**CASE STUDY**

**AIM:** To design and implement a Django-based boutique e-commerce web application that allows users to browse products, add items to a cart, view cart details including total billing, and perform login-based access control for secure cart operations.

**PROCEDURE:**

1.**Install django:**

To install the Django framework in your system so that you can use its builtin tools to create web projects and apps.

**pip install Django**

**2. Create Django project:**

To initialize new Django project. The project is the overall container for your web application it holds all the configuration files and main settings for the site.

**django-admin startproject boutique**

**cd boutique**

**3. Create Django apps:**

An app is a modular component in Django. Each app performs a specific function. Here, app1 will handle employee input and display.

**python manage.py startapp shop**

**python manage.py startapp cart**

**4. Register the apps:**

To inform Django that we have created a new app and want it included in the project. Without registration, Django won’t recognize the app or its features.

**Settings.py:**

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'shop',

'cart', ]

STATIC\_URL = 'static/'

STATICFILES\_DIRS = [os.path.join(BASE\_DIR, 'static')]

STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

# Default primary key field type

# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field

MEDIA\_URL = '/media/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

**5. Update the urls in project:**

To inform Django that we have created a new app we have to update the urls in the project.

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('shop.urls')),

path('cart/',include('cart.urls')),

path('shop/', include('shop.urls')),

path('accounts/', include('django.contrib.auth.urls')),]

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**6.Write the logic in shop views:**

We have to write the code to perform operations in the boutique/shop/views.py.

from django.shortcuts import render, get\_object\_or\_404

from .models import Category, Product

from django.db.models import Q

def product\_list(request, category\_slug=None):

category = None

categories = Category.objects.all()

products = Product.objects.filter(available=True)

if category\_slug:

category = get\_object\_or\_404(Category, slug=category\_slug)

products = products.filter(category=category)

return render(request, 'shop/product\_list.html', {

'category': category,

'categories': categories,

'products': products

})

def product\_detail(request, id, slug):

product = get\_object\_or\_404(Product, id=id, slug=slug, available=True)

return render(request, 'shop/product\_detail.html', {'product': product})

def product\_search(request):

query = request.GET.get('q')

results = []

if query:

results = Product.objects.filter(Q(name\_\_icontains=query) | Q(description\_\_icontains=query), available=True)

return render(request, 'shop/product\_search.html', {'query': query, 'results': results})

**7.Update the urls in shop app:**

We have to write the code to perform operations in the boutique/shop/urls.py

from django.urls import path

from . import views

app\_name = 'shop'

urlpatterns = [

path('', views.product\_list, name='product\_list'),

path('search/', views.product\_search, name='product\_search'),

path('<slug:category\_slug>/', views.product\_list, name='product\_list\_by\_category'),

path('<int:id>/<slug:slug>/', views.product\_detail, name='product\_detail'),

]

**8.Configure shop models:**

We have to write the to create a table to perform operations in the boutique/shop/models.py

from django.db import models

class Category(models.Model):

name = models.CharField(max\_length=200)

slug = models.SlugField(unique=True)

def \_\_str\_\_(self):

return self.name

class Product(models.Model):

name = models.CharField(max\_length=200)

slug = models.SlugField(unique=True)

description = models.TextField()

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

available = models.BooleanField(default=True)

image = models.ImageField(upload\_to='products/')

category = models.ForeignKey(Category, related\_name='products', on\_delete=models.CASCADE)

def get\_absolute\_url(self):

from django.urls import reverse

return reverse('shop:product\_detail', args=[self.id, self.slug])

def \_\_str\_\_(self):

return self.name

**9.Write the base html code in shop app:**

We have to write the code to display as a website in the boutique/shop/templates/shop/base.html

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>{% block title %}My Boutique{% endblock %}</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">

<link rel="stylesheet" href="{% static 'css/styles.css' %}">

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container">

<a class="navbar-brand" href="{% url 'shop:product\_list' %}">My Boutique</a>

<form class="d-flex ms-auto" action="{% url 'shop:product\_search' %}" method="get">

<input class="form-control me-2" type="search" name="q" placeholder="Search products...">

<button class="btn btn-outline-success" type="submit">Search</button>

</form>

</div>

</nav>

<div class="container mt-4">

{% block content %}

{% endblock %}

</div>

<footer class="text-center mt-5 py-3 border-top">

<p>&copy; 2025 My Boutique. All rights reserved.</p>

</footer>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

**10.html code for product\_details in shop app:**

We have to write the code to display as a website by displaying all the details of the products in the boutique/shop/templates/shop/product\_details.html

{% extends 'base.html' %}

{% block title %}{{ product.name }}{% endblock %}

{% block content %}

<div class="row">

<div class="col-md-6">

<img src="{{ product.image.url }}" class="img-fluid" alt="{{ product.name }}">

</div>

<div class="col-md-6">

<h2>{{ product.name }}</h2>

<p class="text-muted">₹{{ product.price }}</p>

<p>{{ product.description }}</p>

<form method="post" action="{% url 'cart:cart\_add' product.id %}">

{% csrf\_token %}

<button type="submit" class="btn btn-success">Add to Cart</button>

</form>

</div>

</div>

{% endblock %}

**11**.**html code for product\_list in shop app:**

We have to write the code to display as a website by listing the products in the boutique/shop/templates/shop/product\_list.html

{% extends 'base.html' %}

{% block title %}Products{% endblock %}

{% block content %}

<h2>{% if category %}{{ category.name }}{% else %}All Products{% endif %}</h2>

<div class="row">

{% for product in products %}

<div class="col-md-4 mb-4">

<div class="card h-100">

<a href="{{ product.get\_absolute\_url }}">

<img src="{{ product.image.url }}" class="card-img-top" alt="{{ product.name }}">

</a>

<div class="card-body">

<h5 class="card-title">{{ product.name }}</h5>

<p class="card-text">₹{{ product.price }}</p>

<a href="{{ product.get\_absolute\_url }}" class="btn btn-primary">View Details</a>

</div>

</div>

</div>

{% endfor %}

</div>

{% endblock %}

**12.html code for product\_search in shop app:**

We have to write the code to display as a website for the searching in the boutique/shop/templates/shop/product\_search.html

{% extends 'base.html' %}

{% block title %}Search Results{% endblock %}

{% block content %}

<h2>Results for "{{ query }}"</h2>

<div class="row">

{% for product in results %}

<div class="col-md-4 mb-4">

<div class="card h-100">

<a href="{{ product.get\_absolute\_url }}">

<img src="{{ product.image.url }}" class="card-img-top" alt="{{ product.name }}">

</a>

<div class="card-body">

<h5 class="card-title">{{ product.name }}</h5>

<p class="card-text">₹{{ product.price }}</p>

<a href="{{ product.get\_absolute\_url }}" class="btn btn-primary">View Details</a>

</div>

</div>

</div>

{% empty %}

<p>No products found matching your search.</p>

{% endfor %}

</div>

{% endblock %}

**13.html code for login in shop app:**

We have to write the code to display as a website for the login purpose boutique/shop/templates/registration/login.html

{% extends 'base.html' %}

{% block content %}

<h2>Login</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Log in</button>

</form>

{% endblock %}

**Create another app for the cart:**

**14.write the logic in cart views:**

We have to write the code to perform operations in the boutique/cart/urls.py

# cart/views.py

from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login\_required

from .models import Cart, CartItem

from shop.models import Product

@login\_required

def cart\_add(request, product\_id):

product = Product.objects.get(id=product\_id)

cart, created = Cart.objects.get\_or\_create(user=request.user)

cart\_item, created = CartItem.objects.get\_or\_create(cart=cart, product=product)

if not created:

cart\_item.quantity += 1

cart\_item.save()

return redirect('cart:cart\_detail')

@login\_required

def cart\_detail(request):

cart = Cart.objects.get(user=request.user)

cart\_items = cart.items.all()

total\_price = sum(item.total\_price() for item in cart\_items)

return render(request, 'cart/cart\_detail.html', {'cart\_items': cart\_items, 'total\_price': total\_price})

@login\_required

def cart\_remove(request, product\_id):

cart = Cart.objects.get(user=request.user)

try:

cart\_item = CartItem.objects.get(cart=cart, product\_id=product\_id)

cart\_item.delete()

except CartItem.DoesNotExist:

pass # You can also add a message if needed

return redirect('cart:cart\_detail')

**15.update the urls in cart app:**

We have to write the code to perform operations in the boutique/cart/urls.py

# cart/urls.py

from django.urls import path

from . import views

app\_name = 'cart' # Defining namespace for cart URLs

urlpatterns = [

path('add/<int:product\_id>/', views.cart\_add, name='cart\_add'),

path('detail/', views.cart\_detail, name='cart\_detail'),

path('remove/<int:product\_id>/', views.cart\_remove, name='cart\_remove'),

]

**16.configure cart models:**

We have to write the to create a table to perform operations in the boutique/cart/models.py

# cart/models.py

from django.db import models

from django.contrib.auth.models import User

from shop.models import Product # Assuming you have a Product model in the shop app

class Cart(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE, null=True, blank=True) # Optional: User-based cart

created\_at = models.DateTimeField(auto\_now\_add=True)

class CartItem(models.Model):

cart = models.ForeignKey(Cart, related\_name='items', on\_delete=models.CASCADE)

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

quantity = models.PositiveIntegerField(default=1)

created\_at = models.DateTimeField(auto\_now\_add=True)

def total\_price(self):

return self.product.price \* self.quantity

**17.write the code in cart cart\_detail:**

We have to write the code to display as a website in the boutique/cart/templates/cart/cart\_detail.html

<h1>Your Cart</h1>

<table>

<thead>

<tr>

<th>Product</th>

<th>Quantity</th>

<th>Price</th>

<th>Action</th> <!-- New column for remove -->

</tr>

</thead>

<tbody>

{% for item in cart\_items %}

<tr>

<td>{{ item.product.name }}</td>

<td>{{ item.quantity }}</td>

<td>{{ item.total\_price }}</td>

<td>

<form action="{% url 'cart:cart\_remove' item.product.id %}" method="post">

{% csrf\_token %}

<button type="submit">Remove</button>

</form>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<h3>Total Price: {{ total\_price }}</h3>

**18. css code in main project:**

We have to write the code to display as a website in the

boutique/static/css/styles.css

/\* General Body Style \*/

body {

font-family: 'Poppins', sans-serif;

background-color: lightblue; /\* Changed for testing \*/

color: #333;

}

/\* Navbar Test \*/

.navbar {

background-color: #ff6347; /\* Test with a bright color \*/

}

.navbar-brand {

color: #fff !important; /\* Ensure navbar brand text is white \*/

}

/\* Card Test \*/

.card {

border: 2px solid #000; /\* Just added a border for testing \*/

}

/\* Test Button Style \*/

.btn-success {

background-color: green !important;

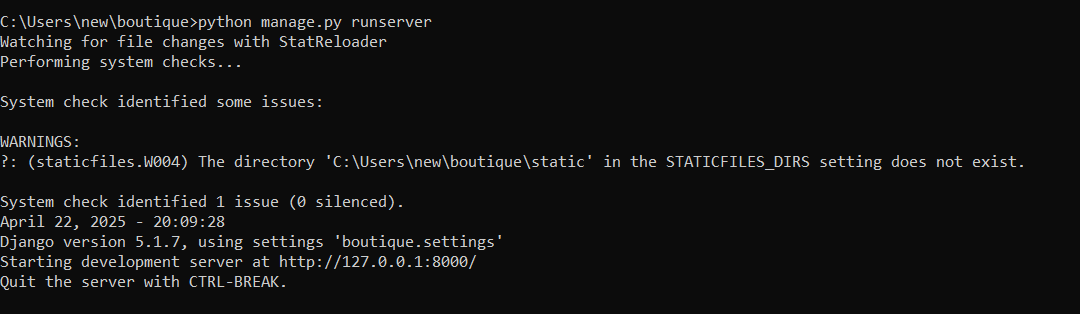
color: white;

}

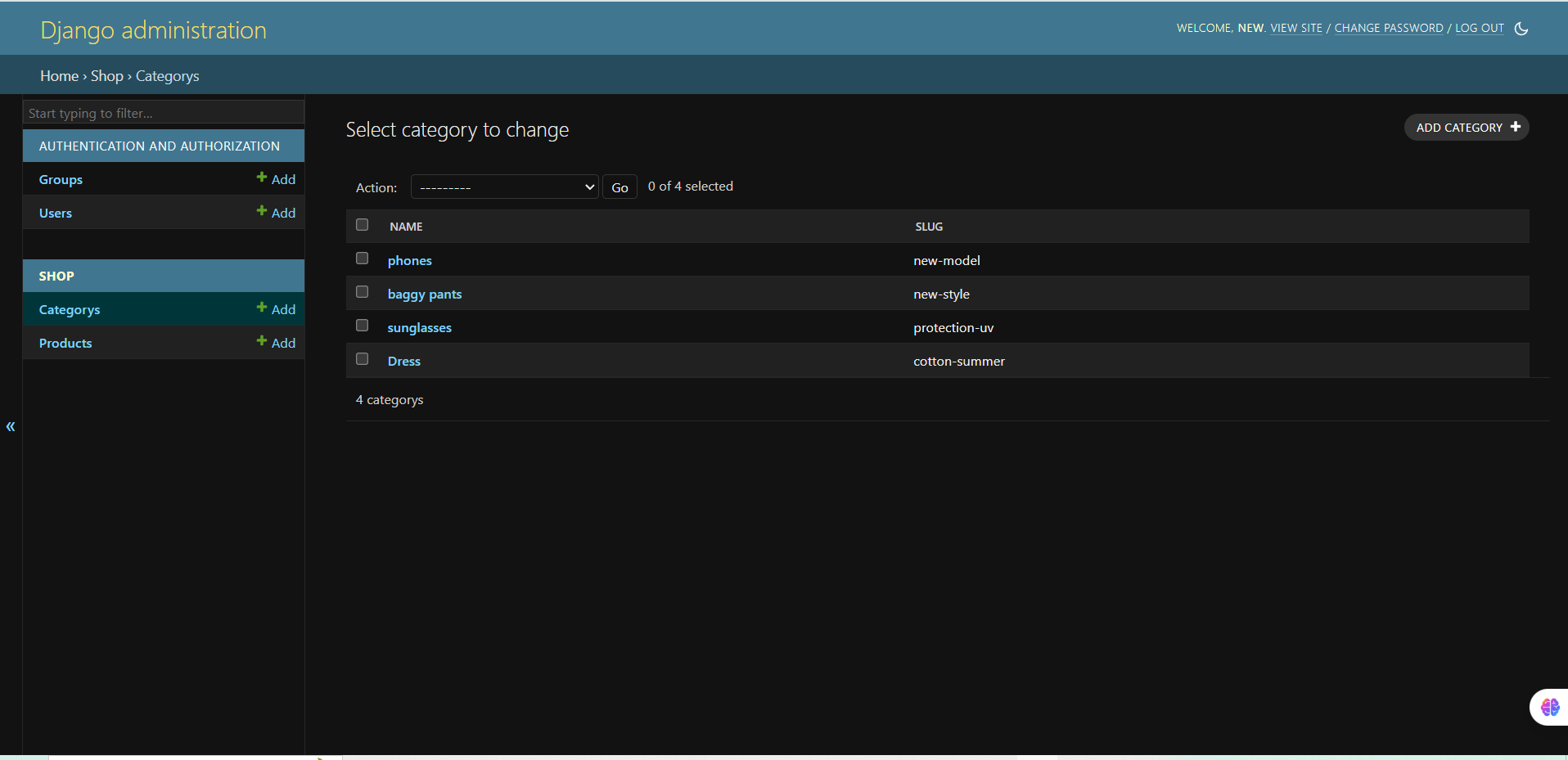
**OUTPUT:**

**19.Run the development server:**

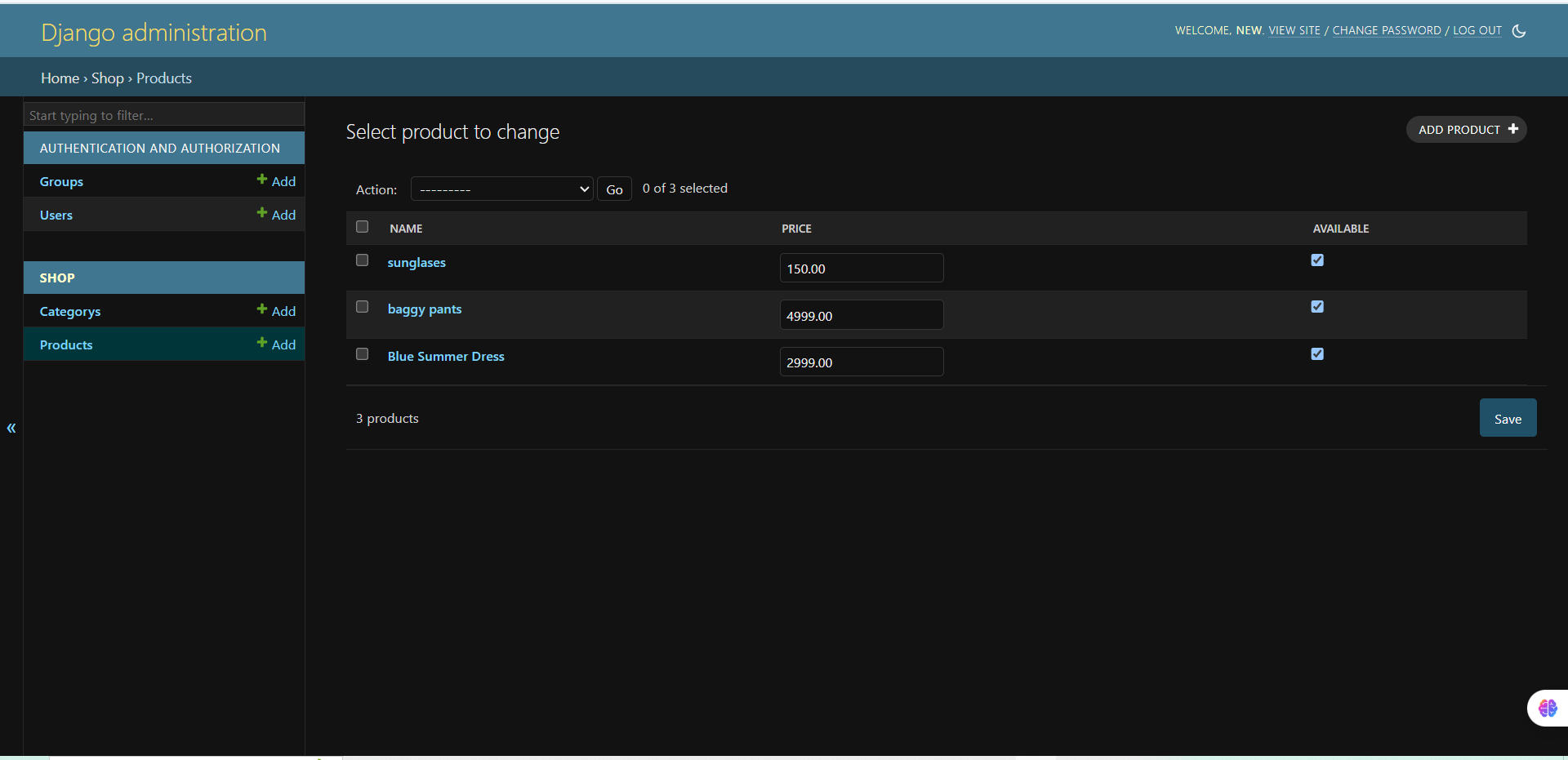
To run the server python manage.py runserver Then move to development server at http://127.0.0.1:8000/admin and login to superuser credentials



After that insert the values into the categories.

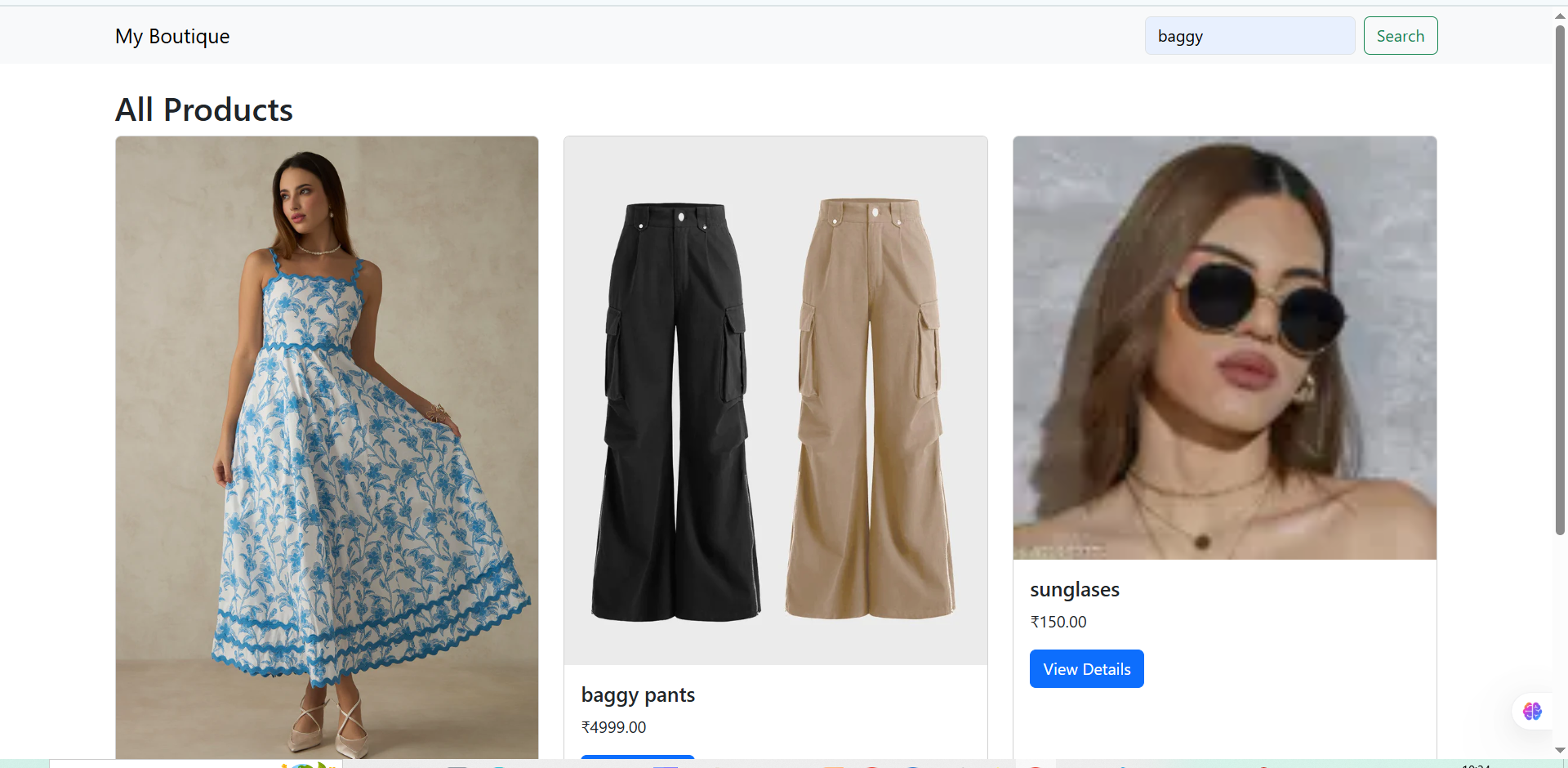


And insert the values into the products

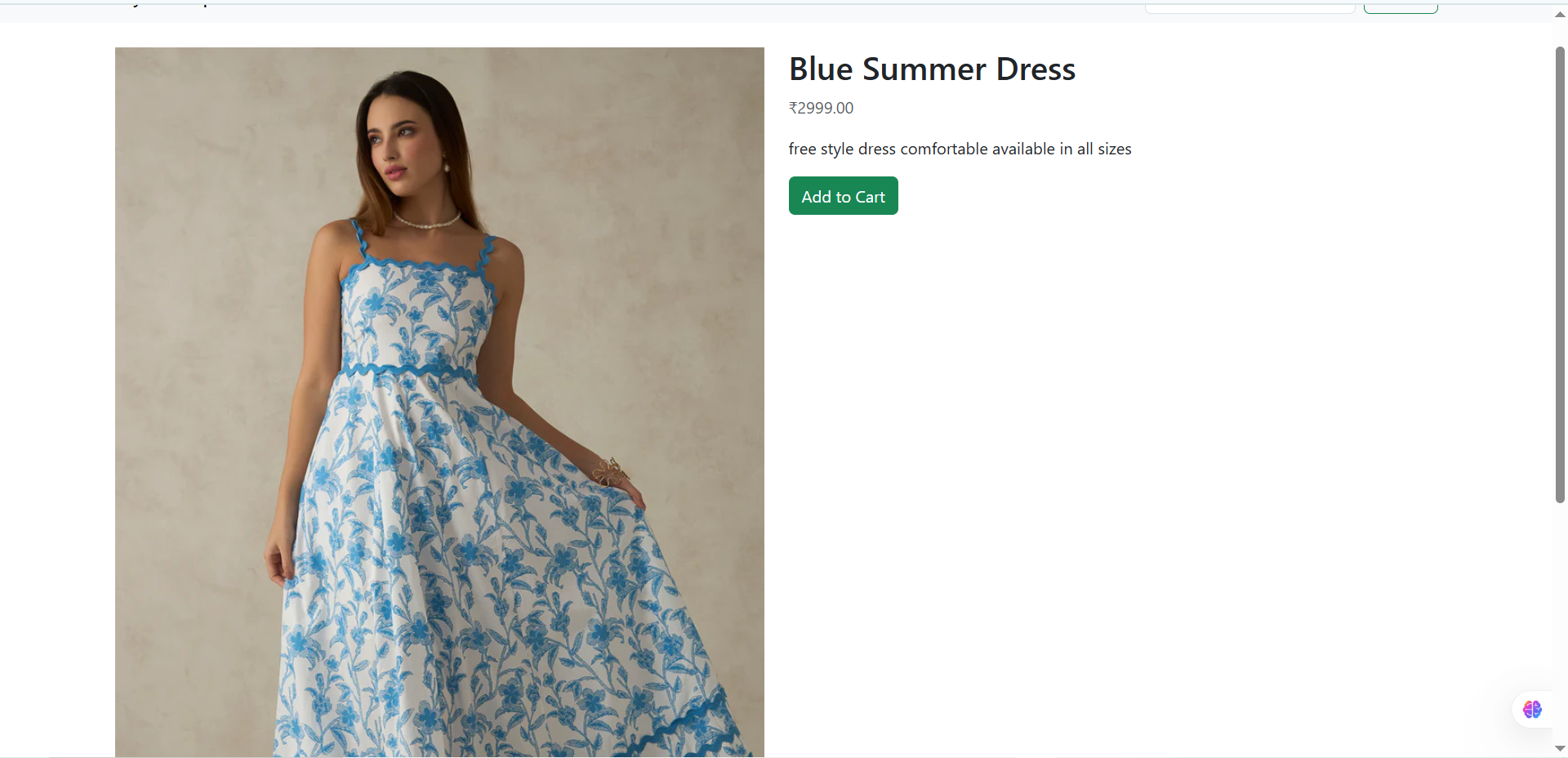


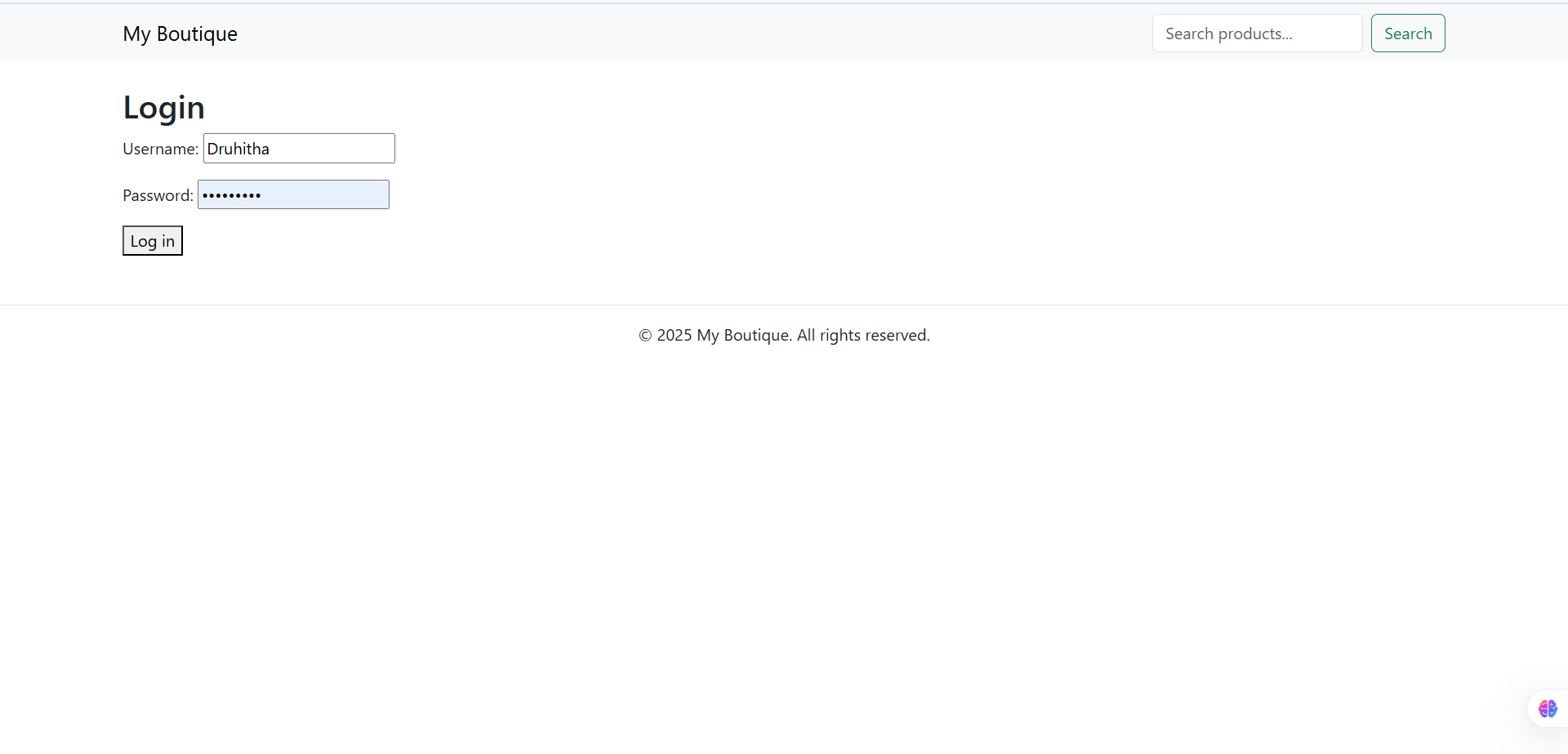
To run the server python manage.py runserver Then move to server at <http://127.0.0.1:8000/>

That is what the website looks like



After the website is created to view the products click on view details



After that it make sures to ask us if whether we logged – in in the site or not

After that we will be processed with the bill and if we want to remoe the item from the cart we can just press the remove button in the cart



To search the products in the website we have the search the products field

