Django\_template

Here's an example of how you can implement this feature using Django Templates:  
  
\*\*order\_placement.html\*\*  
  
```html  
<!-- Order Placement Template -->  
<div class="container">  
 <h1>Order Placement</h1>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Submit</button>  
 </form>  
  
 <!-- Display placed orders -->  
 <table id="placed-orders-table">  
 <thead>  
 <tr>  
 <th>Order ID</th>  
 <th>Customer Name</th>  
 <th>Date Placed</th>  
 <th>Total Cost</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for order in orders %}  
 <tr>  
 <td>{{ order.id }}</td>  
 <td>{{ order.customer.name }}</td>  
 <td>{{ order.date\_placed }}</td>  
 <td>{{ order.total\_cost }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
</div>  
  
<!-- Include CSS file for styling -->  
<link rel="stylesheet" href="{% static'styles/style.css' %}">  
```  
  
\*\*style.css\*\*  
  
```css  
/\* Style for Order Placement page \*/  
.container {  
 max-width: 800px;  
 margin: 40px auto;  
}  
  
table {  
 border-collapse: collapse;  
}  
  
th, td {  
 border: 1px solid #ddd;  
 padding: 10px;  
 text-align: left;  
}  
```  
  
\*\*script.js\*\*  
  
```javascript  
// Script for handling table sorting  
$(document).ready(function() {  
 $('#placed-orders-table').DataTable();  
});  
```  
  
This template includes a basic form for placing new orders, along with a table displaying all previously placed orders. You'll need to update your views and serializer to handle the data correctly, but this should give you a good starting point! Let me know if you have any further questions or need more assistance.

Here's an example of how you can implement this feature using Django Templates:  
  
\*\*order\_placement.html\*\*  
  
```html  
<!-- Order Placement Template -->  
<div class="container">  
 <h1>Order Placement</h1>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Submit</button>  
 </form>  
  
 <!-- Display placed orders -->  
 <table id="placed-orders-table">  
 <thead>  
 <tr>  
 <th>Order ID</th>  
 <th>Customer Name</th>  
 <th>Date Placed</th>  
 <th>Total Cost</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for order in orders %}  
 <tr>  
 <td>{{ order.id }}</td>  
 <td>{{ order.customer.name }}</td>  
 <td>{{ order.date\_placed }}</td>  
 <td>{{ order.total\_cost }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
</div>  
  
<!-- Include CSS file for styling -->  
<link rel="stylesheet" href="{% static'styles/style.css' %}">  
```  
  
\*\*style.css\*\*  
  
```css  
/\* Style for Order Placement page \*/  
.container {  
 max-width: 800px;  
 margin: 40px auto;  
}  
  
table {  
 border-collapse: collapse;  
}  
  
th, td {  
 border: 1px solid #ddd;  
 padding: 10px;  
 text-align: left;  
}  
```  
  
\*\*script.js\*\*  
  
```javascript  
// Script for handling table sorting  
$(document).ready(function() {  
 $('#placed-orders-table').DataTable();  
});  
```  
  
This template includes a basic form for placing new orders, along with a table displaying all previously placed orders. You'll need to update your views and serializer to handle the data correctly, but this should give you a good starting point! Let me know if you have any further questions or need more assistance.

Here's an example of how you can implement this feature using Django Templates:  
  
\*\*order\_placement.html\*\*  
  
```html  
<!-- Order Placement Template -->  
<div class="container">  
 <h1>Order Placement</h1>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Submit</button>  
 </form>  
  
 <!-- Display placed orders -->  
 <table id="placed-orders-table">  
 <thead>  
 <tr>  
 <th>Order ID</th>  
 <th>Customer Name</th>  
 <th>Date Placed</th>  
 <th>Total Cost</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for order in orders %}  
 <tr>  
 <td>{{ order.id }}</td>  
 <td>{{ order.customer.name }}</td>  
 <td>{{ order.date\_placed }}</td>  
 <td>{{ order.total\_cost }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
</div>  
  
<!-- Include CSS file for styling -->  
<link rel="stylesheet" href="{% static'styles/style.css' %}">  
```  
  
\*\*style.css\*\*  
  
```css  
/\* Style for Order Placement page \*/  
.container {  
 max-width: 800px;  
 margin: 40px auto;  
}  
  
table {  
 border-collapse: collapse;  
}  
  
th, td {  
 border: 1px solid #ddd;  
 padding: 10px;  
 text-align: left;  
}  
```  
  
\*\*script.js\*\*  
  
```javascript  
// Script for handling table sorting  
$(document).ready(function() {  
 $('#placed-orders-table').DataTable();  
});  
```  
  
This template includes a basic form for placing new orders, along with a table displaying all previously placed orders. You'll need to update your views and serializer to handle the data correctly, but this should give you a good starting point! Let me know if you have any further questions or need more assistance.

Here's an example of how you can implement this feature using Django Templates:  
  
\*\*order\_placement.html\*\*  
  
```html  
<!-- Order Placement Template -->  
<div class="container">  
 <h1>Order Placement</h1>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Submit</button>  
 </form>  
  
 <!-- Display placed orders -->  
 <table id="placed-orders-table">  
 <thead>  
 <tr>  
 <th>Order ID</th>  
 <th>Customer Name</th>  
 <th>Date Placed</th>  
 <th>Total Cost</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for order in orders %}  
 <tr>  
 <td>{{ order.id }}</td>  
 <td>{{ order.customer.name }}</td>  
 <td>{{ order.date\_placed }}</td>  
 <td>{{ order.total\_cost }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
</div>  
  
<!-- Include CSS file for styling -->  
<link rel="stylesheet" href="{% static'styles/style.css' %}">  
```  
  
\*\*style.css\*\*  
  
```css  
/\* Style for Order Placement page \*/  
.container {  
 max-width: 800px;  
 margin: 40px auto;  
}  
  
table {  
 border-collapse: collapse;  
}  
  
th, td {  
 border: 1px solid #ddd;  
 padding: 10px;  
 text-align: left;  
}  
```  
  
\*\*script.js\*\*  
  
```javascript  
// Script for handling table sorting  
$(document).ready(function() {  
 $('#placed-orders-table').DataTable();  
});  
```  
  
This template includes a basic form for placing new orders, along with a table displaying all previously placed orders. You'll need to update your views and serializer to handle the data correctly, but this should give you a good starting point! Let me know if you have any further questions or need more assistance.

Here's an example of how you can implement this feature using Django Templates:  
  
\*\*order\_placement.html\*\*  
  
```html  
<!-- Order Placement Template -->  
<div class="container">  
 <h1>Order Placement</h1>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Submit</button>  
 </form>  
  
 <!-- Display placed orders -->  
 <table id="placed-orders-table">  
 <thead>  
 <tr>  
 <th>Order ID</th>  
 <th>Customer Name</th>  
 <th>Date Placed</th>  
 <th>Total Cost</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for order in orders %}  
 <tr>  
 <td>{{ order.id }}</td>  
 <td>{{ order.customer.name }}</td>  
 <td>{{ order.date\_placed }}</td>  
 <td>{{ order.total\_cost }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
</div>  
  
<!-- Include CSS file for styling -->  
<link rel="stylesheet" href="{% static'styles/style.css' %}">  
```  
  
\*\*style.css\*\*  
  
```css  
/\* Style for Order Placement page \*/  
.container {  
 max-width: 800px;  
 margin: 40px auto;  
}  
  
table {  
 border-collapse: collapse;  
}  
  
th, td {  
 border: 1px solid #ddd;  
 padding: 10px;  
 text-align: left;  
}  
```  
  
\*\*script.js\*\*  
  
```javascript  
// Script for handling table sorting  
$(document).ready(function() {  
 $('#placed-orders-table').DataTable();  
});  
```  
  
This template includes a basic form for placing new orders, along with a table displaying all previously placed orders. You'll need to update your views and serializer to handle the data correctly, but this should give you a good starting point! Let me know if you have any further questions or need more assistance.