# Assignment

Create a simple listing component in Umbraco using React. This component should allow users to:

- View a list of items
- Filter the list by dropdown values
- Sort the list (e.g., by Name)
- Navigate through pages (basic pagination)

### Content Setup

All list items will be stored as child nodes under one parent node in the Umbraco Content Tree.

Each item should include:

- Name: Text field
- Description: Text area
- Image: Upload field (Media Picker)
- Brand: Dropdown (choose from: Brand 1, Brand 2, Brand 3, Brand 4)
- Type: Dropdown (Type 1, Type 2, Type 3)
- Width: Dropdown (100, 150, 200, 250, 300, 350)

## Suggested Approach:

1. Set up document types in Umbraco to match the item fields.
2. Use Umbraco API (e.g., Content Delivery API or custom controller) to return the list of items as JSON.
3. Build a React component to fetch and display the items.
4. Add basic filtering (e.g., by Brand, Type) and sorting (e.g., alphabetical by Name).
5. Implement simple pagination (e.g., show 5–10 items per page).

### 🗂️ Project Structure

```
/UmbracoReactListing
│
├── /wwwroot/scripts
│   └── listing.bundle.js       --> Webpack output
│
├── /Controllers
```

```
│   └── ListingApiController.cs     --> API controller
│
├── /Views
│   └── Listing.cshtml              --> Razor view rendering React component
│
├── /frontend
│   ├── /src
│   │   └── listing.jsx             --> React component source
│   ├── package.json
│   └── webpack.config.js
│
├── UmbracoReactListing.csproj
└── ...
```

## 🛠️ Umbraco Setup

**Create a Document Type for Listing Item**

- **Name**: Listing Item
- **Allow as child of**: Your parent node
- **Properties**:
    - Name (Textstring)
    - Description (Textarea)
    - Image (Image/Media Picker)
    - Brand (Dropdown): Brand 1, Brand 2, Brand 3, Brand 4
    - Type (Dropdown): Type 1, Type 2, Type 3
    - Width (Dropdown): 100, 150, 200, 250, 300, 350

## 🔄 API Controller (C#)

```
// /Controllers/ListingApiController.cs
using System.Linq;
using System.Web.Http;
using Umbraco.Web.WebApi;
using Umbraco.Core.Models.PublishedContent;
using Umbraco.Web;

public class ListingApiController : UmbracoApiController
{
    [HttpGet]
    public object GetItems()
    {
```

```
        var rootNode = Umbraco.Content(Guid.Parse("YOUR-PARENT-NODE-GUID"));
        var items = rootNode.Children().Select(x => new
        {
            Name = x.Value<string>("name"),
            Description = x.Value<string>("description"),
            Image = x.Value<IPublishedContent>("image")?.Url(),
            Brand = x.Value<string>("brand"),
            Type = x.Value<string>("type"),
            Width = x.Value<string>("width")
        });

        return items;
    }
}
```

Replace "`YOUR-PARENT-NODE-GUID`" with the actual GUID of your content node.

### Razor View (.cshtml)

```
@inherits Umbraco.Web.Mvc.UmbracoViewPage
<div id="react-listing"></div>
<script src="/App_Plugins/ReactListingComponent/listing.js"></script>
```

### React Component

```
// /frontend/src/listing.jsx
import React, { useEffect, useState } from 'react';
import ReactDOM from 'react-dom';

function ListingComponent() {
  const [items, setItems] = useState([]);

  useEffect(() => {
    fetch('/umbraco/api/ListingApi/GetItems')
      .then(res => res.json())
      .then(data => setItems(data));
  }, []);

  return (
    <div>
      <h2>Item List</h2>
      {items.map((item, i) => (
        <div key={i} style={{ border: '1px solid #ccc', padding: 10, marginBottom: 10 }}>
          <h3>{item.name}</h3>
          <img src={item.image} alt={item.name} width="150" />
          <p>{item.description}</p>
          <p><strong>Brand:</strong> {item.brand}</p>
```

```
          <p><strong>Type:</strong> {item.type}</p>
          <p><strong>Width:</strong> {item.width}</p>
        </div>
      ))}
    </div>
  );
}

ReactDOM.render(<ListingComponent />, document.getElementById('react-listing'));
```