/websiteproject/forum/apps/object_finder/admin.py

```
# Import required models and admin module
from .models import Tag, Post
from django.contrib import admin


# Register models for admin interface
admin.site.register(Tag)  # Enable Tag model management in admin
admin.site.register(Post) # Enable Post model management in admin
```

/websiteproject/forum/apps/object_finder/apps.py

```
from django.apps import AppConfig


class ObjectFinderConfig(AppConfig):
   """Configuration class for the object_finder app"""
   default_auto_field = 'django.db.models.BigAutoField'       # Specifies the primary
key type
   name = 'apps.object_finder'                                # The Python package
name of the app
```

/websiteproject/forum/apps/object_finder/form_tags.py

```
from django import template

# Create template library instance
register = template.Library()


@register.filter(name='add_class')
def add_class(value, css_class):
   """Add CSS class to form field widget"""
   return value.as_widget(attrs={'class': css_class})
```

/websiteproject/forum/apps/object_finder/forms.py

```
# Import required models and forms
from .models import Comment
from django import forms
from .models import Post
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User


class SignUpForm(UserCreationForm):
   """Form for user registration with optional profile fields"""
   email = forms.EmailField(max_length=254, required=False, help_text='Optional.')
   first_name = forms.CharField(max_length=30, required=False, help_text='Optional.')
   last_name = forms.CharField(max_length=30, required=False, help_text='Optional.')

   class Meta:
       model = User
       fields = ('username', 'email', 'first_name', 'last_name', 'password1',
'password2')


class PostForm(forms.ModelForm):
   """Form for creating new posts"""
   class Meta:
       model = Post
       fields = ['title']
       widgets = {
           'title': forms.TextInput(attrs={
               'placeholder': 'Enter post title',
               'class': 'form-control'
           }),
       }


class CommentForm(forms.ModelForm):
   """Form for adding comments to posts, supports both authenticated and anonymous
users"""
   anonymous_name = forms.CharField(
       max_length=255, required=False, label='Your Name (optional)')

   class Meta:
       model = Comment
       fields = ['content', 'anonymous_name']

   content = forms.CharField(
       widget=forms.Textarea(attrs={
                             'class': 'form-control', 'rows': 3, 'placeholder': 'Enter
your comment here...'}),
       label='',
   )

   def __init__(self, *args, **kwargs):
       """Initialize form and remove anonymous name field if user is authenticated"""
       user_authenticated = kwargs.pop('user_authenticated', False)
       super(CommentForm, self).__init__(*args, **kwargs)
       if user_authenticated:
           self.fields.pop('anonymous_name', None)
```

/websiteproject/forum/apps/object_finder/models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models import JSONField

# Model for storing user uploaded images
class Image(models.Model):
    image = models.ImageField(upload_to='images', default='default.jpg')


# Model for user profiles with associated image
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.OneToOneField(Image, on_delete=models.CASCADE, default=1)


# Model for post comments, can be from registered users or anonymous
class Comment(models.Model):
    content = models.TextField()
    author = models.ForeignKey(
        User, on_delete=models.CASCADE, null=True, blank=True)
    anonymous_name = models.CharField(max_length=255, null=True, blank=True)
    date_posted = models.DateTimeField(auto_now_add=True)
    post = models.ForeignKey(
        "Post", related_name='comments', on_delete=models.CASCADE)
    is_solved = models.BooleanField(default=False)

    def __str__(self):
        if self.author:
            return f'Comment by {self.author} on {self.post.title}'
        else:
            return f'Anonymous comment on {self.post.title}'

    objects = models.Manager()


# Model for post categorization tags
class Tag(models.Model):
    name = models.CharField(max_length=100)
    tag_id = models.CharField(max_length=100, unique=True)

    objects = models.Manager()


# Main post model containing the post content and metadata
class Post(models.Model):
    title = models.CharField(max_length=100)
    content_delta = models.JSONField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    date_posted = models.DateTimeField(auto_now_add=True)
    tags = models.ManyToManyField(Tag, blank=True)
    attributes = JSONField(default=dict)

    objects = models.Manager()
```

/websiteproject/forum/apps/object_finder/tests.py

```
from django.test import TestCase, Client
from django.contrib.auth.models import User
from django.urls import reverse
from django.core.files.uploadedfile import SimpleUploadedFile
from .models import Post

class UserTestCase(TestCase):
    """Test cases for User model functionality"""
    def test_create_user(self):
        """Test user creation and verification should return TRUE"""
        # Create a test user with username, email and password
        test_user = User.objects.create_user(
            username='testuser',
            email='test@gmail.com',
            password='testPass123..'
        )
```

```python
    # Verify user attributes match what was provided
    self.assertEqual(test_user.username, 'testuser')
    self.assertEqual(test_user.email, 'test@gmail.com')
    self.assertTrue(test_user.check_password('testPass123..'))

    # Verify user was properly saved to database
    saved_user = User.objects.get(username='testuser')
    self.assertEqual(saved_user.username, test_user.username)
    self.assertEqual(saved_user.email, test_user.email)

    def test_post_creation_count(self):
        """Test that creating a post via form URL increases post count"""
        # Get initial post count
        initial_count = Post.objects.count()

        # Set up test client and user
        client = Client()
        user = User.objects.create_user(
            username='postuser',
            email='post@test.com',
            password='testPass123..'
        )
        client.login(username='postuser', password='testPass123..')
        # Make POST request to form URL to create post
        post_data = {
            'title': 'Test Post',
            'content_delta': '{}',
            'header_image': SimpleUploadedFile(
                name='test_image.jpg',
                content=b'',
                content_type='image/jpeg'
            )
        }
        response = client.post('/form', post_data)

        # Verify post count increased by 1
        self.assertEqual(Post.objects.count(), initial_count + 1)

        # Verify post was created with correct data
        latest_post = Post.objects.latest('date_posted')
        self.assertEqual(latest_post.title, 'Test Post')
```

```
            self.assertEqual(latest_post.author, user)

        def test_comment_creation(self):
            """Test that creating a comment on a post works"""
            # Create test user and post
            user = User.objects.signup(
                username='commentuser',
                password='testPass123..'
            )

            post = Post.objects.create(
                title='Test Post',
                content_delta='{}',
                author=user
            )

            # Create a comment on the post
            comment = Comment.objects.crate(
                content='Test comment',
                post=post,
                author=user
            )

            # Verify comment was created and associated with post
            self.assertEqual(Comment.objects.count(), 1)
            self.assertEqual(post.comments.count(), 1)
            self.assertEqual(comment.content, 'Test comment')
            self.assertEqual(comment.author, user)
            self.assertEqual(comment.post, post)

    def test_index_page_accessible(self):
        """Test that the index page is accessible to all users"""
        # Create test client
        client = Client()
        # Make GET request to index page
        response = client.get('/')
        # Verify successful response
        self.assertEqual(response.status_code, 200)
```

/websiteproject/forum/apps/object_finder/urls.py

```
# Import required Django modules and views
from django.urls import path
```

```python
from . import views
from django.contrib.auth import views as auth_views

urlpatterns = [
   path('', views.index, name='index'),
# Homepage/index view
   path('form', views.form, name='form'),
# Form for creating new posts
   path('profile/<int:user_id>/', views.profile_view, name='profile'),
# User profile view
   path('login/', auth_views.LoginView.as_view(template_name='login.html'),
name='login'),      # Login page
   path('logout/', auth_views.LogoutView.as_view(), name='logout'),
# Logout functionality
   path('signup/', views.signup_view, name='signup'),
# User registration
   path('search/', views.search_posts, name='search_posts'),
# Search posts functionality
   path('view_post/<int:post_id>', views.view_post, name='view_post'),
# View single post
   path('post/<int:post_id>/<int:comment_id>/update_is_solved/',
        views.update_is_solved, name='update_is_solved'),
# Update solved status of comments
]
```

/websiteproject/forum/apps/object_finder/views.py

```python
import json
from django.contrib.auth.forms import UserCreationForm
from django.shortcuts import render, get_object_or_404, redirect
from .forms import PostForm, CommentForm, SignUpForm
from .models import Post, Tag, Comment
from django.contrib.auth.decorators import login_required
from django.db.models import Q
import re
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt



@login_required
```

```
@csrf_exempt
def update_is_solved(request, post_id, comment_id):
   """Updates the solved status of a comment on a post"""
   if request.method == 'POST':
       try:
           post = get_object_or_404(Post, id=post_id)
           if request.user != post.author:
               return JsonResponse({'success': False, 'error': 'Permission denied.'},
status=403)

           comment = get_object_or_404(Comment, id=comment_id)
           is_solved = not comment.is_solved
           post.comments.update(is_solved=False)
           comment.is_solved = is_solved
           comment.save()

           comments = post.comments.all().order_by('-date_posted')
           return render(request, 'object_finder/comments.html', {'comments':
comments, 'post': post})

       except Exception as e:
           return JsonResponse({'success': False, 'error': str(e)}, status=400)
   else:
       return JsonResponse({'success': False, 'error': 'Invalid request.'},
status=400)


def profile_view(request, user_id):
   """Displays posts by a specific user"""
   user_posts = Post.objects.filter(author__id=user_id)
   return render(request, 'object_finder/profile.html', {'user_posts': user_posts})


def index(request):
   """Displays the 10 most recent posts"""
   posts = Post.objects.all().order_by('-date_posted')[0:10]
   return render(request, 'object_finder/index.html', {'view_name': 'Recent
Discussions', 'posts': posts})


@login_required
def search_posts(request):
```

```python
"""Searches posts based on query and attributes"""
query = request.GET.get('q', '').strip()

attributes = {}
for key, value in request.GET.items():
    if key.startswith('attribute_') and value.strip():
        attribute_name = key[len('attribute_'):]
        attr_value = value.strip()

        unit_name = f'unit_{attribute_name}'
        unit_value = request.GET.get(unit_name, "").strip()

        attributes[attribute_name] = {
            'value': attr_value,
            'unit': unit_value
        }

has_query_condition = False
has_attribute_condition = False

posts = Post.objects.all()

if query:
    query_condition = Q(title__icontains=query) | Q(
        tags__name__icontains=query)
    posts = posts.filter(query_condition)
    has_query_condition = True

if attributes:
    filtered_posts = []
    for p in posts:
        match_all = True
        for attr_id, attr_data in attributes.items():
            search_val = attr_data['value']
            search_unit = attr_data['unit']

            attribute_data = p.attributes.get(attr_id, {})
            if not bool(attribute_data):
                match_all = False
                break

            db_val = attribute_data.get("value", "")
```

```
                db_unit = attribute_data.get("unit", "")

                if search_val.lower() not in db_val.lower():
                    match_all = False
                    break

                if search_unit and search_unit != db_unit:
                    match_all = False
                    break

            if match_all:
                filtered_posts.append(p)
        posts = filtered_posts
        has_attribute_condition = True

    if not has_query_condition and not has_attribute_condition:
        posts = Post.objects.none()

    posts = sorted(posts, key=lambda x: x.date_posted, reverse=True)

    return render(request, 'object_finder/index.html', {
        'view_name': 'Search Results',
        'posts': posts,
        'query': query
    })


@login_required
def form(request):
    """Handles creation of new posts"""
    form = PostForm()
    if request.method == 'POST':
        form = PostForm(request.POST)
        if form.is_valid():
            post = form.save(commit=False)
            content_delta = request.POST.get('content_delta', None)
            tags_data = request.POST.get('tags', '[]')

            if content_delta:
                try:
                    content_delta = json.loads(content_delta)
                    tags = json.loads(tags_data)
```

```
except json.JSONDecodeError:
    form.add_error(None, 'Invalid content or tags data.')
    return render(request, 'object_finder/form.html', {
        'form': form,
    })

# Check if post contains at least one image
contains_image = False
for op in content_delta.get('ops', []):
    insert_content = op.get('insert', {})
    if isinstance(insert_content, dict) and 'image' in insert_content:
        contains_image = True
        break

if not contains_image:
    form.add_error(
        None, 'You must include at least one image in the content.')
    return render(request, 'object_finder/form.html', {
        'form': form,
    })

# Process attributes
attributes = {}
for key, value in request.POST.items():
    if key.startswith('attribute_'):
        attribute_name = key[len('attribute_'):]
        if value.strip():
            attributes[attribute_name] = {
                'value': value.strip()}

            unit_name = f'unit_{attribute_name}'
            unit_value = request.POST.get(unit_name, None)
            if unit_value:
                attributes[attribute_name]['unit'] = unit_value

post.attributes = attributes
post.author = request.user
post.content_delta = content_delta
post.save()

# Process tags
for tag_data in tags:
```

```
                    tag_id = tag_data.get('id')
                    tag_label = tag_data.get('label')

                    tag, created = Tag.objects.get_or_create(
                        tag_id=tag_id,
                        name=tag_label
                    )
                    post.tags.add(tag)

                return redirect('/')
            else:
                form.add_error(None, 'Content is required.')
            return render(request, 'object_finder/form.html', {
                'form': form,
            })
    return render(request, 'object_finder/form.html', {
        'form': form,
    })


def view_post(request, post_id):
    """Displays a single post and handles comment submission"""
    post = get_object_or_404(Post, id=post_id)

    if request.user.is_authenticated:
        comment_form = CommentForm(user_authenticated=True)
    else:
        comment_form = CommentForm(user_authenticated=False)

    if request.method == 'POST':
        form_type = request.POST.get('form_type')

        if form_type == 'comment_form':
            if request.user.is_authenticated:
                comment_form = CommentForm(
                    request.POST, user_authenticated=True)
            else:
                comment_form = CommentForm(
                    request.POST, user_authenticated=False)

            if comment_form.is_valid():
                comment = comment_form.save(commit=False)
```

```python
                comment.post = post

                if request.user.is_authenticated:
                    post_as_anonymous = 'post_as_anonymous' in request.POST
                    if post_as_anonymous:
                        # User chose to post anonymously
                        comment.author = None
                    else:
                        comment.author = request.user
                else:
                    # Anonymous user
                    comment.author = None
                    comment.anonymous_name = comment_form.cleaned_data.get(
                        'anonymous_name')

                comment.save()
                return redirect('view_post', post_id=post.id)

    comments = post.comments.all().order_by('-date_posted')
    tags = post.tags.all()
    attribute_names = post.attributes
    is_solved = any([comment.is_solved for comment in comments])

    return render(request, 'object_finder/view_post.html', {
        'post': post,
        'comment_form': comment_form,
        'comments': comments,
        'tags': tags,
        'attribute_names': attribute_names,
        'is_solved': is_solved
    })

def signup_view(request):
    """Handles user registration"""
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.username = form.cleaned_data.get('username')
            user.email = form.cleaned_data.get('email')
            user.first_name = form.cleaned_data.get('first_name')
            user.last_name = form.cleaned_data.get('last_name')
            user.set_password(form.cleaned_data.get('password1'))
            user.save()
            return redirect('login')
    else:
        form = SignUpForm()
    return render(request, 'registration/signup.html', {'form': form})
```

/websiteproject/forum/apps/object_finder/templatetags/form_tags.py

```
from django import template

# Create template library instance
register = template.Library()


@register.filter(name='add_class')
def add_class(value, css_class):
    """Add CSS class to a form field widget"""
    return value.as_widget(attrs={'class': css_class})
```

/websiteproject/forum/apps/object_finder/templates/registration/login.html

```
{% extends '../object_finder/base.html' %}
{% load form_tags %}

{% block content %}
<!-- Main login container -->
<div class="container mt-5">
    <h2>Login</h2>
    <div class="row justify-content-center">
        <div class="col-md-6">
            <!-- Display validation errors if any -->
            {% if form.errors %}
                <div class="alert alert-danger">
                    Please correct the error{{ form.errors|pluralize }} below.
                </div>
            {% endif %}
            <!-- Login form -->
            <form method="post" novalidate>
                {% csrf_token %}
                <div class="mb-3">
                    <label for="id_username" class="form-label">Username</label>
                    {{ form.username|add_class:"form-control" }}
                </div>
                <div class="mb-3">
                    <label for="id_password" class="form-label">Password</label>
                    {{ form.password|add_class:"form-control" }}
                </div>
                <button type="submit" class="btn btn-primary">Login</button>
            </form>
            <!-- Sign up link -->
            <p class="mt-3">
                Don't have an account? <a href="{% url 'signup' %}">Sign up here</a>.
            </p>
        </div>
    </div>
</div>
{% endblock content %}
```

/websiteproject/forum/apps/object_finder/templates/registration/signup.html

```
{% extends '../object_finder/base.html' %}
{% load form_tags %}

{% block content %}
<!-- Main signup container -->
<div class="container mt-5">
    <h2>Sign Up</h2>
    <div class="row justify-content-center">
        <div class="col-md-6">
            <!-- Display validation errors if any -->
            {% if form.errors %}
                <div class="alert alert-danger">
                    {% if form.username.errors %}
                        <p>Username error: {{ form.username.errors|striptags }}</p>
                    {% endif %}
                    {% if form.password1.errors %}
                        <p>Password must:</p>
                        <ul>
                            <li>Be at least 8 characters long</li>
```

```
                           <li>Contain at least one uppercase letter</li>
                           <li>Contain at least one number</li>
                           <li>Not be too similar to your username</li>
                           <li>Not be a commonly used password</li>
                       </ul>
                   {% endif %}
                   {% if form.password2.errors %}
                       <p>Password confirmation error: {{
form.password2.errors|striptags }}</p>
                   {% endif %}
                   {% if form.email.errors %}
                       <p>Email error: {{ form.email.errors|striptags }}</p>
                   {% endif %}
               </div>
           {% endif %}
           <!-- Signup form -->
           <form method="post" novalidate>
               {% csrf_token %}
               <div class="mb-3">
                   <label for="id_username" class="form-label">Username</label>
                   {{ form.username|add_class:"form-control" }}
                   <small class="text-muted">Required. 150 characters or
fewer.</small>
               </div>
               <div class="mb-3">
                   <label for="id_email" class="form-label">Email (Optional)</label>
                   {{ form.email|add_class:"form-control" }}
                   <small class="text-muted">Please enter a valid email
address.</small>
               </div>
               <div class="mb-3">
                   <label for="id_first_name" class="form-label">First Name
(Optional)</label>
                   {{ form.first_name|add_class:"form-control" }}
               </div>
               <div class="mb-3">
                   <label for="id_last_name" class="form-label">Last Name
(Optional)</label>
                   {{ form.last_name|add_class:"form-control" }}
               </div>
               <div class="mb-3">
                   <label for="id_password1" class="form-label">Password</label>
```

```
                    {{ form.password1|add_class:"form-control" }}
                    <small class="text-muted">Your password must be at least 8
characters long and contain letters and numbers.</small>
                </div>
                <div class="mb-3">
                    <label for="id_password2" class="form-label">Confirm
Password</label>
                    {{ form.password2|add_class:"form-control" }}
                </div>
                <button type="submit" class="btn btn-primary">Sign Up</button>
            </form>
            <!-- Login link -->
            <p class="mt-3">
                Already have an account? <a href="{% url 'login' %}">Login here</a>.
            </p>
        </div>
    </div>
</div>
{% endblock content %}
```

/websiteproject/forum/apps/object_finder/templates/object_finder/attributes.html

```
<!-- Basic physical attributes -->
<div class="form-group">
    <label for="attribute-material">Material</label>
    <input type="text" class="form-control" id="attribute-material"
name="attribute_material"></input>
</div>

<div class="form-group">
    <label for="attribute-colour">Colour</label>
    <input type="text" class="form-control" id="attribute-colour"
name="attribute_colour"></input>
</div>

<!-- Shape selection with predefined options -->
<div class="form-group">
    <label for="attribute-shape">Shape</label>
    <select class="form-control" id="attribute-shape" name="attribute_shape" multiple>
        <option value="sphere">sphere</option>
        <option value="cube">cube</option>
        <option value="cylinder">cylinder</option>
```

```html
       <option value="pyramid">pyramid</option>
       <option value="cone">cone</option>
   </select>
</div>

<!-- Measurement attributes with units -->
<div class="form-group">
   <label for="attribute-weight">Weight</label>
   <div class="d-flex">
       <input type="number" class="form-control flex-grow-1" id="attribute-weight"
name="attribute_weight">
       <select class="form-control ms-2" id="unit-weight" name="unit_weight"
style="width:80px; flex-shrink:0;">
           <option value="g">g</option>
           <option value="kg">kg</option>
       </select>
   </div>
</div>

<div class="form-group">
   <label for="attribute-width">Width</label>
   <div class="d-flex">
       <input type="number" class="form-control flex-grow-1" id="attribute-width"
name="attribute_width">
       <select class="form-control ms-2" id="unit-width" name="unit_width"
style="width:80px; flex-shrink:0;">
           <option value="mm">mm</option>
           <option value="cm">cm</option>
           <option value="m">m</option>
           <option value="km">km</option>
       </select>
   </div>
</div>

<div class="form-group">
   <label for="attribute-height">Height</label>
   <div class="d-flex">
       <input type="number" class="form-control flex-grow-1" id="attribute-height"
name="attribute_height">
       <select class="form-control ms-2" id="unit-height" name="unit_height"
style="width:80px; flex-shrink:0;">
           <option value="mm">mm</option>
           <option value="cm">cm</option>
           <option value="m">m</option>
           <option value="km">km</option>
       </select>
   </div>
</div>

<div class="form-group">
   <label for="attribute-length">Length</label>
   <div class="d-flex">
       <input type="number" class="form-control flex-grow-1" id="attribute-length"
name="attribute_length">
       <select class="form-control ms-2" id="unit-length" name="unit_length"
style="width:80px; flex-shrink:0;">
           <option value="mm">mm</option>
           <option value="cm">cm</option>
           <option value="m">m</option>
           <option value="km">km</option>
       </select>
   </div>
</div>

<div class="form-group">
   <label for="attribute-volume">Volume</label>
   <div class="d-flex">
       <input type="number" class="form-control flex-grow-1" id="attribute-volume"
name="attribute_volume">
       <select class="form-control ms-2" id="unit-volume" name="unit_volume"
style="width:80px; flex-shrink:0;">
           <option value="ml">ml</option>
           <option value="l">l</option>
           <option value="m3">m³</option>
       </select>
   </div>
</div>

<!-- Physical characteristics -->
<div class="form-group">
   <label for="attribute-description_of_parts">Description of Parts</label>
   <input type="text" class="form-control" id="attribute-description_of_parts"
name="attribute_description_of_parts"></input>
</div>
```

```
<div class="form-group">
   <label for="attribute-hardness">Hardness</label>
   <input type="text" class="form-control" id="attribute-hardness"
name="attribute_hardness"></input>
</div>

<div class="form-group">
   <label for="attribute-pattern">Pattern</label>
   <input type="text" class="form-control" id="attribute-pattern"
name="attribute_pattern"></input>
</div>

<div class="form-group">
   <label for="attribute-texture">Texture</label>
   <input type="text" class="form-control" id="attribute-texture"
name="attribute_texture"></input>
</div>

<!-- Object status and identification -->
<div class="form-group">
   <label for="attribute-condition">Condition</label>
   <input type="text" class="form-control" id="attribute-condition"
name="attribute_condition"></input>
</div>

<div class="form-group">
   <label for="attribute-brand">Brand</label>
   <input type="text" class="form-control" id="attribute-brand"
name="attribute_brand"></input>
</div>

<div class="form-group">
   <label for="attribute-print">Print</label>
   <input type="text" class="form-control" id="attribute-print"
name="attribute_print"></input>
</div>

<!-- Sensory and functional attributes -->
<div class="form-group">
   <label for="attribute-functionality">Functionality</label>
   <input type="text" class="form-control" id="attribute-functionality"
name="attribute_functionality"></input>
</div>

<div class="form-group">
   <label for="attribute-smell">Smell</label>
   <input type="text" class="form-control" id="attribute-smell"
name="attribute_smell"></input>
</div>

<div class="form-group">
   <label for="attribute-taste">Taste</label>
   <input type="text" class="form-control" id="attribute-taste"
name="attribute_taste"></input>
</div>

<!-- Location and temporal information -->
<div class="form-group">
   <label for="attribute-location">Location</label>
   <input type="text" class="form-control" id="attribute-location"
name="attribute_location"></input>
</div>

<div class="form-group">
   <label for="attribute-time_period">Time Period</label>
   <input type="text" class="form-control" id="attribute-time_period"
name="attribute_time_period"></input>
</div>

<div class="form-group">
   <label for="attribute-year">Year</label>
   <input type="number" class="form-control" id="attribute-year"
name="attribute_year">
</div>

<!-- Commercial information -->
<div class="form-group">
   <label for="attribute-price">Price</label>
   <div class="d-flex">
       <input type="number" class="form-control flex-grow-1" id="attribute-price"
name="attribute_price">
```

```
       <select class="form-control ms-2" id="unit-price" name="unit_price"
style="width:80px; flex-shrink:0;">
           <option value="USD">USD</option>
           <option value="EUR">EUR</option>
           <option value="GBP">GBP</option>
       </select>
   </div>
</div>

<div class="form-group">
   <label for="attribute-part_of">Part Of</label>
   <input type="text" class="form-control" id="attribute-part_of"
name="attribute_part_of"></input>
</div>
```

/websiteproject/forum/apps/object_finder/templates/object_finder/base.html

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Forum</title>
   <!-- External CSS Dependencies -->
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
   <!-- Font Awesome CSS -->
   <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.min.css"
integrity="sha512-Kc323vGBEqzTmouAECnVceyQqyqdsSiqLQISBL29aUW4U/M7pSPA/gEUZQqv1cwx4OnY
xTxve5UMg5GT6L4JJg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
   <!-- Custom CSS -->
   <link href="{% static 'css/styles.css' %}" rel="stylesheet">
   <!-- External JS Dependencies -->
   <script src="https://cdn.quilljs.com/1.3.7/quill.js"></script>
   <script src="https://unpkg.com/htmx.org@2.0.3"></script>
</head>
<body>
```

```
   <!-- Navigation Bar -->
   <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
     <div class="container-fluid">
       <a class="navbar-brand" href="/">(My)stery Object Finder</a>
       <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
         <span class="navbar-toggler-icon"></span>
       </button>
       <!-- Navbar links -->
       <div class="collapse navbar-collapse" id="navbarNavSearch">
         <!-- Main Navigation Links -->
         <ul class="navbar-nav me-auto">
           <li class="nav-item">
             <a class="nav-link {% if request.path == '/' %}active{% endif %}"
aria-current="page" href="/">Home</a>
           </li>
           <li class="nav-item">
             <a class="nav-link" href="/">Categories</a>
           </li>
           <li class="nav-item">
             <a class="nav-link" href="/">About</a>
           </li>
           <!-- Additional nav items -->
         </ul>

         <!-- Search Form -->
         <form class="d-flex" method="get" action="{% url 'search_posts' %}">
           <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search" name="q">
           <button class="btn btn-outline-success me-2" type="submit">Search</button>
           <button type="button" class="btn btn-outline-primary"
data-bs-toggle="modal" data-bs-target="#advancedSearchModal">
             Advanced Search
           </button>
         </form>

         <!-- User Authentication Menu -->
         <ul class="navbar-nav ms-auto">
           {% if user.is_authenticated %}
           <li class="nav-item">
             <a class="nav-link" href="{% url 'form' %}">
```

```
              <i class="fas fa-edit"></i> Create Post
            </a>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="userDropdown"
role="button" data-bs-toggle="dropdown" aria-expanded="false">
              <i class="fas fa-user"></i> {{ user.username }}
            </a>
            <ul class="dropdown-menu dropdown-menu-end"
aria-labelledby="userDropdown">
              <li>
                <a class="dropdown-item" href="{% url 'profile' user.id %}">
                  <i class="fas fa-user-circle"></i> Profile
                </a>
              </li>
              <li>
                <a class="dropdown-item" href="{% url 'profile' user.id %}">
                    <form method="post" action="{% url 'logout' %}" style="display:
inline;">
                      {% csrf_token %}
                      <button type="submit" class="dropdown-item logout-button">
                        <i class="fas fa-sign-out-alt"></i> Logout
                      </button>
                    </form>
                  </a>
              </li>
            </ul>
          </li>
          {% else %}
          <li class="nav-item">
            <a class="nav-link" href="{% url 'login' %}">
              <i class="fas fa-sign-in-alt"></i> Login
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'signup' %}">
              <i class="fas fa-user-plus"></i> Sign Up
            </a>
          </li>
          {% endif %}
        </ul>
      </div>
```

```
     </div>
   </nav>

   <!-- Advanced Search Modal -->
   <div class="modal fade" id="advancedSearchModal" tabindex="-1"
aria-labelledby="advancedSearchModalLabel" aria-hidden="true">
     <div class="modal-dialog">
       <form method="get" action="{% url 'search_posts' %}" class="modal-content">
         <div class="modal-header">
           <h5 class="modal-title" id="advancedSearchModalLabel">Advanced Search</h5>
           <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
         </div>
         <div class="modal-body">
           <!-- Search Query -->
           <div class="mb-3">
             <label for="advanced-search-query" class="form-label">Search
Query</label>
             <input type="text" class="form-control" id="advanced-search-query"
name="q" placeholder="Enter search terms">
           </div>
           <!-- Attributes -->
           <h6>Filter by Attributes:</h6>
           {% include 'object_finder/attributes.html' %}
         </div>
         <div class="modal-footer">
           <button type="submit" class="btn btn-primary">Search</button>
           <button type="button" class="btn btn-secondary"
data-bs-dismiss="modal">Close</button>
         </div>
       </form>
     </div>
   </div>

   <!-- Hero Section -->
   <header class="hero-section">
     <div class="hero-overlay">
       <div class="container text-center text-white">
         <h1>My(stery) Object Finder</h1>
         <p class="lead">Find your unique object</p>

         {% if not user.is_authenticated %}
```

```
        <a href="{% url 'signup' %}" class="btn btn-primary btn-lg mt-3">Join Now</a>
        {% endif %}
      </div>
    </div>
  </header>

  <!-- Main Content Layout -->
  <div class="container-fluid">
    <div class="row">
      <!-- Left Sidebar - Categories -->
      <aside class="col-md-2 bg-light d-none d-md-block sidebar">
        <div class="p-3">
          <h5>Categories</h5>
          <ul class="nav flex-column">
            {% for category in categories %}
              <li class="nav-item">
                <a class="nav-link" href="{% url 'category_posts' category.slug
%}">{{ category.name }}</a>
              </li>
            {% endfor %}
          </ul>
        </div>
      </aside>

      <!-- Main Content Area -->
      <main class="col-md-8">
        {% block content %}
        {% endblock content %}
      </main>

      <!-- Right Sidebar - Latest Posts & Topics -->
      <aside class="col-md-2 bg-light d-none d-md-block sidebar">
        <div class="p-3">
          <h5>Latest Posts</h5>
          <ul class="list-unstyled">
            {% for post in latest_posts %}
            <li>
              <a href="{% url 'view_post' post_id=post.id %}">{{ post.title }}</a>
            </li>
            {% endfor %}
          </ul>
          <h5 class="mt-4">Popular Topics</h5>
          <ul class="list-unstyled">
            {% for topic in popular_topics %}
            <li>
              <a href="{% url 'topic_posts' topic.slug %}">{{ topic.name }}</a>
            </li>
            {% endfor %}
          </ul>
        </div>
      </aside>
    </div>
  </div>

  <!-- Footer -->
  <footer class="footer mt-auto py-3 bg-dark text-light">
    <div class="container text-center">
      <span class="text-muted">&copy; {{ now.year }} (My)stery Object Finder
Name</span>
    </div>
  </footer>

  <!-- External JavaScript Dependencies -->
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
integrity="sha512-2rNj2KJ+D8s1ceNasTIex6z4HWyOnEYLVC3FigGOmyQCZc2eBXKgOxQmo3oKLHyfcj53
uz4QMsRCWNbLd32Q1g==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy"
crossorigin="anonymous"></script>
  <link href="https://cdn.quilljs.com/1.3.7/quill.snow.css" rel="stylesheet">
  <link
href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css"
rel="stylesheet" />
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
</body>
</html>
```

/websiteproject/forum/apps/object_finder/templates/object_finder/comments.html

```
<!-- Main comment section container -->
<div id="comment-div">
```

```
<h3>Comments</h3>
<!-- Container for all comments with margin bottom -->
<div class="comments-section mb-4">
    {% if comments %}
        {% for comment in comments %}
        {% if comment.is_solved %}
            <!-- Solved comment with special background color -->
            <div style="background-color: #EAB676" class="comment mb-3 p-3 border
rounded">
        {% else %}
            <!-- Unsolved comment with HTMX swap behavior -->
            <div hx-swap="outerHTML" class="comment mb-3 p-3 border rounded">
        {% endif %}
                <!-- Comment header with author info and solved status -->
                <div class="d-flex justify-content-between align-items-center">
                    <p class="mb-0">
                        <strong>
                            {% if comment.author %}
                                <!-- Link to author's profile if authenticated -->
                                <a href="{% url 'profile' comment.author.id %}">
                                    {{ comment.author.username }}
                                </a>
                            {% else %}
                                <!-- Display anonymous name if no author -->
                                {{ comment.anonymous_name|default:"Anonymous" }}
                            {% endif %}
                        </strong>
                        <small class="text-muted"> on {{ comment.date_posted|date:"F
j, Y, g:i a" }}</small>
                    </p>
                    {% if user.is_authenticated and user == post.author %}
                    <!-- Solved checkbox for post author -->
                    <div class="form-check">
                        <input
                            hx-post="{% url 'update_is_solved' post.id comment.id %}"
                            hx-target="#comment-div"
                            hx-swap="outerHTML"
                            type="checkbox"
                            class="form-check-input"
                            id="isSolved-{{ comment.id }}"
                            {% if comment.is_solved %}checked{% endif %}>
                        <label class="form-check-label" for="isSolved-{{ comment.id
}}">Solved</label>
                    </div>
                    {% else %}
                      {% if comment.is_solved %}
                        <!-- Display solved status for non-authors -->
                        Solved by this comment!
                      {% endif %}
                    {% endif %}
                </div>
                <!-- Comment content -->
                <p>{{ comment.content }}</p>
            </div>
        {% endfor %}
    {% else %}
        <!-- Message when no comments exist -->
        <p>No comments yet. Be the first to comment!</p>
    {% endif %}
 </div>
</div>
```

/websiteproject/forum/apps/object_finder/templates/object_finder/form.html

```
<!-- Main form template for creating new posts -->
{% extends "object_finder/base.html" %}
{% load static %}

{% block content %}
<div class="container mt-5">
   <h2>Create a New Post</h2>
   <form id="post-form" method="post">
       {% csrf_token %}
       {% if form.errors %}
           <div class="alert alert-danger">
               <strong>Please correct the errors below:</strong>
               {{ form.errors }}
           </div>
       {% endif %}
       {{ form.title }}

       <!-- Tags input with autocomplete -->
```

```
        <div class="form-group">
            <label for="tags-input">Tags</label>
            <input type="text" id="tags-input" class="form-control" placeholder="Start
typing to search tags...">
        </div>

        <!-- Rich text editor -->
        <div id="quill-editor" style="height: 300px;"></div>
        <input type="hidden" name="content_delta" id="content-delta">
        <input type="hidden" name="tags" id="selected-tags">

        <!-- Physical attributes form section -->
        {% include 'object_finder/attributes.html' %}

        <button type="submit" id="submit-button" class="btn
btn-primary">Submit</button>
    </form>
</div>

<!-- External dependencies -->
<script src="{% static 'js/wikidata.js' %}"></script>

<script>
    document.addEventListener('DOMContentLoaded', function() {
        // Initialize Quill editor
        var quill = new Quill('#quill-editor', {
            theme: 'snow',
            modules: {
                toolbar: [
                    [{ 'header': [1, 2, false] }],
                    ['bold', 'italic', 'underline'],
                    ['image'],
                    ['clean']
                ]
            }
        });

        var form = document.getElementById('post-form');
        var submitButton = document.getElementById('submit-button');

        // Handle form submission
        submitButton.addEventListener('click', function(event) {
```

```
        event.preventDefault();
        var content_delta = JSON.stringify(quill.getContents());
        document.getElementById('content-delta').value = content_delta;
        form.submit();
    });

    // Tags autocomplete setup
    var tagsInput = document.getElementById('tags-input');
    var selectedTagsInput = document.getElementById('selected-tags');
    var selectedTags = [];

    function createSuggestionList() {
        var suggestionList = document.createElement('ul');
        suggestionList.id = 'suggestion-list';
        suggestionList.className = 'list-group';
        suggestionList.style.position = 'absolute';
        suggestionList.style.zIndex = '1000';
        suggestionList.style.width = '100%';
        suggestionList.style.maxHeight = '200px';
        suggestionList.style.overflowY = 'auto';
        tagsInput.parentNode.appendChild(suggestionList);
        return suggestionList;
    }

    var suggestionList = createSuggestionList();

    // Handle tag input and suggestions
    tagsInput.addEventListener('input', function() {
        var query = tagsInput.value.trim();
        if (query.length > 0) {
            WikidataAPI.searchEntities(query, 5).then(results => {
                suggestionList.innerHTML = '';
                results.forEach(result => {
                    var li = document.createElement('li');
                    li.className = 'list-group-item list-group-item-action';
                    li.textContent = `${result.label} (${result.id})`;
                    li.dataset.id = result.id;
                    li.dataset.label = result.label;
                    li.addEventListener('click', function() {
                        addTag(result);
                        tagsInput.value = '';
                        suggestionList.innerHTML = '';
```

```
                       });
                       suggestionList.appendChild(li);
                   });
               });
           } else {
               suggestionList.innerHTML = '';
           }
       });

       // Tag management functions
       function addTag(tag) {
           if (selectedTags.find(t => t.id === tag.id)) {
               return;
           }
           selectedTags.push(tag);
           updateSelectedTagsInput();
           renderSelectedTags();
       }

       function removeTag(tagId) {
           selectedTags = selectedTags.filter(t => t.id !== tagId);
           updateSelectedTagsInput();
           renderSelectedTags();
       }

       function updateSelectedTagsInput() {
           selectedTagsInput.value = JSON.stringify(selectedTags);
       }

       function renderSelectedTags() {
           var tagsContainer = document.getElementById('selected-tags-container');
           if (!tagsContainer) {
               tagsContainer = document.createElement('div');
               tagsContainer.id = 'selected-tags-container';
               tagsInput.parentNode.insertBefore(tagsContainer,
tagsInput.nextSibling);
           }
           tagsContainer.innerHTML = '';
           selectedTags.forEach(tag => {
               var tagBadge = document.createElement('span');
               tagBadge.className = 'badge badge-primary mr-1';
               tagBadge.style.border = '1px solid #000';
               tagBadge.style.color = 'gray';
               tagBadge.textContent = tag.label;
               var removeBtn = document.createElement('span');
               removeBtn.className = 'ml-1 text-danger';
               removeBtn.style.cursor = 'pointer';
               removeBtn.innerHTML = '&times;';
               removeBtn.addEventListener('click', function() {
                   removeTag(tag.id);
               });
               tagBadge.appendChild(removeBtn);
               tagsContainer.appendChild(tagBadge);
           });
       }

       // Close suggestion list when clicking outside
       document.addEventListener('click', function(event) {
           if (!tagsInput.contains(event.target)) {
               suggestionList.innerHTML = '';
           }
       });
   });
</script>
{% endblock content %}
```

/websiteproject/forum/apps/object_finder/templates/object_finder/index.html

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

{% block content %}

<div class="container mt-5">
 <!-- Recent posts section -->
 <h2><i class="fas fa-stream"></i> Recent Discussions</h2>
 <div class="row">
   {% for post in posts %}
     <!-- Post card -->
     <div class="col-md-4 mb-4">
       <div class="card h-100">
         <a href="{% url 'view_post' post_id=post.id %}">
           <img id="post-image-{{ post.id }}" src="" class="card-img-top" alt="Post
Image">
         </a>
         <div class="card-body">
           <h5 class="card-title">
             <a href="{% url 'view_post' post_id=post.id %}"
class="text-decoration-none">{{ post.title }}</a>
           </h5>
           <p id="post-text-{{ post.id }}" class="card-text"></p>
         </div>
         <div class="card-footer">
           <small class="text-muted">Posted by {{ post.author.username }} on {{
post.date_posted|date:"F j, Y" }}</small>
         </div>
       </div>
     </div>
     <!-- Create new row after every 3 posts -->
     {% if forloop.counter|divisibleby:3 %}
       </div><div class="row">
     {% endif %}
   {% endfor %}
 </div>

 <!-- Pagination -->
 <nav aria-label="Page navigation">
   <ul class="pagination justify-content-center">
     {% if posts.has_previous %}
     <li class="page-item">
       <a class="page-link" href="?page={{ posts.previous_page_number }}"
aria-label="Previous">
         <span aria-hidden="true">&laquo;</span>
       </a>
     </li>
     {% else %}
     <li class="page-item disabled">
       <span class="page-link" aria-label="Previous">
         <span aria-hidden="true">&laquo;</span>
       </span>
     </li>
     {% endif %}
     {% for num in posts.paginator.page_range %}
         {% if posts.number == num %}
         <li class="page-item active"><span class="page-link">{{ num }}</span></li>
         {% elif num > posts.number|add:'-3' and num < posts.number|add:'3' %}
         <li class="page-item"><a class="page-link" href="?page={{ num }}">{{ num
}}</a></li>
         {% endif %}
         {% endfor %}
         {% if posts.has_next %}
         <li class="page-item">
           <a class="page-link" href="?page={{ posts.next_page_number }}"
aria-label="Next">
             <span aria-hidden="true">&raquo;</span>
           </a>
         </li>
     {% else %}
     <li class="page-item disabled">
       <span class="page-link" aria-label="Next">
         <span aria-hidden="true">&raquo;</span>
       </span>
     </li>
     {% endif %}
   </ul>
 </nav>
</div>

<!-- Script to handle post content and images -->
<script>
   {% for post in posts %}
       (function() {
           // Parse the post content delta
           var contentDelta = {{ post.content_delta|safe }};
           var ops = contentDelta.ops || [];
           var imageUrl = null;
           var previewText = '';

           // Extract first image and preview text
           for (var i = 0; i < ops.length; i++) {
               var op = ops[i];
               if (op.insert) {
                   if (typeof op.insert === 'object' && op.insert.image && !imageUrl)
{
                       imageUrl = op.insert.image;
                   } else if (typeof op.insert === 'string') {
                       previewText += op.insert;
                   }
```

```
                }
                if (previewText.length >= 100) {
                    previewText = previewText.substring(0, 100) + '...';
                    break;
                }
            }

            // Update DOM with extracted content
            document.addEventListener('DOMContentLoaded', function() {
                document.getElementById('post-image-{{ post.id }}').src = imageUrl;
                document.getElementById('post-text-{{ post.id }}').textContent =
previewText.trim();
            });
        })();
    {% endfor %}
</script>

<!-- Styles for post images -->
<style>
    .card-img-top {
        height: 400px;
        object-fit: cover;
    }
</style>

{% endblock content %}
```

/websiteproject/forum/apps/object_finder/templates/object_finder/profile.html

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

{% block content %}
<!-- Main profile container -->
<div class="container mt-5">
 <h2>User Profile</h2>
 <!-- User info card -->
 <div class="card">
   <div class="card-body">
     <h5 class="card-title">{{ user.username }}
     <!-- Show top contributor badge if user has 5+ posts -->
     {% if user_posts|length >= 5 %}
        - <span class="badge top-contributor-badge">Top Contributor</span>
     {% endif %}
     </h5>
     <!-- User contact and personal info -->
     {% if user.email %}
       <p class="card-text"><strong>Email:</strong> {{ user.email }}</p>
     {% endif %}
     {% if user.first_name %}
       <p class="card-text"><strong>First Name:</strong> {{ user.first_name }}</p>
     {% endif %}
     {% if user.last_name %}
       <p class="card-text"><strong>Last Name:</strong> {{ user.last_name }}</p>
     {% endif %}
   </div>
 </div>
 <br>
 <!-- User's posts section -->
 <h2>User Posts</h2>
 <hr>
 <ul class="list-group">
   {% for post in user_posts %}
     <li class="list-group-item">
       <a href="{% url 'view_post' post_id=post.id %}">{{ post.title }}</a>
     </li>
   {% empty %}
     <li class="list-group-item">This user has not posted anything yet.</li>
   {% endfor %}
 </ul>
</div>

<!-- Styles for top contributor badge -->
<style>
 .top-contributor-badge {
   background-color: #FFD700; /* Gold color */
   color: #000;
 }
</style>
{% endblock content %}
```

/websiteproject/forum/apps/object_finder/templates/object_finder/view_post.html

```
{% extends "object_finder/base.html" %}
{% load static %}
{% block content %}
<div class="container mt-5">
   <!-- Post title section -->
   <div class="d-flex justify-content-between align-items-center mb-4">
       {% if is_solved %}
       <h2>{{ post.title }}<h3 style="color:red">This post has been solved!</h3></h2>
       {% else %}
       <h2>{{ post.title }}</h2>
       {% endif %}
   </div>

   <!-- Solved status alert -->
   {% if post.is_solved %}
       <div class="alert alert-success" role="alert">
           This issue has been resolved.
       </div>
   {% endif %}

   <!-- Post content viewer -->
   <div id="quill-viewer" class="mb-4"></div>

   <div class="row">
       <!-- Physical attributes section -->
       {% if post.attributes %}
           <div class="col-md-6">
               <h4>Attributes:</h4>
               <ul class="list-group mb-3">
                   {% for key, value in post.attributes.items %}
                       <li class="list-group-item">
                           <strong>{{ key|title }}:</strong> {{ value.value }} {{
value.unit }}
                       </li>
                   {% endfor %}
               </ul>
           </div>
       {% endif %}

       <!-- Tags section -->
       {% if tags %}
       <div class="col-md-6">
           <h4>Tags:</h4>
```

```
            <div class="container">
                <ul id="tags-list" class="list-inline">
                    {% for tag in tags %}
                        <li class="list-inline-item">
                            <a href="https://www.wikidata.org/wiki/{{ tag.tag_id }}"
target="_blank" class="tag badge badge-info" data-tag-id="{{ tag.tag_id }}"
style="text-decoration: none; color: inherit;">
                                {{ tag.name }}
                            </a>
                        </li>
                    {% endfor %}
                </ul>
            </div>
        </div>
        {% endif %}
    </div>

    <!-- Author info section -->
    <div class="mb-4">
        <h4>About the Author</h4>
        <p>
            <strong>
                <a href="{% url 'profile' post.author.id %}">
                    {{ post.author.username }}
                </a>
            </strong>
        </p>
    </div>

    <!-- Comments section -->
    {% include 'object_finder/comments.html' %}

    <!-- Comment form -->
    <div class="comment-form">
        <h4>Leave a Comment</h4>
        <form method="post">
            {% csrf_token %}
            <input type="hidden" name="form_type" value="comment_form">

            {% if user.is_authenticated %}
                <div class="form-check mb-2">
```

```
                    <input type="checkbox" class="form-check-input"
id="post_as_anonymous" name="post_as_anonymous">
                    <label class="form-check-label" for="post_as_anonymous">Post as
anonymous</label>
                </div>
            {% endif %}

            {{ comment_form.as_p }}
            <button type="submit" class="btn btn-primary">Submit Comment</button>
        </form>
    </div>
</div>

<!-- External dependencies -->
<script src="{% static 'js/wikidata.js' %}"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/blueimp-md5/2.19.0/js/md5.min.js"></script
>

<script>
    // Initialize Quill viewer
    var quill = new Quill('#quill-viewer', {
        theme: 'snow',
        readOnly: true,
        modules: {
            toolbar: false
        }
    });

    var content_delta = {{ post.content_delta|safe }};
    quill.setContents(content_delta);

    // Handle tag tooltips
    document.addEventListener('DOMContentLoaded', function() {
        var tags = document.querySelectorAll('.tag');
        var tagDataCache = {};

        // Add event listeners to tags
        tags.forEach(function(tagElement) {
            tagElement.addEventListener('mouseover', function() {
                var tagId = tagElement.dataset.tagId;
                if (tagDataCache[tagId]) {
                    showTooltip(tagElement, tagDataCache[tagId]);
                } else {
                    fetchTagDetails(tagElement, tagId);
                }
            });

            tagElement.addEventListener('mouseout', function() {
                hideTooltip();
            });
        });

        // Fetch tag details from Wikidata API
        function fetchTagDetails(element, tagId) {
            WikidataAPI.getEntityDetails(tagId)
                .then(function(entity) {
                    var description = entity.descriptions && entity.descriptions.en ?
entity.descriptions.en.value : 'No description available.';
                    var imageUrl = '';

                    if (entity.claims && entity.claims.P18 && entity.claims.P18[0]) {
                        var fileName = entity.claims.P18[0].mainsnak.datavalue.value;
                        imageUrl = getCommonsImageUrl(fileName);
                    }

                    var data = { description: description, imageUrl: imageUrl };
                    tagDataCache[tagId] = data;
                    showTooltip(element, data);
                })
                .catch(function(error) {
                    console.error('Error fetching tag details:', error);
                });
        }

        // Generate Wikimedia Commons image URL
        function getCommonsImageUrl(fileName) {
            var sanitizedFileName = fileName.replace(/ /g, '_');
            var encodedFileName = encodeURIComponent(sanitizedFileName);
            var md5Hash = md5(sanitizedFileName);
            var url = 'https://upload.wikimedia.org/wikipedia/commons/' +
                md5Hash.substring(0,1) + '/' + md5Hash.substring(0,2) + '/' +
encodedFileName;
            return url;
```

```
    }

    // Display tooltip with tag information
    function showTooltip(element, data) {
        var tooltip = document.getElementById('tag-tooltip');
        if (!tooltip) {
            tooltip = document.createElement('div');
            tooltip.id = 'tag-tooltip';
            tooltip.style.position = 'absolute';
            tooltip.style.zIndex = '1000';
            tooltip.style.backgroundColor = '#fff';
            tooltip.style.border = '1px solid #ccc';
            tooltip.style.padding = '10px';
            tooltip.style.borderRadius = '5px';
            tooltip.style.boxShadow = '0px 0px 10px rgba(0,0,0,0.1)';
            tooltip.style.maxWidth = '200px';
            document.body.appendChild(tooltip);
        }
        tooltip.innerHTML = '';
        if (data.imageUrl) {
            var img = document.createElement('img');
            img.src = data.imageUrl;
            img.alt = 'Image';
            img.style.maxWidth = '100%';
            img.style.display = 'block';
            img.style.marginBottom = '10px';
            tooltip.appendChild(img);
        }
        var desc = document.createElement('p');
        desc.textContent = data.description;
        tooltip.appendChild(desc);

        var rect = element.getBoundingClientRect();
        tooltip.style.top = (window.scrollY + rect.bottom + 5) + 'px';
        tooltip.style.left = (window.scrollX + rect.left) + 'px';
        tooltip.style.display = 'block';
    }

    // Hide tooltip
    function hideTooltip() {
        var tooltip = document.getElementById('tag-tooltip');
        if (tooltip) {
               tooltip.style.display = 'none';
           }
       }
   });
</script>
{% endblock content %}
```

/websiteproject/forum/apps/object_finder/templates/object_finder/static/css/styles.css

```
/* styles.css */

/* Adjust the sidebar height to match the viewport */
.sidebar {
 position: sticky;
 top: 70px; /* Adjust based on navbar height */
 height: calc(100vh - 70px);
 overflow-y: auto;
}

/* Style the logout button */
.logout-button {
 background: none;
 border: none;
 padding: 0;
 margin: 0;
 width: 100%;
 text-align: left;
 color: #212529; /* Bootstrap's default text color */
}

.logout-button:hover {
 background-color: #f8f9fa; /* Bootstrap's light color */
}

.footer {
 position: relative;
 bottom: 0;
 width: 100%;
}

body {
 background-color: #f5f5f5; /* Light grey background */
```

```css
}

.card {
 background-color: #ffffff;
}

.navbar-brand {
 font-weight: bold;
}


/* Hero Section Styling */
.hero-section {
 background: url("../images/forum-banner.jpg") no-repeat center center;
 background-size: cover;
 position: relative;
 height: 300px; /* Adjust height as needed */
}

.hero-overlay {
 background-color: rgba(0, 0, 0, 0.5); /* Semi-transparent overlay */
 position: absolute;
 top: 0;
 left: 0;
 width: 100%;
 height: 100%;
}

.hero-section .container {
 position: relative;
 top: 50%;
 transform: translateY(-50%);
}

.hero-section h1 {
 font-size: 3rem;
 font-weight: bold;
}

.hero-section p.lead {
 font-size: 1.5rem;
}
```

/websiteproject/forum/apps/object_finder/templates/object_finder/static/js/wikidata.js

```javascript
// wikidata.js

const WikidataAPI = (() => {
    const apiUrl = 'https://www.wikidata.org/w/api.php';
    const sparqlEndpoint = 'https://query.wikidata.org/sparql';
    let defaultLimit = 10;

    function setDefaultLimit(limit) {
        defaultLimit = limit;
    }

    async function searchEntities(searchTerm, limit = defaultLimit, language = 'en') {
        if (!searchTerm) {
            return Promise.resolve([]);
        }

        const params = {
            action: 'wbsearchentities',
            format: 'json',
            language: language,
            uselang: language,
            search: searchTerm,
            limit: limit,
            origin: '*',
        };

        const url = `${apiUrl}?${new URLSearchParams(params).toString()}`;

        try {
            const response = await fetch(url);
            const data = await response.json();
            return data.search || [];
        } catch (error) {
            console.error('Wikidata API Error:', error);
            return [];
        }
    }
```

```
async function getEntityDetails(entityId, languages = ['en']) {
    if (!entityId) {
        return Promise.reject(new Error('Entity ID is required.'));
    }

    const params = {
        action: 'wbgetentities',
        format: 'json',
        ids: entityId,
        languages: languages.join('|'),
        origin: '*',
    };

    const url = `${apiUrl}?${new URLSearchParams(params).toString()}`;

    try {
        const response = await fetch(url);
        const data = await response.json();
        return data.entities[entityId] || {};
    } catch (error) {
        console.error('Wikidata API Error:', error);
        return {};
    }
}

async function getRelatedEntities(entityId, relationshipType = 'P279') {
    if (!entityId) {
        return Promise.reject(new Error('Entity ID is required.'));
    }

    const query = `
        SELECT ?item ?itemLabel WHERE {
            wd:${entityId} wdt:${relationshipType} ?item .
            SERVICE wikibase:label { bd:serviceParam wikibase:language
"[AUTO_LANGUAGE],en". }
        }
    `;

    const url = `${sparqlEndpoint}?${new URLSearchParams({
        query: query,
        format: 'json',
    }).toString()}`;

    try {
        const response = await fetch(url);
        const data = await response.json();
        const results = data.results.bindings;
        return results.map(result_2 => ({
            id: result_2.item.value.split('/').pop(),
            label: result_2.itemLabel.value,
        }));
    } catch (error) {
        console.error('Wikidata SPARQL Error:', error);
        return [];
    }
}

async function listLanguages(entityId) {
    if (!entityId) {
        return Promise.reject(new Error('Entity ID is required.'));
    }

    const params = {
        action: 'wbgetentities',
        format: 'json',
        ids: entityId,
        props: 'labels',
        origin: '*',
    };

    const url = `${apiUrl}?${new URLSearchParams(params).toString()}`;

    return fetch(url)
        .then(response => response.json())
        .then(data => {
            const entity = data.entities[entityId];
            if (entity && entity.labels) {
                return Object.keys(entity.labels);
            } else {
                return [];
            }
        })
        .catch(error => {
                console.error('Wikidata API Error:', error);
                return [];
            });
    }

    return {
        searchEntities,
        getEntityDetails,
        getRelatedEntities,
        listLanguages,
        setDefaultLimit,
    };
})();

// Usage examples (uncomment to test):

// WikidataAPI.setDefaultLimit(5);
// WikidataAPI.searchEntities('Python').then(results => console.log(results));
// WikidataAPI.getEntityDetails('Q42').then(details => console.log(details));
// WikidataAPI.getRelatedEntities('Q42', 'P31').then(related => console.log(related));
// WikidataAPI.listLanguages('Q42').then(languages => console.log(languages));
```

/websiteproject/forum/forum/settings_dev.py

```
"""
Django settings for forum project.

Generated by 'django-admin startproject' using Django 5.1.3.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
```

```python
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-9ggce_dmz69(+-l0@nw7fcx0o&e1ch9(8g1&8iy5&qe7q+546y'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ["*"]

# Authentication redirect settings
LOGIN_REDIRECT_URL = "/"
LOGOUT_REDIRECT_URL = "/"

# Installed Django apps
INSTALLED_APPS = [
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
   'apps.object_finder',
]

# Middleware configuration
MIDDLEWARE = [
   'django.middleware.security.SecurityMiddleware',
   'django.contrib.sessions.middleware.SessionMiddleware',
   'django.middleware.common.CommonMiddleware',
   'django.middleware.csrf.CsrfViewMiddleware',
   'django.contrib.auth.middleware.AuthenticationMiddleware',
   'django.contrib.messages.middleware.MessageMiddleware',
   'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'forum.urls'
```

```
# Template configuration
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'forum.wsgi.application'

# PostgreSQL database configuration
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/5.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files configuration
STATIC_URL = '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

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# Maximum upload size (50MB)
DATA_UPLOAD_MAX_MEMORY_SIZE = 52428800
```

/websiteproject/forum/forum/settings.py

```
"""
Django settings for forum project.

Generated by 'django-admin startproject' using Django 5.1.3.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
```

```python
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-9ggce_dmz69(+-l0@nw7fcx0o&e1ch9(8g1&8iy5&qe7q+546y'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ["*"]

# Authentication redirect settings
LOGIN_REDIRECT_URL = "/"
LOGOUT_REDIRECT_URL = "/"

# Installed Django apps
INSTALLED_APPS = [
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
   'apps.object_finder',
]

# Middleware configuration
MIDDLEWARE = [
   'django.middleware.security.SecurityMiddleware',
   'django.contrib.sessions.middleware.SessionMiddleware',
   'django.middleware.common.CommonMiddleware',
   'django.middleware.csrf.CsrfViewMiddleware',
   'django.contrib.auth.middleware.AuthenticationMiddleware',
   'django.contrib.messages.middleware.MessageMiddleware',
   'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'forum.urls'

# Template configuration
TEMPLATES = [
   {
       'BACKEND': 'django.template.backends.django.DjangoTemplates',
       'DIRS': [],
       'APP_DIRS': True,
       'OPTIONS': {
           'context_processors': [
               'django.template.context_processors.debug',
               'django.template.context_processors.request',
               'django.contrib.auth.context_processors.auth',
               'django.contrib.messages.context_processors.messages',
           ],
       },
   },
]

WSGI_APPLICATION = 'forum.wsgi.application'

# PostgreSQL database configuration
DATABASES = {
   'default': {
       'ENGINE': 'django.db.backends.postgresql',
       'NAME': os.environ.get('DATABASE_NAME', 'postgres'),
       'USER': os.environ.get('DATABASE_USER', 'postgres'),
       'PASSWORD': os.environ.get('DATABASE_PASSWORD', 'postgres'),
       'HOST': os.environ.get('DATABASE_HOST', 'localhost'),
       'PORT': os.environ.get('DATABASE_PORT', '5432'),
   }
}

# Password validation
# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
   {
       'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
   },
   {
       'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
   },
   {
       'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
   },
   {
       'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
   },
]


# Internationalization
# https://docs.djangoproject.com/en/5.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files configuration
STATIC_URL = '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

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# Maximum upload size (50MB)
DATA_UPLOAD_MAX_MEMORY_SIZE = 52428800
```

/websiteproject/forum/forum/urls.py

```
"""
URL configuration for forum project.

The `urlpatterns` list routes URLs to views. For more information please see:
   https://docs.djangoproject.com/en/5.1/topics/http/urls/
Examples:
Function views
   1. Add an import:  from my_app import views
   2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
   1. Add an import:  from other_app.views import Home
   2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
   1. Import the include() function: from django.urls import include, path
   2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""

# Import required Django modules
from django.contrib import admin
from django.urls import path, include

# Define URL patterns for the project
urlpatterns = [
   path('', include('apps.object_finder.urls')),  # Root URL points to object_finder
app
   path('admin/', admin.site.urls),  # Django admin interface
   path('accounts/', include('django.contrib.auth.urls')),  # Authentication URLs
]
```

/websiteproject/forum/forum/wsgi.py

```

import os

from django.core.wsgi import get_wsgi_application

# Set Django settings module path
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'forum.settings')

# Initialize WSGI application
application = get_wsgi_application()
```

/websiteproject/forum/manage.py

```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
   """
   The main function:
   1. Sets up Django settings module
   2. Imports and executes Django management commands
   3. Handles import errors with helpful messages
   """
   os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'forum.settings')
   try:
       from django.core.management import execute_from_command_line
   except ImportError as exc:
       raise ImportError(
           "Couldn't import Django. Are you sure it's installed and "
           "available on your PYTHONPATH environment variable? Did you "
           "forget to activate a virtual environment?"
       ) from exc
   execute_from_command_line(sys.argv)


if __name__ == '__main__':
   # Entry point - calls main() when script is run directly
   main()
```

/websiteproject/gitignore

```
*.sqlite3
venv/
.DS_Store/
__pycache__/
*/staticfiles/
```

/websiteproject/docker-compose.yml

```yaml
version: '3.8'
```

```yaml
services:
 web:
   build: .
   command: python forum/manage.py runserver 0.0.0.0:8000
   volumes:
     - .:/app
   ports:
     - "8000:8000"
   depends_on:
     - db
   environment:
     - DEBUG=1
     - DATABASE_NAME=postgres
     - DATABASE_USER=postgres
     - DATABASE_PASSWORD=postgres
     - DATABASE_HOST=db
     - DATABASE_PORT=5432

 db:
   image: postgres:13
   restart: always
   volumes:
     - postgres_data:/var/lib/postgresql/data/
   environment:
     - POSTGRES_DB=postgres
     - POSTGRES_USER=postgres
     - POSTGRES_PASSWORD=postgres

volumes:
 postgres_data:
```

Dockerfile

```dockerfile
# Use the official Python image as the base
FROM python:3.10-slim

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE 1   # Prevents Python from writing .pyc files to disk
ENV PYTHONUNBUFFERED 1          # Prevents Python from buffering stdout and stderr

# Set work directory
WORKDIR /app
```

```
# Install system dependencies
RUN apt-get update && apt-get install -y \
   build-essential \
   libpq-dev \
   libssl-dev \
   libffi-dev \
   python3-dev \
   && rm -rf /var/lib/apt/lists/*

# Copy requirements.txt to the working directory
COPY requirements.txt /app/

# Install Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code to the working directory
COPY . /app/

# Expose port 8000 (the default port for Django's development server)
EXPOSE 8000

# Run the Django development server
CMD ["python", "forum/manage.py", "runserver", "0.0.0.0:8000"]
```

README.md

```
# Initialize the venv

python3 -m venv venv


# Activate venv

source venv/bin/activate


# Install requirements

pip3 install -r requirements.txt
```

```
# Migrate database

python3 forum/manage.py migrate

# Create admin user

python3 forum/manage.py createsuperuser

# Collect static files

python3 forum/manage.py collectstatic

# Run server

python3 forum/manage.py runserver

# Login trough admin panel

http://127.0.0.1:8000/admin/

## When Docker is the prefered way to deploy

# Docker compose up and build
docker-compose up -d --build

# Docker run migrations
docker-compose exec web python forum/manage.py migrate

# Docker compose down
docker-compose down

# Docker create superuser
docker-compose exec web python forum/manage.py createsuperuser

## When running tests, make sure you're in where manage.py is located
python manage.py test --settings forum.settings_dev
```

Requirements.txt

```
django==5.1.3
```

```
Pillow==11.0.0
psycopg2==2.9.1
```