**#PROJECT:**

**1)SERACH:**

1. The dealer initiates a search request to check the availability of a product at a specific location by specifying the **product name, quantity , location,warehouse\_id,dealer\_id.**
2. The request is sent to the backend server.--> Validates the request parameters and queries the warehouse database to check the product availability.
3. The server sends the availability status back to the application.
4. The application displays the product availability to the dealer.

**2)ADD TO CART:**

1. The dealer initiates a add to cart request.

2.The request is sent to the backend server🡪validates the product information

-Check if the dealer already has a shopping cart

- if not created🡪 Increment quantity if the product is already in the shopping cart.

**3. Buy Function:**

1. After searching, the dealer selects a product and places an order.
2. The order details are sent to the backend server.
3. The backend server validate the order, including inventory updation. And unique order id is generated and stord in databse along with order details.
4. If the product is available, the server initiates the payment process.
5. Once payment is confirmed, the order status is updated, and a confirmation is sent to the dealer.

**4.Order Status Function:**

1. Dealer initiates a request to track the order status.
2. The application server retrieves the current status from the database.
3. The server sends the status back to the application in real-time.
4. The application updates the user interface with the latest order status.

**# models.py**

**from django.db import models**

**class Product(models.Model):** *🡪 The Product model represents the products in the warehouse*

**name = models.CharField(max\_length=255)**

**location = models.CharField(max\_length=255)**

**quantity = models.PositiveIntegerField()**

**class Order(models.Model):🡪** *represents placed orders*

**product = models.ForeignKey(Product, on\_delete=models.CASCADE)**

**quantity = models.PositiveIntegerField()**

**status = models.CharField(max\_length=50, default='Pending')**

**class ShoppingCart(models.Model):***🡪 model represents the items in a dealer's shopping cart.*

**dealer\_id = models.IntegerField()** *# You might want to use a ForeignKey to a Dealer model in a**real-world scenario*

**product = models.ForeignKey(Product, on\_delete=models.CASCADE)**

**quantity = models.PositiveIntegerField()**

**#views.py**

**from django.http import JsonResponse**

**from django.views.decorators.csrf import csrf\_exempt**

**from .models import Product, ShoppingCart**

**@csrf\_exempt**

**def search\_product(request):**

**if request.method == 'POST':**

**product\_name = request.POST.get('product\_name')**

**location = request.POST.get('location')**

**required\_quantity = int(request.POST.get('required\_quantity', 1)) # Default to 1 if not provided**

**try:**

**product = Product.objects.get(name=product\_name, location=location)**

**if product.quantity >= required\_quantity:**

**return JsonResponse({**

**'status': 'success',**

**'message': f'Product in stock. Available quantity: {product.quantity}',**

**'product\_name': product.name,**

**'location': product.location,**

**'quantity\_available': product.quantity,**

**})**

**else:**

**return JsonResponse({**

**'status': 'error',**

**'message': f'Insufficient stock. Available quantity: {product.quantity}',**

**})**

**except Product.DoesNotExist:**

**return JsonResponse({'status': 'error', 'message': 'Product not found.'})**

**# views.py**

**from django.http import JsonResponse**

**from django.views.decorators.csrf import csrf\_exempt**

**from .models import Product, ShoppingCart**

**@csrf\_exempt**

**def add\_to\_cart(request):**

**if request.method == 'POST':**

**product\_name = request.POST.get('product\_name')**

**location = request.POST.get('location')**

**quantity = int(request.POST.get('quantity', 1))** *# Default to 1 if quantity is not provided*

**try:**

**product = Product.objects.get(name=product\_name, location=location)**

*# Check if the dealer already has a shopping cart*

**dealer\_id = 1** *# Replace with actual dealer identification mechanism*

**shopping\_cart, created = ShoppingCart.objects.get\_or\_create(dealer\_id=dealer\_id, product=product)**

**if not created:**

*# Increment quantity if the product is already in the shopping cart*

**shopping\_cart.quantity += quantity**

**shopping\_cart.save()**

**return JsonResponse({'status': 'success', 'message': 'Product added to cart successfully.'})**

**except Product.DoesNotExist:**

**return JsonResponse({'status': 'error', 'message': 'Product not found.'})**

**# views.py**

**from django.http import JsonResponse**

**from django.views.decorators.csrf import csrf\_exempt**

**from django.db import transaction**

**from .models import Product, Order, ShoppingCart**

**@csrf\_exempt**

**@transaction.atomic**

**def place\_order(request):**

**if request.method == 'POST':**

**product\_name = request.POST.get('product\_name')**

**location = request.POST.get('location')**

**required\_quantity = int(request.POST.get('required\_quantity', 1***)) # Default to 1 if not provided*

**try:**

**with transaction.atomic():**

**product = Product.objects.select\_for\_update().get(name=product\_name, location=location)**

**if product.quantity >= required\_quantity:**

**# Deduct quantity from inventory**

**product.quantity -= required\_quantity**

**product.save()**

*# Create an order*

**Order.objects.create(product=product, quantity=required\_quantity)**

**return JsonResponse({'status': 'success', 'message': 'Order placed successfully.'})**

**else:**

**return JsonResponse({'status': 'error', 'message': 'Insufficient stock.'})**

**except Product.DoesNotExist:**

**return JsonResponse({'status': 'error', 'message': 'Product not found.'})**

**@csrf\_exempt**

**def track\_order(request, order\_id):**

**try:**

**order = Order.objects.get(id=order\_id)**

**return JsonResponse({'status': 'success', 'message': f'Order status: {order.status}'})**

**except Order.DoesNotExist:**

**return JsonResponse({'status': 'error', 'message': 'Order not found.'})**