**Case Study --- CraftNest – Local Artisan Marketplace**

Connect local artisans with nearby buyers for handmade goods. Buyers browse listings with filters; artisans can post and manage products. Frontend has a map-based product search and image gallery. Backend handles product listings, user reviews, and order management. Useful for integrating media uploads, REST APIs, and location services

**1. Need**

There’s a growing demand for authentic handmade goods and local craftsmanship, but artisans often lack digital platforms for visibility. **CraftNest** bridges this gap by enabling buyers to find and order products from local artisans via a location-based marketplace.

**2. Requirements**

* Buyers should be able to browse, search, and filter handmade products.
* Artisans should be able to create accounts, upload product listings with images, and manage orders.
* The platform must support media uploads, reviews, and location tagging.
* Responsive UI with map-based product search and REST API integration for mobile access.

**3. Major Modules**

| **Module** | **Function** |
| --- | --- |
| **User Management** | Registration, Login, Role-based access (Buyer/Artisan) |
| **Product Management** | Create/edit/delete product listings with image uploads |
| **Search & Discovery** | Browse, filter, map-based search (by location, category) |
| **Order Management** | Cart, place orders, order history, order status tracking |
| **Review & Rating** | Buyers review purchased products and artisans |
| **API Services** | RESTful APIs for frontend/mobile usage |
| **Admin Panel** | Superuser manages users, products, abuse reports, etc. |

**4. Minor Modules Inside Major Ones**

| **Major Module** | **Minor Components** |
| --- | --- |
| **User Management** | Profile editing, password reset, profile image, location set |
| **Product Management** | Image gallery per product, product tagging, stock availability |
| **Search** | Category filters, keyword search, price range, map display |
| **Orders** | Order summary, shipping address, cancel order, status timeline |
| **Review System** | Rating input, comment moderation, average rating calculator |
| **API Layer** | Token-based auth, pagination, filterable endpoints |

**5. Tools (3-Tier Architecture)**

| **Layer** | **Tools** |
| --- | --- |
| **Presentation (Frontend)** | HTML, Tailwind CSS, JavaScript, Django Templates |
| **Application (Backend/API)** | Django, Django REST Framework (DRF) |
| **Data (Database & Storage)** | PostgreSQL (e.g., Neon), Amazon S3 / local media for uploads |

**6. Database Tables (Normalized Format)**

| **Table** | **Fields** |
| --- | --- |
| User | id (PK), username, email, password, is\_artisan (bool), date\_joined |
| ArtisanProfile | user\_id (FK), bio, profile\_pic, location |
| BuyerProfile | user\_id (FK), location, saved\_items |
| Product | id (PK), artisan\_id (FK), name, description, price, stock, category\_id (FK), location, created\_at |
| ProductImage | id (PK), product\_id (FK), image\_path |
| Category | id (PK), name |
| Order | id (PK), buyer\_id (FK), created\_at, total\_amount, status |
| OrderItem | id (PK), order\_id (FK), product\_id (FK), quantity, price |
| Review | id (PK), buyer\_id (FK), product\_id (FK), rating, comment, created\_at |

**7. Expected Outcome**

CraftNest will enable buyers to easily **discover and purchase handmade goods** from local artisans using a **location-aware map and filters**. Artisans get a **digital storefront** with image uploads and order tracking. Admins monitor user activity and content. The platform will support REST APIs for mobile app extensions and include **media handling**, **authentication**, and **geospatial features** — all built using a modular and scalable full-stack Django architecture.

**Project Structuring**

| **App Name** | **Responsibility** | **Models Included** |
| --- | --- | --- |
| users | Custom user logic & profiles | User, ArtisanProfile, BuyerProfile |
| products | Product listing & management | Product, ProductImage, Category, Review |
| orders | Orders & cart functionality | Order, OrderItem |

**Start Project CraftNest**

**Create 3 apps in the project**

**Add models to the apps**

**1. App: users**

users/models.py

from django.contrib.auth.models import AbstractUser

from django.db import models

class User(AbstractUser):

is\_artisan = models.BooleanField(default=False)

date\_joined = models.DateTimeField(auto\_now\_add=True)

class ArtisanProfile(models.Model):

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

bio = models.TextField(blank=True)

profile\_pic = models.ImageField(upload\_to='artisan\_profiles/', blank=True, null=True)

location = models.CharField(max\_length=100)

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

return f"ArtisanProfile of {self.user.username}"

class BuyerProfile(models.Model):

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

location = models.CharField(max\_length=100)

saved\_items = models.ManyToManyField('products.Product', blank=True)

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

return f"BuyerProfile of {self.user.username}"

Update AUTH\_USER\_MODEL = 'users.User' in settings.py.

**2. App: products**

products/models.py

from django.db import models

from users.models import ArtisanProfile, User

class Category(models.Model):

name = models.CharField(max\_length=50)

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

return self.name

class Product(models.Model):

artisan = models.ForeignKey(ArtisanProfile, on\_delete=models.CASCADE)

name = models.CharField(max\_length=100)

description = models.TextField()

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

stock = models.IntegerField()

category = models.ForeignKey(Category, on\_delete=models.SET\_NULL, null=True)

location = models.CharField(max\_length=100)

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

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

return self.name

class ProductImage(models.Model):

product = models.ForeignKey ( Product, on\_delete = models.CASCADE, related\_name = 'images' )

image\_path = models.ImageField(upload\_to='product\_images/')

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

return f"Image for {self.product.name}"

class Review(models.Model):

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

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

rating = models.IntegerField()

comment = models.TextField(blank=True)

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

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

return f"Review by {self.buyer.username} on {self.product.name}"

**3. App: orders**

orders/models.py

from django.db import models

from users.models import User

from products.models import Product

class Order(models.Model):

STATUS\_CHOICES = [

('pending', 'Pending'),

('confirmed', 'Confirmed'),

('shipped', 'Shipped'),

('delivered', 'Delivered'),

('cancelled', 'Cancelled'),

]

buyer = models.ForeignKey(User, on\_delete=models.SET\_NULL, null=True)

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

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

status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='pending')

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

return f"Order #{self.id} by {self.buyer.username}"

class OrderItem(models.Model):

order = models.ForeignKey(Order, on\_delete=models.CASCADE, related\_name='items')

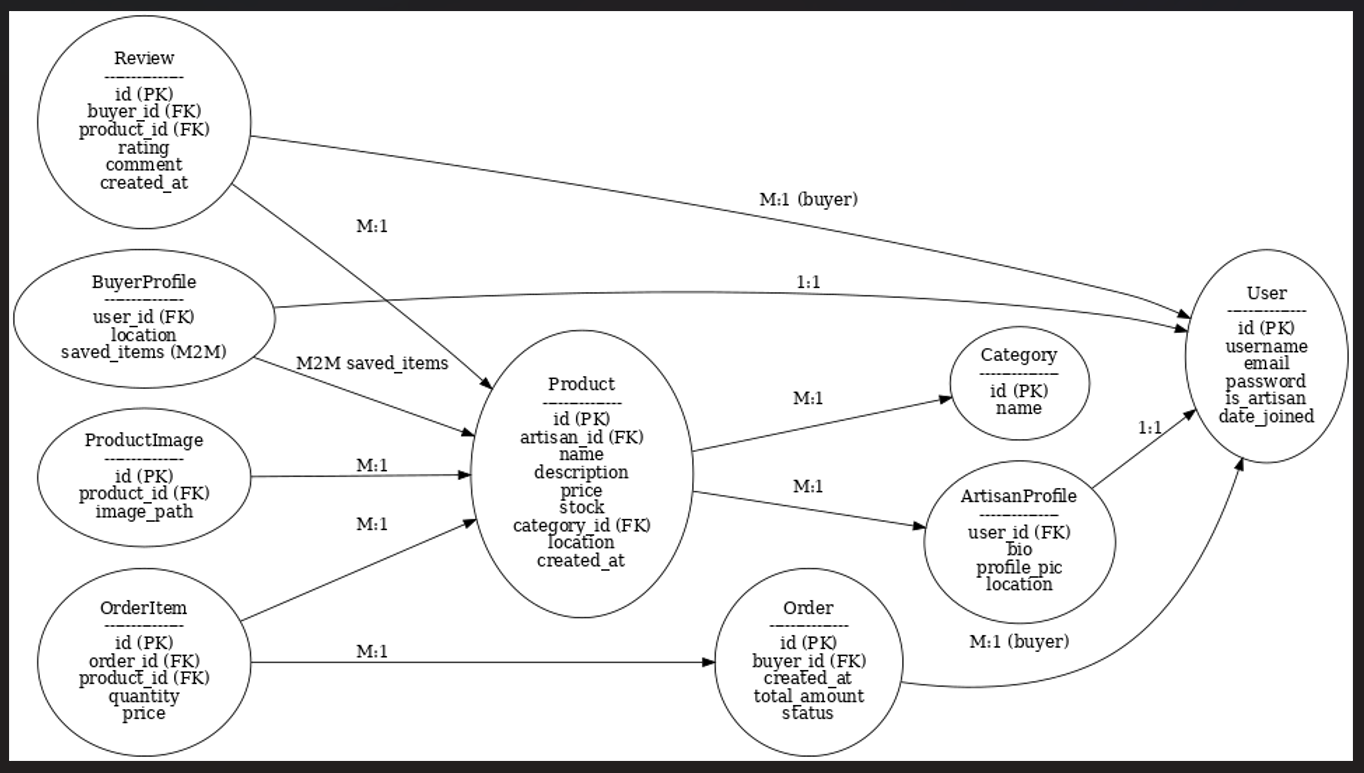
product = models.ForeignKey(Product, on\_delete=models.CASCADE)

quantity = models.PositiveIntegerField()

price = models.DecimalField(max\_digits=10, decimal\_places=2) # price at time of purchase

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

return f"{self.quantity} of {self.product.name} in Order #{self.order.id}"



**PROJECT STRUCTURE**

craftnest/

├── craftnest/ ← Main project folder

│ ├── \_\_init\_\_.py

│ ├── settings.py ← Register apps, templates, DB

│ ├── urls.py ← Global routing

│ └── wsgi.py / asgi.py

│

├── users/ ← User login, artisan, buyer profiles

│ ├── models.py ← User, ArtisanProfile, BuyerProfile

│ ├── views.py ← Registration, login, logout

│ ├── urls.py ← /login/, /register/

│ └── templates/users/ ← login.html, register.html, profiles

│

├── products/ ← Product management

│ ├── models.py ← Product, Category, Image, Review

│ ├── views.py ← List, Detail, Create, Update

│ ├── urls.py ← /products/, /products/<id>/

│ └── templates/products/ ← product\_list.html, detail.html

│

├── orders/ ← Order & cart management

│ ├── models.py ← Order, OrderItem

│ ├── views.py ← Cart, checkout, order tracking

│ ├── urls.py ← /orders/, /cart/

│ └── templates/orders/ ← cart.html, order\_summary.html

│

├── templates/

│ └── base.html ← Common base template

│ └── index.html ← Homepage (your given HTML)

│

├── static/ ← CSS, JS, images

│ └── css/

│ └── js/

│

└── manage.py

**Integration Steps**

**settings.py Configuration**

INSTALLED\_APPS = [

...

'users',

'products',

'orders',

]

TEMPLATES = [

{ 'DIRS': [BASE\_DIR / 'templates'], },

]

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [BASE\_DIR / 'static']

AUTH\_USER\_MODEL = 'users.User'

**craftnest/urls.py – Root URL Config**

from django.contrib import admin

from django.urls import path, include

from django.views.generic import TemplateView

urlpatterns = [

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

path('', TemplateView.as\_view(template\_name='index.html'), name='home'), # homepage

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

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

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

]

**3. URL Sample (products/urls.py)**

from django.urls import path

from . import views

urlpatterns = [

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

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

path('add/', views.product\_add, name='product\_add'),

]

| **Page** | **Template** | **View (FBV)** | **Route** | **App** |
| --- | --- | --- | --- | --- |
| Homepage | index.html | None (static) | / | root |
| Product list | product\_list.html | product\_list() | /products/ | products |
| Product detail | product\_detail.html | product\_detail() | /products/5/ | products |
| Register / Login | register.html | register\_view() | /users/register/ | users |
| Artisan dashboard | dashboard.html | artisan\_dashboard() | /users/dashboard/ | users |
| Cart page | cart.html | view\_cart() | /orders/cart/ | orders |
| Order confirmation | order\_summary.html | checkout() | /orders/checkout/ | orders |

**Best Practices**

* Use base.html for layout (nav, footer) and extend it in index.html, product\_list.html, etc.
* Organize images, CSS, JS under /static/
* Use @login\_required for seller dashboard or cart pages
* Split forms, views, templates per app for maintainability

**UI/UX Design**

**1. base.html (with reusable layout)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>{% block title %}CraftNest – Local Artisan Marketplace{% endblock %}</title>

<script src="https://cdn.tailwindcss.com"></script>

</head>

<body class="font-sans bg-gray-50 text-gray-800">

<!-- Header -->

<header class="bg-white shadow-md py-6">

<div class="container mx-auto text-center">

<h1 class="text-3xl font-bold text-teal-600">CraftNest</h1>

<p class="text-gray-600">Connecting You with Local Artisans & Handmade Treasures</p>

<nav class="mt-4 space-x-4">

<a href="/" class="text-teal-700 hover:underline">Home</a>

<a href="/products/" class="text-teal-700 hover:underline">Products</a>

<a href="/users/login/" class="text-teal-700 hover:underline">Login</a>

<a href="/users/register/" class="text-teal-700 hover:underline">Register</a>

</nav>

</div>

</header>

<!-- Main Content -->

<main class="container mx-auto py-10 px-4">

{% block content %}{% endblock %}

</main>

<!-- Footer -->

<footer class="bg-gray-200 text-center py-4 mt-10">

<p class="text-gray-600">© 2025 CraftNest. All rights reserved.</p>

</footer>

</body>

</html>

**2. index.html (extends base)**

{% extends 'base.html' %}

{% block title %}Welcome to CraftNest{% endblock %}

{% block content %}

<!-- Categories Section -->

<section class="mb-12">

<h2 class="text-2xl font-semibold mb-4">Explore Categories</h2>

<div class="grid grid-cols-2 md:grid-cols-4 gap-6">

<div class="bg-white border rounded-xl p-6 text-center shadow-sm">Home Decor</div>

<div class="bg-white border rounded-xl p-6 text-center shadow-sm">Jewelry</div>

<div class="bg-white border rounded-xl p-6 text-center shadow-sm">Pottery</div>

<div class="bg-white border rounded-xl p-6 text-center shadow-sm">Handwoven</div>

</div>

</section>

<!-- Featured Products Section -->

<section class="mb-12">

<h2 class="text-2xl font-semibold mb-4">Featured Products</h2>

<div class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-6">

<div class="bg-white border rounded-xl p-4 shadow-sm">

<h3 class="text-lg font-semibold">Macrame Wall Hanging</h3>

<p class="text-sm text-gray-500">By: Artisan A</p>

</div>

<div class="bg-white border rounded-xl p-4 shadow-sm">

<h3 class="text-lg font-semibold">Clay Mug Set</h3>

<p class="text-sm text-gray-500">By: Artisan B</p>

</div>

<div class="bg-white border rounded-xl p-4 shadow-sm">

<h3 class="text-lg font-semibold">Beaded Necklace</h3>

<p class="text-sm text-gray-500">By: Artisan C</p>

</div>

</div>

</section>

<!-- Why Choose Section -->

<section class="mb-12">

<h2 class="text-2xl font-semibold mb-4">Why Choose CraftNest?</h2>

<p class="text-gray-700">Support local creators, discover unique handcrafted items, and shop ethically.</p>

</section>

<!-- Become a Seller Section -->

<section class="mb-12">

<h2 class="text-2xl font-semibold mb-4">Become a Seller</h2>

<p class="mb-2 text-gray-700">Ready to showcase your creations?</p>

<a href="/users/register/" class="inline-block bg-teal-600 text-white px-5 py-2 rounded hover:bg-teal-700">

Sign up now

</a>

</section>

{% endblock %}

**REGISTRATION MODULE**

**STEP 1: Configure settings.py**

AUTH\_USER\_MODEL = 'users.User'

TEMPLATES = [

{

'DIRS': [BASE\_DIR / 'templates'],

}

]

**STEP 2: Update users/models.py (DONE)**

from django.contrib.auth.models import AbstractUser

from django.db import models

class User(AbstractUser):

is\_artisan = models.BooleanField(default=False)

date\_joined = models.DateTimeField(auto\_now\_add=True)

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

return self.username

python manage.py makemigrations users

python manage.py migrate

**STEP 3: Create users/forms.py**

from django import forms

from django.contrib.auth.forms import UserCreationForm

from .models import User

class RegisterForm(UserCreationForm):

email = forms.EmailField(required=True)

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2', 'is\_artisan']

**STEP 4: Create users/views.py**

from django.shortcuts import render, redirect

from django.contrib.auth import login, authenticate, logout

from .forms import RegisterForm

from django.contrib import messages

def register\_view(request):

if request.method == 'POST':

form = RegisterForm(request.POST)

if form.is\_valid():

user = form.save()

login(request, user) # Auto login after register

messages.success(request, "Registration successful.")

return redirect('home')

else:

messages.error(request, "Please correct the errors below.")

else:

form = RegisterForm()

return render(request, 'users/register.html', {'form': form})

def login\_view(request):

if request.method == 'POST':

username = request.POST['username']

password = request.POST['password']

user = authenticate(request, username=username, password=password)

if user:

login(request, user)

return redirect('home')

else:

messages.error(request, "Invalid username or password.")

return render(request, 'users/login.html')

def logout\_view(request):

logout(request)

return redirect('home')

**STEP 5: Setup users/urls.py**

from django.urls import path

from . import views

urlpatterns = [

path('register/', views.register\_view, name='register'),

path('login/', views.login\_view, name='login'),

path('logout/', views.logout\_view, name='logout'),

]

And include it in the main craftnest/urls.py:

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

**STEP 6: Template – templates/users/register.html**

{% extends 'base.html' %}

{% block title %}Register{% endblock %}

{% block content %}

<div class="max-w-md mx-auto bg-white p-8 shadow-md rounded">

<h2 class="text-2xl font-semibold mb-4">Register</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="bg-teal-600 text-white px-4 py-2 rounded mt-2">Register</button>

</form>

<p class="mt-4 text-sm text-gray-600">Already have an account? <a href="/users/login/" class="text-teal-700">Login</a></p>

</div>

{% endblock %}

**STEP 7: Template – templates/users/login.html**

{% extends 'base.html' %}

{% block title %}Login{% endblock %}

{% block content %}

<div class="max-w-md mx-auto bg-white p-8 shadow-md rounded">

<h2 class="text-2xl font-semibold mb-4">Login</h2>

<form method="post">

{% csrf\_token %}

<div class="mb-4">

<label class="block mb-1">Username:</label>

<input type="text" name="username" class="w-full border rounded px-3 py-2" required>

</div>

<div class="mb-4">

<label class="block mb-1">Password:</label>

<input type="password" name="password" class="w-full border rounded px-3 py-2" required>

</div>

<button type="submit" class="bg-teal-600 text-white px-4 py-2 rounded">Login</button>

</form>

<p class="mt-4 text-sm text-gray-600">Don't have an account? <a href="/users/register/" class="text-teal-700">Register</a></p>

</div>

{% endblock %}

**STEP 8: Test !**

* Navigate to /users/register/ → register a new user.
* Redirects to home.
* Logout via /users/logout/
* Then login via /users/login/

**ARTISAN DASHBOARD MODULE**

**STEP 1: Update views.py in users**

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

from django.shortcuts import render, redirect

from products.models import Product # Assuming product model is in products app

@login\_required

def artisan\_dashboard(request):

if not request.user.is\_artisan:

return redirect('home') # or show access denied message

products = Product.objects.filter(artisan=request.user)

return render(request, 'users/dashboard.html', {

'user': request.user,

'products': products

})

**STEP 2: Add URL Route in users/urls.py**

from django.urls import path

from . import views

urlpatterns = [

path('register/', views.register\_view, name='register'),

path('login/', views.login\_view, name='login'),

path('logout/', views.logout\_view, name='logout'),

path('dashboard/', views.artisan\_dashboard, name='artisan\_dashboard'),

]

**STEP 3: Create Template templates/users/dashboard.html**

{% extends 'base.html' %}

{% block title %}Artisan Dashboard{% endblock %}

{% block content %}

<div class="max-w-4xl mx-auto bg-white p-8 shadow rounded">

<h2 class="text-2xl font-bold mb-4">Welcome, {{ user.username }}</h2>

<p class="text-gray-600 mb-6">You are logged in as an artisan. Below are your listed products.</p>

{% if products %}

<div class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">

{% for product in products %}

<div class="border p-4 rounded shadow">

<h3 class="text-lg font-semibold">{{ product.name }}</h3>

<p class="text-sm text-gray-500">Price: ₹{{ product.price }}</p>

<p class="text-sm text-gray-500">Stock: {{ product.stock }}</p>

</div>

{% endfor %}

</div>

{% else %}

<p class="text-gray-500">You haven’t posted any products yet.</p>

{% endif %}

<div class="mt-6">

<a href="/products/add/" class="inline-block bg-teal-600 text-white px-4 py-2 rounded hover:bg-teal-700">

+ Add New Product

</a>

</div>

</div>

{% endblock %}

**Step 4: Optional – Protect Product Model for Artisan Field**

Make sure your Product model (in products/models.py) has a field:

artisan = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

**Step 5: Add "Dashboard" to Header (Optional in base.html)**

<a href="/users/dashboard/" class="text-teal-700 hover:underline">Dashboard</a>

Only show it if the user is logged in. You can enhance it later with {% if user.is\_authenticated and user.is\_artisan %} logic.

**Show Logout if Logged In, Otherwise Login/Register**

**base.html (change <nav> tag)**

<nav class="mt-4 space-x-4">

<a href="/" class="text-teal-700 hover:underline">Home</a>

<a href="/products/" class="text-teal-700 hover:underline">Products</a>

{% if user.is\_authenticated %}

{% if user.is\_artisan %}

<a href="/users/dashboard/" class="text-teal-700 hover:underline">Dashboard</a>

{% endif %}

<span class="text-gray-600">Hi, {{ user.username }}!</span>

<a href="/users/logout/" class="text-teal-700 hover:underline">Logout</a>

{% else %}

<a href="/users/login/" class="text-teal-700 hover:underline">Login</a>

<a href="/users/register/" class="text-teal-700 hover:underline">Register</a>

{% endif %}

</nav>

**Tailwind-Styled Registration Form**

Replace: {{ form.as\_p }}

with a manual layout

<form method="post" novalidate>

{% csrf\_token %}

<div class="mb-4">

<label for="id\_username" class="block text-sm font-medium text-gray-700">Username</label>

{{ form.username|add\_class:"w-full mt-1 border px-3 py-2 rounded shadow-sm focus:ring-teal-500 focus:border-teal-500" }}

{% if form.username.errors %}

<p class="text-sm text-red-600">{{ form.username.errors.0 }}</p>

{% endif %}

</div>

<div class="mb-4">

<label for="id\_email" class="block text-sm font-medium text-gray-700">Email</label>

{{ form.email|add\_class:"w-full mt-1 border px-3 py-2 rounded shadow-sm focus:ring-teal-500 focus:border-teal-500" }}

{% if form.email.errors %}

<p class="text-sm text-red-600">{{ form.email.errors.0 }}</p>

{% endif %}

</div>

<div class="mb-4">

<label for="id\_password1" class="block text-sm font-medium text-gray-700">Password</label>

{{ form.password1|add\_class:"w-full mt-1 border px-3 py-2 rounded shadow-sm focus:ring-teal-500 focus:border-teal-500" }}

{% if form.password1.errors %}

<ul class="text-sm text-red-600 list-disc ml-5">

{% for err in form.password1.errors %}

<li>{{ err }}</li>

{% endfor %}

</ul>

{% endif %}

</div>

<div class="mb-4">

<label for="id\_password2" class="block text-sm font-medium text-gray-700">Password Confirmation</label>

{{ form.password2|add\_class:"w-full mt-1 border px-3 py-2 rounded shadow-sm focus:ring-teal-500 focus:border-teal-500" }}

{% if form.password2.errors %}

<p class="text-sm text-red-600">{{ form.password2.errors.0 }}</p>

{% endif %}

</div>

<div class="mb-6">

<label class="inline-flex items-center">

{{ form.is\_artisan }}

<span class="ml-2 text-gray-700">I am an artisan</span>

</label>

</div>

<div>

<button type="submit" class="bg-teal-600 text-white px-6 py-2 rounded hover:bg-teal-700 transition">

Register

</button>

</div> </form>

**Add add\_class Template Filter**

users/templatetags/form\_filters.py (templatetags is a package)

from django import template

register = template.Library()

@register.filter(name='add\_class')

def add\_class(field, css\_class):

return field.as\_widget(attrs={"class": css\_class})

**{% load form\_filters %} on top of any html page that needs tailwind styling**

**Adding artisan & buyer profile data automatically on registering**

**Create signals.py inside your users app**

from django.db.models.signals import post\_save  
from django.dispatch import receiver  
from django.contrib.auth import get\_user\_model  
from .models import ArtisanProfile, BuyerProfile  
  
User = get\_user\_model()  
  
@receiver(post\_save, sender=User)  
def create\_user\_profile(sender, instance, created, \*\*kwargs):  
 if created:  
 if instance.is\_artisan:  
 ArtisanProfile.objects.create(user=instance)  
 else:  
 BuyerProfile.objects.create(user=instance)

**Connect the Signal in users/apps.py**

from django.apps import AppConfig

class UsersConfig(AppConfig):

name = 'users'

def ready(self):

import users.signals

settings.py

INSTALLED\_APPS = [

...

'users.apps.UsersConfig',

...

]

**Update register view**

def register\_view(request):

if request.method == 'POST':

form = RegisterForm(request.POST)

if form.is\_valid():

user = form.save(commit=False)

user.is\_artisan = form.cleaned\_data.get('is\_artisan')

user.save()

if user.is\_artisan:

ArtisanProfile.objects.create(user=user)

login(request, user)

return redirect('artisan\_dashboard' if user.is\_artisan else 'home')

else:

form = RegisterForm()

return render(request, 'users/register.html', {'form': form})

**Update artisan\_dashboard view**

Previous product line with this

Product.objects.filter(artisan=request.user.artisanprofile)

**CRUD for product (logging as artisan)**

**products/urls.py**

from django.urls import path

from . import views

urlpatterns = [

path('add/', views.add\_product, name='add\_product'),

path('my-products/', views.my\_products, name='my\_products'),

path('edit/<int:pk>/', views.edit\_product, name='edit\_product'),

path('delete/<int:pk>/', views.delete\_product, name='delete\_product'),

]

main urls.py

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

**products/forms.py**

from django import forms

from .models import Product

class ProductForm(forms.ModelForm):

class Meta:

model = Product

fields = ['name', 'description', 'price', 'stock', 'category', 'location']

**products/views.py**

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

from .forms import ProductForm

from .models import Product

from users.models import ArtisanProfile

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

@login\_required

def add\_product(request):

try:

artisan\_profile = request.user.artisanprofile

except ArtisanProfile.DoesNotExist:

return redirect('dashboard') # or show an error

if request.method == 'POST':

form = ProductForm(request.POST)

if form.is\_valid():

product = form.save(commit=False)

product.artisan = artisan\_profile

product.save()

return redirect('my\_products')

else:

form = ProductForm()

return render(request, 'products/add\_product.html', {'form': form})

@login\_required

def my\_products(request):

try:

artisan\_profile = request.user.artisanprofile

products = Product.objects.filter(artisan=artisan\_profile)

except ArtisanProfile.DoesNotExist:

products = []

return render(request, 'products/my\_products.html', {'products': products})

@login\_required

def edit\_product(request, pk):

product = get\_object\_or\_404(Product, pk=pk, artisan=request.user.artisanprofile)

if request.method == 'POST':

form = ProductForm(request.POST, instance=product)

if form.is\_valid():

form.save()

return redirect('my\_products')

else:

form = ProductForm(instance=product)

return render(request, 'products/edit\_product.html', {'form': form})

@login\_required

def delete\_product(request, pk):

product = get\_object\_or\_404(Product, pk=pk, artisan=request.user.artisanprofile)

if request.method == 'POST':

product.delete()

return redirect('my\_products')

return render(request, 'products/confirm\_delete.html', {'product': product})

**add\_product.html**

{% extends 'base.html' %}

{% block title %}Add Product{% endblock %}

{% block content %}

<h2 class="text-xl font-semibold mb-4">Add New Product</h2>

<form method="post" class="bg-white p-6 rounded shadow-md max-w-md">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="mt-4 bg-teal-600 text-white px-4 py-2 rounded">Add Product</button>

</form>

{% endblock %}

**my\_products.html**

{% extends 'base.html' %}

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

{% block content %}

<h2 class="text-xl font-semibold mb-4">My Products</h2>

<a href="{% url 'add\_product' %}" class="inline-block mb-4 bg-teal-700 text-white px-4 py-2 rounded">+ Add New</a>

<ul class="space-y-4">

{% for product in products %}

<li class="bg-white p-4 shadow rounded">

<div class="flex justify-between items-center">

<div>

<strong>{{ product.name }}</strong><br>

₹{{ product.price }} – {{ product.stock }} in stock

</div>

<div class="space-x-2">

<a href="{% url 'edit\_product' product.pk %}" class="text-blue-600">Edit</a>

<a href="{% url 'delete\_product' product.pk %}" class="text-red-600">Delete</a>

</div>

</div>

</li>

{% empty %}

<li>No products added yet.</li>

{% endfor %}

</ul>

{% endblock %}

**edit\_product.html**

{% extends 'base.html' %}

{% block title %}Edit Product{% endblock %}

{% block content %}

<h2 class="text-xl font-semibold mb-4">Edit Product</h2>

<form method="post" class="bg-white p-6 rounded shadow-md max-w-md">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="mt-4 bg-teal-600 text-white px-4 py-2 rounded">Save Changes</button>

</form>

{% endblock %}

**confirm\_delete.html**

{% extends 'base.html' %}

{% block title %}Delete Product{% endblock %}

{% block content %}

<div class="max-w-md bg-white p-6 shadow rounded">

<h2 class="text-xl mb-4">Confirm Deletion</h2>

<p>Are you sure you want to delete <strong>{{ product.name }}</strong>?</p>

<form method="post" class="mt-4">

{% csrf\_token %}

<button type="submit" class="bg-red-600 text-white px-4 py-2 rounded">Yes, Delete</button>

<a href="{% url 'my\_products' %}" class="ml-4 text-gray-700">Cancel</a>

</form>

</div>

{% endblock %}

**Products display after Artisan login**

**Step 1: View Artisan's Products**

products/views.py

from django.shortcuts import render, redirect

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

from products.models import Product

from users.models import ArtisanProfile

@login\_required

def artisan\_dashboard(request):

# Ensure the user is an artisan

try:

artisan = request.user.artisanprofile # Get ArtisanProfile linked to User

products = Product.objects.filter(artisan=artisan)

except ArtisanProfile.DoesNotExist:

return redirect('home') # or show an error page

return render(request, 'products/artisan\_dashboard.html', {

'products': products

})

**Step 2: Add URL for Dashboard**

products/urls.py

from django.urls import path

from . import views

urlpatterns = [

# ... other routes

path('dashboard/', views.artisan\_dashboard, name='artisan\_dashboard'),

]

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

**Step 3: Template – artisan\_dashboard.html**

templates/products/artisan\_dashboard.html

{% extends 'base.html' %}

{% block title %}Artisan Dashboard{% endblock %}

{% block content %}

<h2 class="text-2xl font-bold mb-4">Welcome, {{ user.username }}</h2>

<h3 class="text-lg font-semibold mb-2">Your Products</h3>

<a href="{% url 'add\_product' %}" class="inline-block bg-teal-700 text-white px-4 py-2 rounded mb-4">+ Add New Product</a>

<ul class="space-y-4">

{% for product in products %}

<li class="bg-white p-4 shadow rounded">

<div class="flex justify-between items-center">

<div>

<p class="font-semibold">{{ product.name }}</p>

<p>Price: ₹{{ product.price }}</p>

<p>Stock: {{ product.stock }}</p>

</div>

<div class="space-x-2">

<a href="{% url 'edit\_product' product.pk %}" class="text-blue-600">Edit</a>

<a href="{% url 'delete\_product' product.pk %}" class="text-red-600">Delete</a>

</div>

</div>

</li>

{% empty %}

<li>No products yet. <a href="{% url 'add\_product' %}" class="text-teal-700">Add one now</a>.</li>

{% endfor %}

</ul>

{% endblock %}

**Step 4: Add "Dashboard" Link in Navigation (Optional)**

base.html:

{% if user.is\_authenticated and user.is\_artisan %}

<a href="{% url 'artisan\_dashboard' %}" class="mr-4">Dashboard</a>

{% endif %}

**Step 5: Redirect After Login**

settings.py

LOGIN\_REDIRECT\_URL = '/products/dashboard/'

**Place Order by Buyer**

**Step 1: Add Order Form**

orders/forms.py

from django import forms

class OrderForm(forms.Form):

quantity = forms.IntegerField(min\_value=1, label="Quantity")

**Step 2: Order Views**

orders/views.py

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

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

from products.models import Product

from .models import Order, OrderItem

from .forms import OrderForm

from users.models import BuyerProfile

@login\_required

def place\_order(request, product\_id):

product = get\_object\_or\_404(Product, id=product\_id)

if not request.user.is\_authenticated or not hasattr(request.user, 'buyerprofile'):

return redirect('login')

if request.method == 'POST':

form = OrderForm(request.POST)

if form.is\_valid():

quantity = form.cleaned\_data['quantity']

total\_price = product.price \* quantity

# Create order

order = Order.objects.create(

buyer=request.user,

total\_amount=total\_price,

status='Placed'

)

# Create order item

OrderItem.objects.create(

order=order,

product=product,

quantity=quantity,

price=product.price

)

return render(request, 'orders/order\_success.html', {

'product': product,

'order': order

})

else:

form = OrderForm()

return render(request, 'orders/place\_order.html', {

'product': product,

'form': form

})

**Step 3: Add URLs**

orders/urls.py

from django.urls import path

from . import views

urlpatterns = [

path('place/<int:product\_id>/', views.place\_order, name='place\_order'),

]

**Main urls.py:**

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

**Step 4: Templates**

**place\_order.html**

{% extends 'base.html' %}

{% block title %}Place Order{% endblock %}

{% block content %}

<h2 class="text-xl font-bold mb-4">Order: {{ product.name }}</h2>

<p>Price: ₹{{ product.price }}</p>

<form method="post" class="bg-white p-4 rounded shadow max-w-sm">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="mt-2 bg-teal-600 text-white px-4 py-2 rounded">Confirm Order</button>

</form>

{% endblock %}

**order\_success.html**

{% extends 'base.html' %}

{% block title %}Order Placed{% endblock %}

{% block content %}

<h2 class="text-xl font-bold mb-4">✅ Order Placed Successfully!</h2>

<p>You ordered <strong>{{ order.orderitem\_set.first.quantity }}</strong> unit(s) of <strong>{{ product.name }}</strong>.</p>

<p>Total: ₹{{ order.total\_amount }}</p>

<a href="/" class="mt-4 inline-block text-teal-700 underline">Go back to home</a>

{% endblock %}

**Step 5: Add “Place Order” Button in Product Detail View**

**product\_detail.html**

<a href="{% url 'place\_order' product.id %}" class="bg-teal-700 text-white px-4 py-2 rounded">Place Order</a>