Django\_template

Based on your requirements, I will provide you with Django template fragments for the 'Product Search' use case.  
  
\*\*product\_search.html\*\*  
```html  
<!-- product\_search.html -->  
{% extends "base.html" %}  
  
{% block content %}  
 <h1>Product Search</h1>  
 <form method="get">  
 <label for="name">Name:</label>  
 <input type="text" id="name" name="name"><br><br>  
 <label for="description">Description:</label>  
 <textarea id="description" name="description"></textarea><br><br>  
 <button type="submit">Search</button>  
 </form>  
  
 {% if products %}  
 <ul>  
 {% for product in products %}  
 <li>  
 {{ product.name }} - {{ product.description }}  
 </li>  
 {% endfor %}  
 </ul>  
 {% else %}  
 <p>No results found.</p>  
 {% endif %}  
{% endblock %}  
```  
  
\*\*style.css\*\*  
```css  
/\* style.css \*/  
.product-search {  
 max-width: 800px;  
 margin: 40px auto;  
 padding: 20px;  
 border: 1px solid #ccc;  
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
.product-search form {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
}  
  
.product-search input[type="text"],.product-search textarea {  
 width: 100%;  
 height: 30px;  
 padding: 10px;  
 font-size: 16px;  
 border: 1px solid #ccc;  
}  
  
.product-search button[type="submit"] {  
 background-color: #4CAF50;  
 color: #fff;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
}  
  
.product-search ul {  
 list-style: none;  
 padding: 0;  
 margin: 0;  
}  
  
.product-search li {  
 padding: 10px;  
 border-bottom: 1px solid #ccc;  
}  
  
.product-search li:last-child {  
 border-bottom: none;  
}  
```  
  
Please note that this is just one possible implementation of the `product\_search.html` file, and you may need to adjust it to fit your specific needs. Additionally, you will need to create a corresponding view function in your Django application to handle the GET requests sent by the form and return the searched products. You can do this by creating a new view class in your app's `views.py` file:  
  
```python  
# views.py  
from django.shortcuts import render  
from.models import Product  
  
def product\_search(request):  
 name = request.GET.get('name')  
 description = request.GET.get('description')  
  
 if name and description:  
 products = Product.objects.filter(name\_\_icontains=name, description\_\_icontains=description)  
 elif name:  
 products = Product.objects.filter(name\_\_icontains=name)  
 elif description:  
 products = Product.objects.filter(description\_\_icontains=description)  
 else:  
 products = Product.objects.all()  
  
 return render(request, 'product\_search.html', {'products': products})  
```  
  
This view function retrieves the `name` and `description` parameters from the URL query string, filters the products accordingly, and passes them as an argument to the `render` shortcut function along with the `product\_search.html` template. The rendered page will then display the filtered products or an empty message if no matches were found.

Based on your requirements, I will provide you with Django template fragments for the 'Product Search' use case.  
  
\*\*product\_search.html\*\*  
```html  
<!-- product\_search.html -->  
{% extends "base.html" %}  
  
{% block content %}  
 <h1>Product Search</h1>  
 <form method="get">  
 <label for="name">Name:</label>  
 <input type="text" id="name" name="name"><br><br>  
 <label for="description">Description:</label>  
 <textarea id="description" name="description"></textarea><br><br>  
 <button type="submit">Search</button>  
 </form>  
  
 {% if products %}  
 <ul>  
 {% for product in products %}  
 <li>  
 {{ product.name }} - {{ product.description }}  
 </li>  
 {% endfor %}  
 </ul>  
 {% else %}  
 <p>No results found.</p>  
 {% endif %}  
{% endblock %}  
```  
  
\*\*style.css\*\*  
```css  
/\* style.css \*/  
.product-search {  
 max-width: 800px;  
 margin: 40px auto;  
 padding: 20px;  
 border: 1px solid #ccc;  
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
.product-search form {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
}  
  
.product-search input[type="text"],.product-search textarea {  
 width: 100%;  
 height: 30px;  
 padding: 10px;  
 font-size: 16px;  
 border: 1px solid #ccc;  
}  
  
.product-search button[type="submit"] {  
 background-color: #4CAF50;  
 color: #fff;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
}  
  
.product-search ul {  
 list-style: none;  
 padding: 0;  
 margin: 0;  
}  
  
.product-search li {  
 padding: 10px;  
 border-bottom: 1px solid #ccc;  
}  
  
.product-search li:last-child {  
 border-bottom: none;  
}  
```  
  
Please note that this is just one possible implementation of the `product\_search.html` file, and you may need to adjust it to fit your specific needs. Additionally, you will need to create a corresponding view function in your Django application to handle the GET requests sent by the form and return the searched products. You can do this by creating a new view class in your app's `views.py` file:  
  
```python  
# views.py  
from django.shortcuts import render  
from.models import Product  
  
def product\_search(request):  
 name = request.GET.get('name')  
 description = request.GET.get('description')  
  
 if name and description:  
 products = Product.objects.filter(name\_\_icontains=name, description\_\_icontains=description)  
 elif name:  
 products = Product.objects.filter(name\_\_icontains=name)  
 elif description:  
 products = Product.objects.filter(description\_\_icontains=description)  
 else:  
 products = Product.objects.all()  
  
 return render(request, 'product\_search.html', {'products': products})  
```  
  
This view function retrieves the `name` and `description` parameters from the URL query string, filters the products accordingly, and passes them as an argument to the `render` shortcut function along with the `product\_search.html` template. The rendered page will then display the filtered products or an empty message if no matches were found.

Based on your requirements, I will provide you with Django template fragments for the 'Product Search' use case.  
  
\*\*product\_search.html\*\*  
```html  
<!-- product\_search.html -->  
{% extends "base.html" %}  
  
{% block content %}  
 <h1>Product Search</h1>  
 <form method="get">  
 <label for="name">Name:</label>  
 <input type="text" id="name" name="name"><br><br>  
 <label for="description">Description:</label>  
 <textarea id="description" name="description"></textarea><br><br>  
 <button type="submit">Search</button>  
 </form>  
  
 {% if products %}  
 <ul>  
 {% for product in products %}  
 <li>  
 {{ product.name }} - {{ product.description }}  
 </li>  
 {% endfor %}  
 </ul>  
 {% else %}  
 <p>No results found.</p>  
 {% endif %}  
{% endblock %}  
```  
  
\*\*style.css\*\*  
```css  
/\* style.css \*/  
.product-search {  
 max-width: 800px;  
 margin: 40px auto;  
 padding: 20px;  
 border: 1px solid #ccc;  
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
.product-search form {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
}  
  
.product-search input[type="text"],.product-search textarea {  
 width: 100%;  
 height: 30px;  
 padding: 10px;  
 font-size: 16px;  
 border: 1px solid #ccc;  
}  
  
.product-search button[type="submit"] {  
 background-color: #4CAF50;  
 color: #fff;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
}  
  
.product-search ul {  
 list-style: none;  
 padding: 0;  
 margin: 0;  
}  
  
.product-search li {  
 padding: 10px;  
 border-bottom: 1px solid #ccc;  
}  
  
.product-search li:last-child {  
 border-bottom: none;  
}  
```  
  
Please note that this is just one possible implementation of the `product\_search.html` file, and you may need to adjust it to fit your specific needs. Additionally, you will need to create a corresponding view function in your Django application to handle the GET requests sent by the form and return the searched products. You can do this by creating a new view class in your app's `views.py` file:  
  
```python  
# views.py  
from django.shortcuts import render  
from.models import Product  
  
def product\_search(request):  
 name = request.GET.get('name')  
 description = request.GET.get('description')  
  
 if name and description:  
 products = Product.objects.filter(name\_\_icontains=name, description\_\_icontains=description)  
 elif name:  
 products = Product.objects.filter(name\_\_icontains=name)  
 elif description:  
 products = Product.objects.filter(description\_\_icontains=description)  
 else:  
 products = Product.objects.all()  
  
 return render(request, 'product\_search.html', {'products': products})  
```  
  
This view function retrieves the `name` and `description` parameters from the URL query string, filters the products accordingly, and passes them as an argument to the `render` shortcut function along with the `product\_search.html` template. The rendered page will then display the filtered products or an empty message if no matches were found.

Based on your requirements, I will provide you with Django template fragments for the 'Product Search' use case.  
  
\*\*product\_search.html\*\*  
```html  
<!-- product\_search.html -->  
{% extends "base.html" %}  
  
{% block content %}  
 <h1>Product Search</h1>  
 <form method="get">  
 <label for="name">Name:</label>  
 <input type="text" id="name" name="name"><br><br>  
 <label for="description">Description:</label>  
 <textarea id="description" name="description"></textarea><br><br>  
 <button type="submit">Search</button>  
 </form>  
  
 {% if products %}  
 <ul>  
 {% for product in products %}  
 <li>  
 {{ product.name }} - {{ product.description }}  
 </li>  
 {% endfor %}  
 </ul>  
 {% else %}  
 <p>No results found.</p>  
 {% endif %}  
{% endblock %}  
```  
  
\*\*style.css\*\*  
```css  
/\* style.css \*/  
.product-search {  
 max-width: 800px;  
 margin: 40px auto;  
 padding: 20px;  
 border: 1px solid #ccc;  
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
.product-search form {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
}  
  
.product-search input[type="text"],.product-search textarea {  
 width: 100%;  
 height: 30px;  
 padding: 10px;  
 font-size: 16px;  
 border: 1px solid #ccc;  
}  
  
.product-search button[type="submit"] {  
 background-color: #4CAF50;  
 color: #fff;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
}  
  
.product-search ul {  
 list-style: none;  
 padding: 0;  
 margin: 0;  
}  
  
.product-search li {  
 padding: 10px;  
 border-bottom: 1px solid #ccc;  
}  
  
.product-search li:last-child {  
 border-bottom: none;  
}  
```  
  
Please note that this is just one possible implementation of the `product\_search.html` file, and you may need to adjust it to fit your specific needs. Additionally, you will need to create a corresponding view function in your Django application to handle the GET requests sent by the form and return the searched products. You can do this by creating a new view class in your app's `views.py` file:  
  
```python  
# views.py  
from django.shortcuts import render  
from.models import Product  
  
def product\_search(request):  
 name = request.GET.get('name')  
 description = request.GET.get('description')  
  
 if name and description:  
 products = Product.objects.filter(name\_\_icontains=name, description\_\_icontains=description)  
 elif name:  
 products = Product.objects.filter(name\_\_icontains=name)  
 elif description:  
 products = Product.objects.filter(description\_\_icontains=description)  
 else:  
 products = Product.objects.all()  
  
 return render(request, 'product\_search.html', {'products': products})  
```  
  
This view function retrieves the `name` and `description` parameters from the URL query string, filters the products accordingly, and passes them as an argument to the `render` shortcut function along with the `product\_search.html` template. The rendered page will then display the filtered products or an empty message if no matches were found.

Based on your requirements, I will provide you with Django template fragments for the 'Product Search' use case.  
  
\*\*product\_search.html\*\*  
```html  
<!-- product\_search.html -->  
{% extends "base.html" %}  
  
{% block content %}  
 <h1>Product Search</h1>  
 <form method="get">  
 <label for="name">Name:</label>  
 <input type="text" id="name" name="name"><br><br>  
 <label for="description">Description:</label>  
 <textarea id="description" name="description"></textarea><br><br>  
 <button type="submit">Search</button>  
 </form>  
  
 {% if products %}  
 <ul>  
 {% for product in products %}  
 <li>  
 {{ product.name }} - {{ product.description }}  
 </li>  
 {% endfor %}  
 </ul>  
 {% else %}  
 <p>No results found.</p>  
 {% endif %}  
{% endblock %}  
```  
  
\*\*style.css\*\*  
```css  
/\* style.css \*/  
.product-search {  
 max-width: 800px;  
 margin: 40px auto;  
 padding: 20px;  
 border: 1px solid #ccc;  
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
.product-search form {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
}  
  
.product-search input[type="text"],.product-search textarea {  
 width: 100%;  
 height: 30px;  
 padding: 10px;  
 font-size: 16px;  
 border: 1px solid #ccc;  
}  
  
.product-search button[type="submit"] {  
 background-color: #4CAF50;  
 color: #fff;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
}  
  
.product-search ul {  
 list-style: none;  
 padding: 0;  
 margin: 0;  
}  
  
.product-search li {  
 padding: 10px;  
 border-bottom: 1px solid #ccc;  
}  
  
.product-search li:last-child {  
 border-bottom: none;  
}  
```  
  
Please note that this is just one possible implementation of the `product\_search.html` file, and you may need to adjust it to fit your specific needs. Additionally, you will need to create a corresponding view function in your Django application to handle the GET requests sent by the form and return the searched products. You can do this by creating a new view class in your app's `views.py` file:  
  
```python  
# views.py  
from django.shortcuts import render  
from.models import Product  
  
def product\_search(request):  
 name = request.GET.get('name')  
 description = request.GET.get('description')  
  
 if name and description:  
 products = Product.objects.filter(name\_\_icontains=name, description\_\_icontains=description)  
 elif name:  
 products = Product.objects.filter(name\_\_icontains=name)  
 elif description:  
 products = Product.objects.filter(description\_\_icontains=description)  
 else:  
 products = Product.objects.all()  
  
 return render(request, 'product\_search.html', {'products': products})  
```  
  
This view function retrieves the `name` and `description` parameters from the URL query string, filters the products accordingly, and passes them as an argument to the `render` shortcut function along with the `product\_search.html` template. The rendered page will then display the filtered products or an empty message if no matches were found.