Django\_template

Here is the Django template fragment for the 'Product Viewing' use case:  
```html  
<!-- product\_details.html -->  
  
<h1>{{ product.name }}</h1>  
<p> {{ product.description }} </p>  
  
<div class="product-image">  
 <img src="{{ product.image.url }}" alt="Product Image">  
</div>  
  
<ul>  
 {% for attribute, value in product.attributes.items %}  
 <li>  
 {{ attribute }}: {{ value }}  
 </li>  
 {% endfor %}  
</ul>  
  
<button class="add-to-cart">Add to Cart</button>  
  
<script type="text/javascript" src="{% static 'js/script.js' %}"></script>  
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">  
```  
This template displays the product name, description, image, attributes, and provides an "Add to Cart" button. The template uses Django's built-in template syntax, including variable interpolation (`{{ }}`) and loops (`{% for %}`).  
  
Note that this template assumes that the `product` object has been passed as a context variable from your view. You will need to modify your view to pass the correct data to the template.  
  
Also, the CSS file (`style.css`) and JavaScript file (`script.js`) mentioned in the template are assumed to be located in the same directory as the template, and have been loaded correctly. You may need to adjust these paths depending on your project structure.

Here is the Django template fragment for the 'Product Viewing' use case:  
```html  
<!-- product\_details.html -->  
  
<h1>{{ product.name }}</h1>  
<p> {{ product.description }} </p>  
  
<div class="product-image">  
 <img src="{{ product.image.url }}" alt="Product Image">  
</div>  
  
<ul>  
 {% for attribute, value in product.attributes.items %}  
 <li>  
 {{ attribute }}: {{ value }}  
 </li>  
 {% endfor %}  
</ul>  
  
<button class="add-to-cart">Add to Cart</button>  
  
<script type="text/javascript" src="{% static 'js/script.js' %}"></script>  
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">  
```  
This template displays the product name, description, image, attributes, and provides an "Add to Cart" button. The template uses Django's built-in template syntax, including variable interpolation (`{{ }}`) and loops (`{% for %}`).  
  
Note that this template assumes that the `product` object has been passed as a context variable from your view. You will need to modify your view to pass the correct data to the template.  
  
Also, the CSS file (`style.css`) and JavaScript file (`script.js`) mentioned in the template are assumed to be located in the same directory as the template, and have been loaded correctly. You may need to adjust these paths depending on your project structure.

Here is the Django template fragment for the 'Product Viewing' use case:  
```html  
<!-- product\_details.html -->  
  
<h1>{{ product.name }}</h1>  
<p> {{ product.description }} </p>  
  
<div class="product-image">  
 <img src="{{ product.image.url }}" alt="Product Image">  
</div>  
  
<ul>  
 {% for attribute, value in product.attributes.items %}  
 <li>  
 {{ attribute }}: {{ value }}  
 </li>  
 {% endfor %}  
</ul>  
  
<button class="add-to-cart">Add to Cart</button>  
  
<script type="text/javascript" src="{% static 'js/script.js' %}"></script>  
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">  
```  
This template displays the product name, description, image, attributes, and provides an "Add to Cart" button. The template uses Django's built-in template syntax, including variable interpolation (`{{ }}`) and loops (`{% for %}`).  
  
Note that this template assumes that the `product` object has been passed as a context variable from your view. You will need to modify your view to pass the correct data to the template.  
  
Also, the CSS file (`style.css`) and JavaScript file (`script.js`) mentioned in the template are assumed to be located in the same directory as the template, and have been loaded correctly. You may need to adjust these paths depending on your project structure.

Here is the Django template fragment for the 'Product Viewing' use case:  
```html  
<!-- product\_details.html -->  
  
<h1>{{ product.name }}</h1>  
<p> {{ product.description }} </p>  
  
<div class="product-image">  
 <img src="{{ product.image.url }}" alt="Product Image">  
</div>  
  
<ul>  
 {% for attribute, value in product.attributes.items %}  
 <li>  
 {{ attribute }}: {{ value }}  
 </li>  
 {% endfor %}  
</ul>  
  
<button class="add-to-cart">Add to Cart</button>  
  
<script type="text/javascript" src="{% static 'js/script.js' %}"></script>  
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">  
```  
This template displays the product name, description, image, attributes, and provides an "Add to Cart" button. The template uses Django's built-in template syntax, including variable interpolation (`{{ }}`) and loops (`{% for %}`).  
  
Note that this template assumes that the `product` object has been passed as a context variable from your view. You will need to modify your view to pass the correct data to the template.  
  
Also, the CSS file (`style.css`) and JavaScript file (`script.js`) mentioned in the template are assumed to be located in the same directory as the template, and have been loaded correctly. You may need to adjust these paths depending on your project structure.

Here is the Django template fragment for the 'Product Viewing' use case:  
```html  
<!-- product\_details.html -->  
  
<h1>{{ product.name }}</h1>  
<p> {{ product.description }} </p>  
  
<div class="product-image">  
 <img src="{{ product.image.url }}" alt="Product Image">  
</div>  
  
<ul>  
 {% for attribute, value in product.attributes.items %}  
 <li>  
 {{ attribute }}: {{ value }}  
 </li>  
 {% endfor %}  
</ul>  
  
<button class="add-to-cart">Add to Cart</button>  
  
<script type="text/javascript" src="{% static 'js/script.js' %}"></script>  
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">  
```  
This template displays the product name, description, image, attributes, and provides an "Add to Cart" button. The template uses Django's built-in template syntax, including variable interpolation (`{{ }}`) and loops (`{% for %}`).  
  
Note that this template assumes that the `product` object has been passed as a context variable from your view. You will need to modify your view to pass the correct data to the template.  
  
Also, the CSS file (`style.css`) and JavaScript file (`script.js`) mentioned in the template are assumed to be located in the same directory as the template, and have been loaded correctly. You may need to adjust these paths depending on your project structure.