TASK – 1

Build a basic e-commerce site with product listings.

Frontend: HTML, CSS, JavaScript.

Backend: Use Django (Python) or Express.js (Node.js).

Add features like:

● Shopping cart

● Product details page

● Order processing

FRONTEND : INDEX.HTML

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>E-commerce Dashboard</title>

<link rel="stylesheet" href="{% static 'css/style.css' %}">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css">

<style>
* { margin:0; padding:0; box-sizing:border-box; font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; }
body { background:#f8fafc; color:#333; }

/* ---------- Navbar ---------- */
.navbar {
  display:flex; justify-content:space-between; align-items:center;
  padding:1rem 2rem; background: #1e3a8a;
  box-shadow:0 4px 10px rgba(0,0,0,0.1); color:white;
}
.navbar .logo { font-size:1.6rem; font-weight:bold; display:flex; align-items:center; gap:10px;
}
.navbar ul { display:flex; gap:2rem; list-style:none; }
.navbar ul li a { color:white; text-decoration:none; display:flex; gap:6px; transition: all 0.3s
ease; font-weight:500; }
.navbar ul li a:hover { color:#facc15; transform:scale(1.1); }

/* ---------- Header ---------- */
header { text-align:center; padding:3rem 1rem; }
header h1 { font-size:2.2rem; color:#1f2937; }
header span { color:#2563eb; }
header p { color:#6b7280; margin-top:0.5rem; font-size:1.05rem; }

/* ---------- Container ---------- */
.container {
  display:grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap:1.5rem;
  padding:2rem; max-width:1200px; margin:auto;
  min-height: 50vh;
}

/* ---------- Card Styles ---------- */
.card {
  border-radius:18px; padding:2rem;
  min-height:180px; display:flex; flex-direction:column; justify-content:center; text-
align:center;
```

```css
  color:white; position:relative; overflow:hidden;
  transition: all 0.35s ease; cursor:pointer;
  box-shadow: 0 6px 18px rgba(0,0,0,0.1);
}
.card i { font-size:2.3rem; margin-bottom:1rem; transition: transform 0.3s; }
.card h3 { font-size:1.2rem; margin-bottom:0.4rem; }
.card p { font-size:0.9rem; opacity:0.9; }
.card:hover { transform:translateY(-8px); box-shadow:0 12px 28px rgba(0,0,0,0.2); }
.card:hover i { transform: scale(1.2); }

/* ---------- Distinct Gradient Colors ---------- */
.user       { background: linear-gradient(135deg, #6a11cb, #2575fc); }
.category   { background: linear-gradient(135deg, #FF7E5F, #FEB47B); }
.product    { background: linear-gradient(135deg, #6A82FB, #FC5C7D); }
.cart       { background: linear-gradient(135deg, #43CEA2, #185A9D); }
.order      { background: linear-gradient(135deg, #8E2DE2, #4A00E0); }
.review     { background: linear-gradient(135deg, #11998E, #38EF7D); }
.shipping   { background: linear-gradient(135deg, #F7971E, #FFD200); }

/* ---------- Product List ---------- */
#product-list {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 1.2rem;
  padding: 2rem;
  max-width: 1200px;
  margin: auto;
}
.product-card {
  background:white;
  border-radius:15px;
  padding:1.2rem;
  box-shadow:0 4px 10px rgba(0,0,0,0.1);
  transition:0.3s;
}
.product-card:hover { transform:translateY(-5px); }
.product-card h4 { color:#1e3a8a; margin-bottom:0.5rem; }
.product-card p { font-size:0.9rem; color:#555; }
</style>
</head>
<body>

<!-- Navbar -->
<nav class="navbar">
  <div class="logo"><i class="fa-solid fa-store"></i> MyShop</div>
  <ul>
    <li><a href="#"><i class="fa fa-home"></i> Home</a></li>
    <li><a href="#"><i class="fa fa-user"></i> Profile</a></li>
    <li><a href="#"><i class="fa fa-cog"></i> Settings</a></li>
  </ul>
</nav>

<!-- Header -->
<header>
  <h1>Welcome to <span>E-commerce Dashboard</span></h1>
  <p>Manage your store efficiently with quick access below</p>
</header>

<!-- Dashboard Cards -->
<div class="container">
```

```
  <a href="{% url 'users' %}"><div class="card user"><i class="fa fa-
users"></i><h3>Users</h3><p>Manage all users</p></div></a>

  <a href="{% url 'categories' %}"><div class="card category"><i class="fa fa-
list"></i><h3>Categories</h3><p>Organize product categories</p></div></a>

  <a href="{% url 'products' %}"><div class="card product"><i class="fa fa-
box"></i><h3>Products</h3><p>Manage product details</p></div></a>

  <a href="{% url 'carts' %}"><div class="card cart"><i class="fa fa-shopping-
cart"></i><h3>Carts</h3><p>Customer shopping carts</p></div></a>

  <a href="{% url 'orders' %}"><div class="card order"><i class="fa fa-clipboard-
list"></i><h3>Orders</h3><p>Track customer orders</p></div></a>

  <a href="{% url 'reviews' %}"><div class="card review"><i class="fa fa-
star"></i><h3>Reviews</h3><p>Customer feedback</p></div></a>

  <a href="{% url 'shipping' %}"><div class="card shipping"><i class="fa fa-
truck"></i><h3>Shipping</h3><p>Manage shipping addresses</p></div></a>

</div>


<!-- Product List Section -->

<section>

  <h2 style="text-align:center; margin-top:2rem; color:#1f2937;">Available Products</h2>

  <div id="product-list">

    <!-- JS will insert products here -->

  </div>

</section>


<!-- Footer -->

<footer>

  <p>© 2025 E-commerce Platform | Built with <i class="fa fa-heart"></i> by Nathiya</p>

</footer>


<script>

// Fetch products from API and display dynamically

fetch("http://127.0.0.1:8000/products/")

  .then(res => res.json())
  .then(data => {
    let container = document.getElementById("product-list");
    container.innerHTML = "";
    data.forEach(p => {
      container.innerHTML += `
        <div class="product-card">
          <h4>${p.name}</h4>
          <p>${p.description}</p>
          <p><strong>Price:</strong> ₹${p.price}</p>
          <p><strong>Stock:</strong> ${p.stock}</p>
        </div>
      `;
    });
  })
  .catch(err => console.error("Error fetching products:", err));
</script>

<script src="{% static 'js/main.js' %}"></script>
</body>
</html>
```

FRONTEND URLS_PY

```
# frontend/frontend_urls.py
from django.urls import path
from . import frontend_views as views

urlpatterns = [
    path("", views.index, name="index"),
    path("products/", views.products, name="products"),
    path("categories/", views.categories, name="categories"),
    path("carts/", views.carts, name="carts"),
    path("orders/", views.orders, name="orders"),
    path("reviews/", views.reviews, name="reviews"),
    path("users/", views.users, name="users"),
    path("shipping/", views.shipping, name="shipping"),
]
```

FRONTEND VIWS.PY

```
# frontend/frontend_views.py
from django.shortcuts import render


def index(request):
    return render(request, "index.html")


def products(request):
    return render(request, "products.html")


def categories(request):
    return render(request, "categories.html")


def carts(request):
    return render(request, "carts.html")


def orders(request):
    return render(request, "orders.html")


def reviews(request):
    return render(request, "reviews.html")


def users(request):
    return render(request, "users.html")
```

```
def shipping(request):
    return render(request, "shipping.html")
```

BACKEND :DJANGO (MODELS.PY)

```
from django.db import models
from django.contrib.auth.models import AbstractUser

# ---------------- USER ----------------
class User(AbstractUser):
    phone_number = models.CharField(max_length=15, blank=True, null=True)

    groups = models.ManyToManyField(
        'auth.Group',
        related_name='custom_user_set',
        blank=True
    )
    user_permissions = models.ManyToManyField(
        'auth.Permission',
        related_name='custom_user_set_permissions',
        blank=True
    )

    def _str_(self):
        return self.username

# ---------------- CATEGORY ----------------
class Category(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField(blank=True, null=True)
    active = models.BooleanField(default=True)

    def _str_(self):
        return self.name

# ---------------- PRODUCT ----------------
class Product(models.Model):
    name = models.CharField(max_length=200)
    category = models.ForeignKey(Category, on_delete=models.SET_NULL, null=True)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.PositiveIntegerField(default=0)
    image = models.ImageField(upload_to='products/', blank=True, null=True)

    def _str_(self):
        return self.name

# ---------------- CART ----------------
class Cart(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)

    def _str_(self):
        return f"Cart of {self.user.username}"

# ---------------- ORDER ----------------
class Order(models.Model):
    STATUS_CHOICES = [
        ('PENDING', 'Pending'),
        ('PROCESSING', 'Processing'),
        ('COMPLETED', 'Completed'),
        ('CANCELLED', 'Cancelled')
    ]

    user = models.ForeignKey(User, on_delete=models.CASCADE)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='PENDING')
    total_price = models.DecimalField(max_digits=10, decimal_places=2, default=0)

    def _str_(self):
        return f"Order #{self.id} by {self.user.username}"

# ---------------- REVIEW ----------------
class Review(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, related_name='reviews',
on_delete=models.CASCADE)
    rating = models.PositiveSmallIntegerField(default=5)
    comment = models.TextField(blank=True, null=True)

    def _str_(self):
        return f"{self.user.username} - {self.product.name}"

# ---------------- SHIPPING ----------------
class ShippingAddress(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    order = models.ForeignKey(Order, related_name='shipping',
on_delete=models.CASCADE)
    address_line = models.TextField()
    city = models.CharField(max_length=100)
    postal_code = models.CharField(max_length=20)
    country = models.CharField(max_length=50)

    def _str_(self):
        return f"{self.address_line}, {self.city}"
```

URLS.PY

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

urlpatterns = [
    # USERS
    path('users/', views.get_users),
    path('users/<int:pk>/', views.get_user_by_id),
    path('users/add/', views.post_user),
    path('users/update/<int:pk>/', views.update_user),
    path('users/delete/<int:pk>/', views.delete_user),

    # CATEGORY
    path('categories/', views.get_categories),
    path('categories/<int:pk>/', views.get_category_by_id),
    path('categories/add/', views.post_category),
    path('categories/update/<int:pk>/', views.update_category),
    path('categories/delete/<int:pk>/', views.delete_category),

    # PRODUCTS
    path('products/', views.get_products),
    path('products/<int:pk>/', views.get_product_by_id),
    path('products/add/', views.post_product),
    path('products/update/<int:pk>/', views.update_product),
    path('products/delete/<int:pk>/', views.delete_product),
    # CART
    path('cart/<int:user_id>/', views.get_cart),
    path('cart/add/', views.post_cart),
    path('cart/update/<int:pk>/', views.update_cart),
    path('cart/delete/<int:pk>/', views.delete_cart),

    # ORDERS
    path('orders/', views.get_orders),
    path('orders/<int:pk>/', views.get_order_by_id),
    path('orders/add/', views.post_order),
    path('orders/update/<int:pk>/', views.update_order),
    path('orders/delete/<int:pk>/', views.delete_order),

    # REVIEWS
    path('reviews/<int:pk>/', views.get_review),
    path('reviews/add/', views.post_review),
    path('reviews/update/<int:pk>/', views.update_review),
    path('reviews/delete/<int:pk>/', views.delete_review),

    # SHIPPING
    path('shipping/', views.get_shipping_addresses),
    path('shipping/add/', views.post_shipping_address),
]
```

VIEWS.PY

```
from django.shortcuts import get_object_or_404
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status
from .models import User, Category, Product, Cart, Order, Review, ShippingAddress
from .serializers import (
    UserSerializer, CategorySerializer, ProductSerializer,
    CartSerializer, OrderSerializer, ReviewSerializer, ShippingAddressSerializer
)

# ---------------- USERS ----------------
@api_view(['GET'])
def get_users(request):
    users = User.objects.all()
    serializer = UserSerializer(users, many=True)
    return Response(serializer.data)

@api_view(['GET'])
def get_user_by_id(request, pk):
    user = get_object_or_404(User, pk=pk)
    serializer = UserSerializer(user)
    return Response(serializer.data)

@api_view(['POST'])
def post_user(request):
    serializer = UserSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['PUT'])
def update_user(request, pk):
    user = get_object_or_404(User, pk=pk)
    serializer = UserSerializer(user, data=request.data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_user(request, pk):
    user = get_object_or_404(User, pk=pk)
    user.delete()
    return Response({"message": "User deleted successfully"},
status=status.HTTP_204_NO_CONTENT)

# ---------------- CATEGORY ----------------
@api_view(['GET'])
def get_categories(request):
    categories = Category.objects.all()
    serializer = CategorySerializer(categories, many=True)
    return Response(serializer.data)

@api_view(['GET'])
def get_category_by_id(request, pk):
    category = get_object_or_404(Category, pk=pk)
    serializer = CategorySerializer(category)
    return Response(serializer.data)

@api_view(['POST'])
def post_category(request):
    serializer = CategorySerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


@api_view(['PUT'])
def update_category(request, pk):
    category = get_object_or_404(Category, pk=pk)
    serializer = CategorySerializer(category, data=request.data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


@api_view(['DELETE'])
def delete_category(request, pk):
    category = get_object_or_404(Category, pk=pk)
    category.delete()
    return Response({"message": "Category deleted"},
status=status.HTTP_204_NO_CONTENT)


# ---------------- PRODUCTS ----------------
@api_view(['GET'])
def get_products(request):
    products = Product.objects.all()
    serializer = ProductSerializer(products, many=True)
    return Response(serializer.data)


@api_view(['GET'])
def get_product_by_id(request, pk):
    product = get_object_or_404(Product, pk=pk)
    serializer = ProductSerializer(product)
    return Response(serializer.data)
```

```
@api_view(['POST'])
def post_product(request):
    serializer = ProductSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['PUT'])
def update_product(request, pk):
    product = get_object_or_404(Product, pk=pk)
    serializer = ProductSerializer(product, data=request.data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_product(request, pk):
    product = get_object_or_404(Product, pk=pk)
    product.delete()
    return Response({"message": "Product deleted"},
status=status.HTTP_204_NO_CONTENT)

# ---------------- CART ----------------
@api_view(['GET'])
def get_cart(request, user_id):
    cart, created = Cart.objects.get_or_create(user_id=user_id)
    serializer = CartSerializer(cart)
    return Response(serializer.data)
```

```
@api_view(['POST'])
def post_cart(request):
    serializer = CartSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['PUT'])
def update_cart(request, pk):
    cart = get_object_or_404(Cart, pk=pk)
    serializer = CartSerializer(cart, data=request.data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_cart(request, pk):
    cart = get_object_or_404(Cart, pk=pk)
    cart.delete()
    return Response({"message": "Cart deleted"}, status=status.HTTP_204_NO_CONTENT)

# ---------------- ORDERS ----------------
@api_view(['GET'])
def get_orders(request):
    orders = Order.objects.all()
    serializer = OrderSerializer(orders, many=True)
    return Response(serializer.data)
```

```
@api_view(['GET'])
def get_order_by_id(request, pk):
    order = get_object_or_404(Order, pk=pk)
    serializer = OrderSerializer(order)
    return Response(serializer.data)

@api_view(['POST'])
def post_order(request):
    serializer = OrderSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['PUT'])
def update_order(request, pk):
    order = get_object_or_404(Order, pk=pk)
    serializer = OrderSerializer(order, data=request.data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_order(request, pk):
    order = get_object_or_404(Order, pk=pk)
    order.delete()
    return Response({"message": "Order deleted"}, status=status.HTTP_204_NO_CONTENT)

# ---------------- REVIEWS ----------------
```

```
@api_view(['GET'])
def get_review(request, pk):
    review = get_object_or_404(Review, pk=pk)
    serializer = ReviewSerializer(review)
    return Response(serializer.data)

@api_view(['POST'])
def post_review(request):
    serializer = ReviewSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['PUT'])
def update_review(request, pk):
    review = get_object_or_404(Review, pk=pk)
    serializer = ReviewSerializer(review, data=request.data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_review(request, pk):
    review = get_object_or_404(Review, pk=pk)
    review.delete()
    return Response({"message": "Review deleted"},
status=status.HTTP_204_NO_CONTENT)

# ---------------- SHIPPING ----------------
```

```
@api_view(['GET'])
def get_shipping_addresses(request):
    addresses = ShippingAddress.objects.all()
    serializer = ShippingAddressSerializer(addresses, many=True)
    return Response(serializer.data)


@api_view(['POST'])
def post_shipping_address(request):
    serializer = ShippingAddressSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

ADMIN.PY

```
from django.contrib import admin
from .models import User, Category, Product, Cart, Order, Review, ShippingAddress


admin.site.register(User)
admin.site.register(Category)
admin.site.register(Product)
admin.site.register(Cart)
admin.site.register(Order)
admin.site.register(Review)
admin.site.register(ShippingAddress)
```