Task 1: JWT-based Authentication

Use Django's built-in authentication system to handle user registration and login.

Upon successful login, generate a JWT (JSON Web Token) for the user.

Use a third-party library like djangorestframework-simplejwt to handle JWT-based authentication and authorization.

Task 2: Create a Post API for Scraping and Saving Data

Create a Django model to represent the data you want to save, including fields like URL, Title, Price, Description, Reviews, Ratings, and Media Count.

Create a serializer class that corresponds to your model to handle the input data.

Implement a class-based view (e.g., using Django's APIView class) for handling the POST request containing the Flipkart URL.

Inside the view, use a library like BeautifulSoup to scrape the required fields from the provided URL.

Validate the logged-in user and save the scraped data along with the user information to the PostgreSQL database using the model and serializer.

Task 3: Check URL Existence and Ownership

Implement a new class-based view for checking the existence of the URL and its ownership by the logged-in user.

Retrieve the user's URLs from the database based on the provided URL.

If the URL exists and belongs to the logged-in user, return the data.

If the URL is not present or does not belong to the user, return an error message.

Remember that this is a high-level overview, and you will need to implement the details and handle edge cases. Also, ensure you follow proper security practices, such as input validation, error handling, and protecting sensitive data.

rough example of how the Django code might look for Task 2:

# models.py

from django.db import models

from django.contrib.auth.models import User

class Product(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

url = models.URLField()

title = models.CharField(max\_length=255)

price = models.DecimalField(max\_digits=10, decimal\_places=2)

description = models.TextField(blank=True)

reviews\_count = models.IntegerField()

ratings = models.FloatField()

media\_count = models.IntegerField()

# serializers.py

from rest\_framework import serializers

from .models import Product

class ProductSerializer(serializers.ModelSerializer):

class Meta:

model = Product

fields = '\_\_all\_\_'

# views.py

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework import status

from .models import Product

from .serializers import ProductSerializer

class ProductCreateView(APIView):

def post(self, request):

serializer = ProductSerializer(data=request.data)

if serializer.is\_valid():

# Perform web scraping here and populate the serializer data

# You can retrieve the logged-in user using request.user

serializer.save(user=request.user)

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)