Advanced Django Assignment: Distributed E-commerce Platform

Problem Statement:

You are tasked with building a comprehensive distributed e-commerce platform using Django REST Framework. The application will include sophisticated models, advanced JWT authentication, and role-based access control, with intricate querying capabilities.

Models:

1. **Role**
   - `role_id`: Integer (Primary Key)
   - `role`: String (e.g., "ADMIN", "CONSUMER", "SELLER", "SUPPLIER")
   - `permissions`: JSONField (to define specific permissions for the role)
2. **User** (Extend Django's default User model)
   - `user_id`: Integer (Primary Key)
   - `username`: String (inherited)
   - `password`: String (inherited)
   - `roles`: ManyToManyField to Role (Users can have multiple roles)
3. **Category**
   - `category_id`: Integer (Primary Key)
   - `category_name`: String (e.g., "Fashion", "Electronics")
   - `parent_category`: ForeignKey to self (to create nested categories)
4. **Product**
   - `product_id`: Integer (Primary Key)
   - `product_name`: String
   - `description`: Text
   - `price`: Float
   - `stock_quantity`: Integer
   - `seller`: ForeignKey to User (User with "SELLER" role)
   - `category`: ForeignKey to Category
5. **Cart**
   - `cart_id`: Integer (Primary Key)
   - `user`: ForeignKey to User (User with "CONSUMER" role)
   - `total_amount`: Float (calculated from CartProduct entries)
6. **CartProduct**
   - `cart_product_id`: Integer (Primary Key)
   - `cart`: ForeignKey to Cart
   - `product`: ForeignKey to Product
   - `quantity`: Integer
7. **Order**
   - `order_id`: Integer (Primary Key)
   - `user`: ForeignKey to User (User with "CONSUMER" role)

  - `total_amount`: Float
  - `order_date`: DateTime
  - `status`: String (e.g., "PENDING", "COMPLETED", "CANCELLED")

API Endpoints:

You are required to create the following APIs using JWT authentication:

Public Endpoints (Accessible without authentication):

1. **GET** `/api/public/products/search?keyword=<keyword>`
   - Returns products with the keyword either in `product_name` or `category_name`.
2. **POST** `/api/public/login`
   - Authenticates users (both consumers and sellers) and returns a JWT token.

Request Body:
json
Copy code
```
{"username": "user", "password": "pass"}
```

   -

Consumer Endpoints (Requires JWT token with consumer role):

1. **GET** `/api/auth/consumer/cart`
   - Returns the logged-in consumer's cart and all its products.
2. **POST** `/api/auth/consumer/cart`
   - Adds a product to the cart.

Request Body:
json
Copy code
```
{"product_id": 3, "quantity": 2}
```

   -
3. **PUT** `/api/auth/consumer/cart`
   - Updates the product quantity in the cart. If quantity is 0, it removes the product from the cart.

Request Body:
json
Copy code
```
{"product_id": 3, "quantity": 3}
```

   -
4. **DELETE** `/api/auth/consumer/cart`

- Removes a product from the cart.

Request Body:
json
Copy code
```
{"product_id": 3}
```

-

5. **POST** `/api/auth/consumer/order`
   - Places an order for the products in the cart.

Seller Endpoints (Requires JWT token with seller role):

1. **GET** `/api/auth/seller/products`
   - Returns all products added by the logged-in seller.
2. **POST** `/api/auth/seller/products`
   - Adds a new product to the database.

Request Body:
json
Copy code
```
{"product_name": "New Phone", "description": "Latest model",
"price": 899.99, "category_id": 2, "stock_quantity": 100}
```

-

3. **PUT** `/api/auth/seller/products/<product_id>`
   - Updates the product details.

Request Body:
json
Copy code
```
{"product_name": "Updated Phone", "price": 999.99, "category_id": 2,
"stock_quantity": 50}
```

-

4. **DELETE** `/api/auth/seller/products/<product_id>`
   - Deletes a product. If the product is not owned by the seller, return a 404 error.

Admin Endpoints (Requires JWT token with admin role):

1. **GET** `/api/admin/users`
   - Returns all users and their roles.
2. **GET** `/api/admin/sales-summary`
   - Admins can view total sales of all products per seller and category, aggregated in a single view.

Example Response:
json

Copy code
```
[
  {"seller": "bob", "category": "Electronics", "total_sales":
120000},
  {"seller": "alice", "category": "Fashion", "total_sales": 30000}
]
```

○

JWT Authentication:

- Use JWT for authentication. Assign roles to users and protect specific endpoints based on their roles.
- If an authenticated endpoint is accessed without a JWT, return a 401 error.
- If a consumer tries to access a seller endpoint or vice versa, return a 403 error.

Advanced Challenge:

1. **Nested Categories**: Implement functionality to allow categories to have subcategories, enabling users to filter products based on the entire category hierarchy.
2. **Product Stock Management**: Ensure that when an order is placed, the stock quantity of the product is updated accordingly. If a product's stock is insufficient, return a 400 error when trying to place an order.
3. **Role-Based Dynamic Permissions**: Allow the admin to dynamically manage user roles and permissions. Implement an API endpoint to update permissions for each role.

Submission:

- Submit your code repository on GitHub along with documentation that explains how to set up and test your API, including details on how to manage user roles and permissions.

This assignment requires you to implement advanced features and handle complex authentication and authorization scenarios, making it an excellent way to deepen your knowledge of Django and the Django REST Framework. Good luck!