Base.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title> {% block title %}{% endblock title %} </title>

  <!-- Link to CSS -->
  <link rel="stylesheet" href="{% static 'style.css' %}">
  <!-- Bootstrap CSS -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.m
in.css" rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&
display=swap" rel="stylesheet">

  <!-- Bootstrap JavaScript -->
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bun
dle.min.js"></script>
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <a class="navbar-brand" href="#">Navbar</a>
            <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarNav"
aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
              <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
              <ul class="navbar-nav">
                <li class="nav-item active">
                  <a class="nav-link" href="{% url 'home' %}">Home</a>
                </li>
                <li class="nav-item">
```

```
                <a class="nav-link" href="{% url 'painting_list'
%}">Painting</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="{% url 'category_list'
%}">Category</a>
              </ul>
            </div>
        </nav>
    </header>

    <main>
        {% block body %} {% endblock body %}
    </main>



</body>
</html>
```

Category_list.html

```
{% extends 'base.html' %}

{% block title %}
    Category Page
{% endblock %}

{% block body %}
    <section class="painting-list-container">
        <h1>Category List</h1>
        {% for category in categories %}
            <div class="category">
                <h2>{{ category.title }}</h2>
                <p>Artist: {{ category.artist }}</p>
            </div>
        {% endfor %}
    </section>
{% endblock %}
```

Home

```
{% extends "base.html" %}

{% block title %}Home{% endblock title %}

{% block body %}
<!-- Carousel Section (Moved from base.html to home.html) -->
<div id="carouselExampleDark" class="carousel carousel-dark slide">
    <div class="carousel-indicators">
      <button type="button" data-bs-target="#carouselExampleDark"
data-bs-slide-to="0" class="active" aria-current="true"
aria-label="Slide 1"></button>
      <button type="button" data-bs-target="#carouselExampleDark"
data-bs-slide-to="1" aria-label="Slide 2"></button>
      <button type="button" data-bs-target="#carouselExampleDark"
data-bs-slide-to="2" aria-label="Slide 3"></button>
    </div>
    <div class="carousel-inner">
      <div class="carousel-item active" data-bs-interval="10000">
        <img
src="https://images.unsplash.com/photo-1515405295579-ba7b45403062?w=100
0&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h
8MjB8fHBhaW50aW5nfGVufDB8fDB8fHww" class="d-block w-100" alt="...">
        <div class="carousel-caption d-none d-md-block">
          <h5>First slide label</h5>
          <p>Some representative placeholder content for the first
slide.</p>
        </div>
      </div>
      <div class="carousel-item" data-bs-interval="2000">
        <img
src="https://images.unsplash.com/photo-1576773689115-5cd2b0223523?w=100
0&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h
8MTV8fHBhaW50aW5nfGVufDB8fDB8fHww" class="d-block w-100" alt="...">
        <div class="carousel-caption d-none d-md-block">
          <h5>Second slide label</h5>
          <p>Some representative placeholder content for the second
slide.</p>
        </div>
      </div>
      <div class="carousel-item">
        <img
src="https://images.unsplash.com/photo-1579541592065-da8a15e49bc7?w=100
```

```
0&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h
8MTl8fHBhaW50aW5nfGVufDB8fDB8fHww" class="d-block w-100" alt="...">
        <div class="carousel-caption d-none d-md-block">
          <h5>Third slide label</h5>
          <p>Some representative placeholder content for the third
slide.</p>
        </div>
      </div>
    </div>
    <button class="carousel-control-prev" type="button"
data-bs-target="#carouselExampleDark" data-bs-slide="prev">
      <span class="carousel-control-prev-icon"
aria-hidden="true"></span>
      <span class="visually-hidden">Previous</span>
    </button>
    <button class="carousel-control-next" type="button"
data-bs-target="#carouselExampleDark" data-bs-slide="next">
      <span class="carousel-control-next-icon"
aria-hidden="true"></span>
      <span class="visually-hidden">Next</span>
    </button>
</div>

<!-- Call to Action Section -->
<section class="bg-primary text-white text-center py-50"
style="position: relative; z-index: 1;">
  <div class="container" style="position: relative;">
    <h2 class="mb-4">Don't Miss Out!</h2>
    <p>Explore our latest collection and bring home a masterpiece.</p>
    <a href="{% url 'painting_list' %}" class="btn btn-light
btn-lg">Browse All Paintings</a>
  </div>
</section>

{% endblock body %}
```

Painting list

```
{% extends 'base.html' %}

{% block title %}Paintings List{% endblock %}

{% block body %}
```

```
<div class="row justify-content-center">
    {% for painting in paintings %}
    <div class="col-md-4 d-flex justify-content-center">
        <div class="card painting-card shadow-sm">
            <img src="{{ painting.image.url }}" class="card-img-top
painting-image" alt="{{ painting.title }}">
            <div class="card-body text-center">
                <h5 class="card-title painting-title">{{ painting.title
}}</h5>
                <p class="card-text painting-description">{{
painting.description }}</p>
                <p class="painting-price fw-bold">${{ painting.price
}}</p>
            </div>
        </div>
    </div>
    {% endfor %}
</div>
{% endblock %}
```

Views.py

```
from django.shortcuts import render
from .models import Painting ,Category
# Create your views here.

def painting_list(request):
    paintings = Painting.objects.all()
    return render(request , 'painting_list.html' , {'paintings':
paintings})


def category_list(request):
    categories = Category.objects.all()
    return render(request , 'Category_list.html' , {'categories':
categories})

def home(request):
    return render(request , 'home.html')
```

Urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [

 path('paintings', views.painting_list, name='painting_list'),
 path('categories', views.category_list, name = 'category_list'),
  path('paintingShop', views.home, name = 'home')
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Create style.css in static folder

```
/* General Styling */
body {
    font-family: 'Poppins', sans-serif;
    background-color: #f5f5f5;
}

/* Container */
.container {
    max-width: 1200px;
    margin: auto;
    padding: 20px;
}

/* Painting Card */
.painting-card {
    border-radius: 12px;
    overflow: hidden;
    transition: transform 0.3s ease-in-out, box-shadow 0.3s
ease-in-out;
    background: white;
}

.painting-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
```

```
}

/* Image */
.painting-image {
    width: 100%;
    height: 250px;
    object-fit: cover;
}

/* Card Body */
.card-body {
    text-align: center;
}

/* Price */
.text-success {
    font-size: 1.2rem;
}

/* Order Button */
.btn-primary {
    width: 100%;
    padding: 10px;
    font-size: 1rem;
    border-radius: 8px;
    transition: background 0.3s ease;
}

.btn-primary:hover {
    background-color: #0056b3;
}
```

Contact Form

Define the Model (Optional)

This step is **optional**. If you want to store the contact form data in the database (for example, saving the name, email, and message), you can create a model for it.

Model Explanation:

- A model in Django defines the structure of the database table.
- In this case, we're creating a model called `Contact`, which has three fields: `name`, `email`, and `message`.

```
class Contact(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    message = models.TextField()

    def __str__(self):
        return self.name
```

Why We Use Models:

- Storing the data in a model allows us to keep track of each submission.
- We can later query this data, generate reports, or send emails.

Steps for the Model:

- After defining the model, run `python manage.py makemigrations` and `python manage.py migrate` to apply the model to the database.

```
python manage.py makemigrations
python manage.py migrate
```

---

3. Create the Form

Forms in Django allow you to capture user input. For this simple contact form, we'll create a basic form.

Form Explanation:

- The form is built using Django's `forms.Form` class.
- The `ContactForm` class has three fields: `name`, `email`, and `message`, each with specific widgets for styling.

`Create forms.py`

```
from django import forms
```

```
class ContactForm(forms.Form):
    name = forms.CharField(max_length=100)
    email = forms.EmailField()
    message = forms.CharField(widget=forms.Textarea)
```

Why We Use Forms:

- Forms help with validating user input.
- They handle things like required fields, field types, and rendering fields to HTML.

---

4. Write the View

The **view** is where the form is processed. This view handles both displaying the form to the user and processing the form when it's submitted.

View Explanation:

- If the request method is `POST`, it means the user submitted the form, and we check if the form is valid.
- If the form is valid, you can process the data (like saving it to the database or sending an email).
- If the form is not valid, it will be redisplayed with error messages.
- If the request method is `GET`, the form is simply displayed (i.e., when the user first visits the page).

```
# views.py in the contact app
from django.shortcuts import render
from .forms import ContactForm

def contact_view(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)  # Create a form instance with POST data
        if form.is_valid():
            # Process the data, for now, let's just print it
            print(form.cleaned_data)  # Print the data (you can save it to the database or send an email)

            # Redirect to a success page or display a success message
            return render(request, 'contact_success.html')  # You can create a success template
```

```
    else:
        form = ContactForm()  # Create an empty form when the page is
first loaded

    return render(request, 'contact.html', {'form': form})  # Render
the form on the contact page
```

Why We Use Views:

- Views handle the logic behind displaying and processing the form.
- They ensure the correct template is rendered and provide the data the template needs.

---

5. Create the Templates

Templates in Django allow you to define the HTML structure for rendering forms and displaying data.

`contact.html`

This template renders the form. It displays the fields and includes the necessary CSRF token for security.

```
<!-- contact.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Contact Us</title>
</head>
<body>
    <h1>Contact Us</h1>
    <form method="POST">
        {% csrf_token %}
        {{ form.as_p }}  <!-- Renders the form fields in paragraphs -->
        <button type="submit">Submit</button>
    </form>
```

```
</body>
</html>
```

- **Why We Use Templates**:
  - Templates help separate logic from presentation. The logic is handled in views, and the HTML is defined in templates.
  - We render the form dynamically using Django's `{{ form.as_p }}` to generate the form fields.

`contact_success.html`

This template will be displayed after a successful form submission.

```
<!-- contact_success.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Success</title>
</head>
<body>
    <h1>Thank you for contacting us!</h1>
    <p>We have received your message and will get back to you
shortly.</p>
</body>
</html>
```

---

6. Configure the URL

To make your form accessible, you need to configure the URL in Django's `urls.py`.

```
# urls.py in the contact app
from django.urls import path
from . import views
```

```
urlpatterns = [
    path('contact/', views.contact_view, name='contact'),  # Link to
the contact form view
]
```

This creates a URL pattern that links to the `contact_view` function when a user visits `/contact/`.

Add login and Registration

Modify the Views.py with login and register view

Modules to import

from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm # For user registration and login forms
from django.contrib.auth import login, logout # For handling user login and logout
from django.contrib import messages # For showing messages like error or success

`django.shortcuts.render`:

- This is used to render the HTML template and pass context data (like the form) to the template.

`django.shortcuts.redirect`:

- This function is used to redirect the user to another view after a successful action (like successful registration or login).

`django.contrib.auth.forms.UserCreationForm`:

- A built-in form provided by Django for creating a new user. It handles fields like username, password, and password confirmation.

`django.contrib.auth.forms.AuthenticationForm`:

- A built-in form for authenticating users based on their username and password.

`django.contrib.auth.login`:

- This is a function that logs the user in and stores their session information.

`django.contrib.auth.logout`:

- This function logs the user out, clearing the session data related to authentication.

`django.contrib.messages`:

- This module is used for sending messages (like error, info, or success messages) to the user. The `messages.error()` function is used here to show error messages if something goes wrong (e.g., invalid credentials or registration failure).

```
# User registration view
def register(request):
    # Check if the HTTP method is POST (form submission)
    if request.method == 'POST':
        # Create an instance of the UserCreationForm with the data from
the request
        form = UserCreationForm(request.POST)

        # Check if the form is valid
        if form.is_valid():
            # If the form is valid, save the new user and create an
instance of the user
            user = form.save()

            # Automatically log the user in after successful
registration
            login(request, user)

            # Redirect to the 'home' page after successful login
            return redirect('home')
        else:
            # If the form is not valid, show an error message
            messages.error(request, "Registration failed. Please try
again.")
    else:
        # If the request method is GET (form load), instantiate an
empty form
        form = UserCreationForm()
```

```
    # Render the registration page with the form
    return render(request, 'register.html', {'form': form})


# User login view

def loginview(request):
    # Check if the HTTP method is POST (form submission)
    if request.method == 'POST':
        # Create an instance of the AuthenticationForm with data from
the POST request
        form = AuthenticationForm(data=request.POST)

        # Check if the form is valid
        if form.is_valid():
            # If valid, get the user object from the form
            user = form.get_user()

            # Log the user in
            login(request, user)

            # Redirect to the 'home' page after successful login
            return redirect('home')
        else:
            # If the form is not valid, display an error message
            messages.error(request, "Invalid credentials. Please try
again.")
            # Re-render the login form with the error messages
            return render(request, 'login.html', {'form': form})
    else:
        # If the request method is GET (form load), create an empty
AuthenticationForm instance
        form = AuthenticationForm()

    # Render the login page with the form
    return render(request, 'login.html', {'form': form})


# User logout view
def logoutview(request):
    logout(request)  # Log the user out
    return redirect('home')  # Redirect to home page after logout
```

`AuthenticationForm`: This is a Django predefined form used to authenticate a user with a username and password.
`form.is_valid()`: This checks if the form fields are valid, meaning the entered credentials are correct.
`form.get_user()`: This method retrieves the user object associated with the credentials provided in the form.
`login(request, user)`: This logs the user in after successful authentication.
`messages.error()`: Displays error messages when the credentials provided are invalid.
`render(request, 'login.html', {'form': form})`: If the form is invalid, the form is rendered again with error messages for the user to correct.

Add Templates to Template Folder

1 . Create Register.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Register</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.
min.css">
</head>
<body>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-md-6">
                <h2 class="text-center mb-4">Register</h2>
                <form method="post">
                    {% csrf_token %}

                    <div class="form-group">
                        <label for="{{ form.username.id_for_label
}}">Username</label>
                        <input type="text" class="form-control"
                               name="{{ form.username.name }}"
                               id="{{ form.username.id }}"
                               value="{{ form.username.value }}"
                               required>
```

```
                        {% if form.username.errors %}
                            <div class="text-danger">
                                {{ form.username.errors }}
                            </div>
                        {% endif %}
                    </div>

                    <div class="form-group">
                        <label for="{{ form.password1.id_for_label
}}">Password</label>
                        <input type="password" class="form-control"
                               name="{{ form.password1.name }}"
                               id="{{ form.password1.id }}"
                               value="{{ form.password1.value }}"
                               required>
                        {% if form.password1.errors %}
                            <div class="text-danger">
                                {{ form.password1.errors }}
                            </div>
                        {% endif %}
                    </div>

                    <div class="form-group">
                        <label for="{{ form.password2.id_for_label
}}">Confirm Password</label>
                        <input type="password" class="form-control"
                               name="{{ form.password2.name }}"
                               id="{{ form.password2.id }}"
                               value="{{ form.password2.value }}"
                               required>
                        {% if form.password2.errors %}
                            <div class="text-danger">
                                {{ form.password2.errors }}
                            </div>
                        {% endif %}
                    </div>

                    <button type="submit" class="btn btn-primary
btn-block">Register</button>
                </form>
            </div>
        </div>
    </div>
```

```
</body>
</html>
```

2. Create login.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.
min.css">
</head>
<body>
    <div class="container mt-5">
        <h1>Login</h1>

        <!-- Display error messages -->
        {% if messages %}
        <div class="alert alert-danger">
            {% for message in messages %}
                <p>{{ message }}</p>
            {% endfor %}
        </div>
        {% endif %}

        <form method="post">
            {% csrf_token %}
            {{ form.as_p }}
            <button type="submit" class="btn
btn-primary">Login</button>
        </form>
    </div>
</body>
</html>
```

Modify Urls

```
path('register/', views.register, name = 'register'),
 path('login/', views.loginview, name = 'login'),
```

Note: The above lab is built using core Django functionalities for user authentication, including user registration, login, and logout. You can extend this implementation by adding additional features such as email verification, password reset, profile management, or social authentication using similar concepts and Django's built-in authentication framework.