/models.py':

In [ ]:
```
from django.db import models

class Vendor(models.Model):
    name = models.CharField(max_length=100)
    contact_details = models.TextField()
    address = models.TextField()
    vendor_code = models.CharField(max_length=50, unique=True)
    on_time_delivery_rate = models.FloatField(default=0)
    quality_rating_avg = models.FloatField(default=0)
    average_response_time = models.FloatField(default=0)
    fulfillment_rate = models.FloatField(default=0)

    def __str__(self):
        return self.name
```

In 'purchase_orders/models.py':

In [ ]:
```
from django.db import models
from vendors.models import Vendor

class PurchaseOrder(models.Model):
    po_number = models.CharField(max_length=100, unique=True)
    vendor = models.ForeignKey(Vendor, on_delete=models.CASCADE)
    order_date = models.DateTimeField()
    delivery_date = models.DateTimeField()
    items = models.JSONField()
    quantity = models.IntegerField()
    status = models.CharField(max_length=50)
    quality_rating = models.FloatField(null=True)
    issue_date = models.DateTimeField()
    acknowledgment_date = models.DateTimeField(null=True)

    def __str__(self):
        return self.po_number
```

In 'metrics/models.py':

In [ ]:
```
from django.db import models
from vendors.models import Vendor

class HistoricalPerformance(models.Model):
    vendor = models.ForeignKey(Vendor, on_delete=models.CASCADE)
    date = models.DateTimeField()
    on_time_delivery_rate = models.FloatField()
    quality_rating_avg = models.FloatField()
    average_response_time = models.FloatField()
    fulfillment_rate = models.FloatField()

    def __str__(self):
        return f"{self.vendor} - {self.date}"
```

These are the basic models. Next, we need to create API endpoints using Django REST Framework.

In 'vendors/api/views.py':

In [ ]:
```
from rest_framework import generics
from .models import Vendor
from .serializers import VendorSerializer

class VendorListCreate(generics.ListCreateAPIView):
    queryset = Vendor.objects.all()
    serializer_class = VendorSerializer

class VendorRetrieveUpdateDestroy(generics.RetrieveUpdateDestroyAPIView):
    queryset = Vendor.objects.all()
    serializer_class = VendorSerializer
```

In 'purchase_orders/api/views.py':

In [ ]:
```
from rest_framework import generics
from .models import PurchaseOrder
from .serializers import PurchaseOrderSerializer

class PurchaseOrderListCreate(generics.ListCreateAPIView):
    queryset = PurchaseOrder.objects.all()
    serializer_class = PurchaseOrderSerializer

class PurchaseOrderRetrieveUpdateDestroy(generics.RetrieveUpdateDestroyAPIView):
    queryset = PurchaseOrder.objects.all()
    serializer_class = PurchaseOrderSerializer
```

In 'metrics/api/views.py':

In [ ]:
```
from rest_framework import generics
from vendors.models import Vendor
from .serializers import VendorPerformanceSerializer

class VendorPerformanceRetrieve(generics.RetrieveAPIView):
    queryset = Vendor.objects.all()
    serializer_class = VendorPerformanceSerializer
```

These views handle CRUD operations for vendors, purchase orders, and retrieving vendor performance metrics.

In the respective api/serializers.py files, you need to define serializers for each model.

After setting up the models, views, and serializers, don't forget to wire up the URLs in urls.py of each app and the project's main urls.py.

This code provides a basic structure for the Django project and apps according to the assignment requirements. You'll need to fill in the details for serializers, authentication, tests, and additional functionalities as per the assignment guidelines.

In [ ]:
```
```