MVT
Model View and Template ,

Model
Data access layer
RESPONSIBILITIES ;
DEFINING STRUCTURE OF YOUR DB

mODELS → PYTHON CLASSES

View
The View fetches data from the Model and processes it.

Template : passes the data to the template for rendering
Url dispatcher : Django match the url to the view

Response: The rendered HTML is sent back to the user as a response.

Key Differences Between MVT and MVC

- **MVC**:
  - **Controller**: Handles user input and updates the Model and View.
- **MVT**:
  - **View**: Acts as the Controller in MVC, handling user input and business logic.
  - **Template**: Acts as the View in MVC, handling presentation logic.

```
class Category(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    def __str__(self):
        return self.title
```

- Abstract
  Abstract painting is a non-representational art form that uses color, shape, and other elements to create an effect, rather than depicting reality: Purpose
- Realism
  Realism is an art movement that depicts everyday life in a naturalistic manner, often with close attention to detail. Realist paintings are characterized by their realistic appearance and their focus on representing life in its ideals or abstract concepts.
- realism
  real

Create a Django Project

Run the following commands:

bash
CopyEdit

```
django-admin startproject painting_shop
cd painting_shop
python manage.py startapp shop
```

Configure `settings.py`

Open `painting_shop/settings.py` and make these changes:

Register the App

python
CopyEdit

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'shop',  # Add your app here
]
```

Set Up Static and Media Files

python
CopyEdit

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / 'static']

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'
```

Configure URL Handling

Create the Models for Categories ,Paintings and Order

```
    from django.db import models  # Importing Django's ORM to define
database models
from django import forms  # Importing Django's forms module to create
form classes
from django.contrib.auth.models import User  # Importing the built-in
User model for authentication

# Create your models here.

# Category Model: Represents different painting categories (e.g.,
Abstract, Portraits)
class Category(models.Model):
    title = models.CharField(max_length=100)  # CharField for storing
category title with max length 100
    description = models.TextField()  # TextField for storing detailed
category description

    def __str__(self):
        return self.title  # Returns the category title as a string
representation

# Alternative:
# You can add `unique=True` to title to prevent duplicate categories
# title = models.CharField(max_length=100, unique=True)

# Painting Model: Represents individual paintings under various
categories
class Painting(models.Model):
    title = models.CharField(max_length=100)  # Name of the painting
    artist = models.CharField(max_length=100)  # Name of the artist who
created the painting
    category = models.ForeignKey(
        Category, on_delete=models.CASCADE, related_name='paintings'
    )  # ForeignKey establishes a Many-to-One relationship with
Category
    # CASCADE ensures that if a category is deleted, all associated
paintings are deleted too
```

```
    created_at = models.DateTimeField(auto_now_add=True)  # Stores
painting creation date and time
    price = models.DecimalField(max_digits=10, decimal_places=2)  #
Price with two decimal places
    image = models.ImageField(upload_to='media/images/', blank=True,
null=True)
    # ImageField stores image file paths, images will be uploaded to
'media/images/'
    # blank=True allows image to be optional in forms
    # null=True allows storing NULL values in the database

    def __str__(self):
        return self.title  # Returns the painting title when displayed

# Alternative:
# You can use PositiveIntegerField for price to ensure non-negative
values
# price = models.PositiveIntegerField()

# ContactForm: A Django form for user inquiries or messages
class ContactForm(forms.Form):
    name = forms.CharField(
        max_length=100,
        widget=forms.TextInput(attrs={'class': 'form-control'})
    )  # Name field with max length 100, styled with Bootstrap
    email = forms.EmailField(
        widget=forms.EmailInput(attrs={'class': 'form-control'})
    )  # Email field with EmailInput widget for validation
    message = forms.CharField(
        widget=forms.Textarea(attrs={'class': 'form-control', 'rows':
5})
    )  # Message field with a textarea widget, 5 rows tall

# Alternative:
# Instead of a Django Form class, you could use Django ModelForm if
storing messages in a model
# class ContactMessage(models.Model):
#     name = models.CharField(max_length=100)
#     email = models.EmailField()
#     message = models.TextField()

# Order Model: Represents an order placed by a user for a specific
painting
```

```
class Order(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    # Links the order to a specific user (Many-to-One with User)
    # CASCADE ensures that if a user is deleted, all their orders are
deleted too
    painting = models.ForeignKey(Painting, on_delete=models.CASCADE)
    # Links the order to a specific painting (Many-to-One with
Painting)
    order_date = models.DateTimeField(auto_now_add=True)
    # Stores the date and time when the order was placed
    quantity = models.PositiveIntegerField(default=1)
    # Quantity of paintings ordered, defaults to 1

    def __str__(self):
        return f"Order by {self.user.username} for
{self.painting.title}"
    # String representation of the order

# Alternative:
# You could add a status field to track the order progress (e.g.,
Pending, Shipped, Delivered)
# status = models.CharField(max_length=20, choices=[('Pending',
'Pending'), ('Shipped', 'Shipped'), ('Delivered', 'Delivered')],
default='Pending')
```

Add changes to admin.py

```
from django.contrib import admin
from .models import Category, Painting
# Register your models here.

admin.site.register(Category)
admin.site.register(Painting)
```

Create a template/core folder in app

Create 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 'styling.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="#">Home</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="#">Features</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="#">Pricing</a>
```

```
                </li>
                <li class="nav-item">
                  <a class="nav-link disabled" href="#">Disabled</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="{% url 'paintinglist'
%}">Painting</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="{% url 'categorylist'
%}">Category</a>
              </ul>
            </div>
        </nav>
    </header>

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

</body>
</html>
```

Create home.html

```
{% extends "core/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 'paintinglist' %}" class="btn btn-light
btn-lg">Browse All Paintings</a>
  </div>
</section>

{% endblock body %}
```

Create Categorylist.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Categories</title>
</head>
<body>
    <h1>Categories</h1>
    <ul>
        {% for category in categories %}
            <li>
                {{ category.title }}
```

```
                <p>{{ category.description }}</p>
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

Create paintinglist.html

```
<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>
                <a href="{% url 'order_painting' painting.id %}"
class="btn btn-primary order-btn">Order Now</a>
            </div>
        </div>
    </div>
    {% endfor %}
</div>
```

Modify View.py

```
# Import necessary modules and classes
from django.contrib.auth import authenticate, login, logout  #
Authentication methods for user login/logout
from django.shortcuts import redirect, render  # Shortcuts for
rendering templates and redirecting users
from .models import Category, Painting, ContactForm  # Import models to
interact with the database
```

```python
from django.contrib.auth.forms import UserCreationForm,
AuthenticationForm  # Predefined forms for user registration and login
from django.contrib import messages  # For adding messages
(success/error) to the template
from django.shortcuts import render, get_object_or_404, redirect  #
Shortcuts for rendering, fetching objects safely, and redirecting
from .forms import OrderForm  # Import a custom form for handling
orders

# Home view - renders the home page
def home(request):
    context = {
        "variable": 'value',  # Context to pass variables to the
template
    }
    return render(request, 'core/home.html', context)  # Render the
template and pass context

# About page - renders the about page
def about(request):
    return render(request, 'core/about.html')  # Simply renders the
about template

# Categories page - shows a list of all categories
def categories(request):
    categories = Category.objects.all()  # Fetch all category objects
from the database
    return render(request, 'core/categorylist.html', {'categories':
categories})  # Render the category list template with the categories

# Painting list page - shows a list of all paintings
def paintinglist(request):
    paintings = Painting.objects.all()  # Fetch all painting objects
from the database
    return render(request, 'core/paintinglist.html', {'paintings':
paintings})  # Render the painting list template with the paintings

# Contact form page - renders a contact form
def contactform(request):
    form = ContactForm()  # Instantiate an empty contact form
    return render(request, 'core/contact.html', {'form': form})  #
Render the contact page with the form
```

```python
# User registration view - handles user registration
def register(request):
    if request.method == 'POST':  # If the request method is POST (form submission)
        form = UserCreationForm(request.POST)  # Instantiate the UserCreationForm with POST data
        if form.is_valid():  # Check if the form is valid
            user = form.save()  # Save the new user to the database
            login(request, user)  # Log the user in immediately after successful registration
            return redirect('home')  # Redirect the user to the home page after registration
        else:
            messages.error(request, "Registration failed. Please try again.")  # Add an error message if form is invalid
    else:
        form = UserCreationForm()  # Instantiate an empty form for GET requests
    return render(request, 'core/register.html', {'form': form})  # Render the registration template with the form

# User login view - handles user login
def loginview(request):
    if request.method == 'POST':  # If the request method is POST (form submission)
        form = AuthenticationForm(data=request.POST)  # Instantiate the AuthenticationForm with POST data
        if form.is_valid():  # Check if the form is valid
            user = form.get_user()  # Get the user object from the form
            login(request, user)  # Log the user in
            return redirect('home')  # Redirect the user to the home page after login
        else:
            messages.error(request, "Invalid credentials. Please try again.")  # Show an error message for invalid credentials
            return render(request, 'core/login.html', {'form': form}) # Return the form with error message
    else:
        form = AuthenticationForm()  # Instantiate an empty form for GET requests

    return render(request, 'core/login.html', {'form': form})  # Render the login template with the form
```

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

# Order painting view - handles placing an order for a painting
def order_painting(request, painting_id):
    painting = get_object_or_404(Painting, id=painting_id)  # Fetch the
painting object by ID or return 404 if not found

    if request.method == 'POST':  # If the request method is POST (form
submission)
        form = OrderForm(request.POST)  # Instantiate the OrderForm
with POST data
        if form.is_valid():  # Check if the form is valid
            order = form.save(commit=False)  # Create an order instance
but don't save to the database yet
            order.user = request.user  # Associate the order with the
logged-in user
            order.painting = painting  # Associate the order with the
selected painting
            order.save()  # Save the order to the database
            messages.success(request, f"Order for {painting.title} has
been placed!")  # Show a success message
            return redirect('paintinglist')  # Redirect to the painting
list page after placing the order
        else:
            messages.error(request, "There was an error with your
order. Please try again.")  # Show an error message for invalid form
submission
    else:
        form = OrderForm()  # Instantiate an empty form for GET
requests

    return render(request, 'core/order_painting.html', {'form': form,
'painting': painting})  # Render the order painting page with the form
and selected painting
```

Create categorylist.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Categories</title>
</head>
<body>
    <h1>Categories</h1>
    <ul>
        {% for category in categories %}
            <li>
               {{ category.title }}
                <p>{{ category.description }}</p>
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

Create order painting.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Order {{ painting.title }}</title>
</head>
<body>
    <h1>Order {{ painting.title }}</h1>

    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Place Order</button>
    </form>

    <a href="{% url 'paintinglist' %}">Back to Painting List</a>
</body>
</html>
```

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>
```

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>
```

Create a file forms.py for order

```
from django import forms
from .models import Order

class OrderForm(forms.ModelForm):
    class Meta:
        model = Order
        fields = ['quantity']  # We only need quantity for simplicity
```

Modify app urls

```
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


admin.site.site_header = "UMSRA Admin"
```

```
admin.site.site_title = "UMSRA Admin Portal"
admin.site.index_title = "Welcome to UMSRA Researcher Portal"

urlpatterns = [
   path('admin/', admin.site.urls),
   path('home/', views.home, name = 'home'),
   path('categories/', views.categories, name = 'categorylist'),
   path('paintings/', views.paintinglist, name = 'paintinglist'),
   path('contact/', views.contactform, name = 'contact'),
   path('register/', views.register, name = 'register'),
   path('login/', views.loginview, name = 'login'),
    path('order/<int:painting_id>/', views.order_painting,
name='order_painting'),  # New path for ordering paintings
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Project level urls

```
from django.contrib import admin
from django.urls import include, path
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('' , include('core.urls')),
]
# + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```