



App Fullstack Developer Program
Backend Project 1 - Server Side Rendered Product Catalogue

Introduction

In this project we will create a server side rendered application for product listing. It consists of 4 pages:

1. Product Listing Page.
2. Product Details Page.
3. Manufacturer Listing Page
4. Manufacturer Page.

Task - Set up the Express application

Instruction

1. Execute the command:

```
npx express-generator --ejs project-express-ssr
```

2. The command should have created a new folder for you.
3. Change directory into that new folder:

```
cd project-express-ssr
```

4. Install all dependencies listed in the package.json file:

```
npm install
```

5. Start the Express web server by running one of the following commands:

Windows PowerShell:

```
$env:DEBUG='project-express-ssr:*'; npm start
```

Windows Command Prompt:

```
set DEBUG=project-express-ssr:* & npm start
```

MacOs or Linus:

```
SET DEBUG=project-express-ssr:* & npm start
```

6. Access the Express application by loading <http://localhost:3000/>

Task - Set up routes and template files

Firstly let's prepare all the routes for this application. Add the following codes into

“/routes/index.js”

```
/* GET home page. */
router.get("/", function (req, res, next) {
  res.redirect(302, "products");
});

/* GET products listing page. */
router.get("/products", function (req, res, next) {
  res.render("products", {
    title: "Product Listing",
    description: "This page shows a list of products.",
  });
});

/* GET product page. */
router.get("/products/:id", function (req, res, next) {
  res.render("product", {
    title: "Product Page",
    description: "This page shows the details of a product",
  });
});

/* GET manufacturer listing page. */
router.get("/manufacturers", function (req, res, next) {
  res.render("manufacturers", {
    title: "Manufacturer Page",
    description: "This page shows a list of manufacturers."
  });
});

/* GET manufacturer page. */
router.get("/manufacturers/:id", function (req, res, next) {
  res.render("manufacturer", {
    title: "Manufacturer Page",
    description: "This page shows a list of products from this manufacturer.",
  });
});
```

From the routes, we can see that we have 4 routes and the **home page route will redirect the user to the product listing page.**

After this, please also create the corresponding template files:

- “/views/products.ejs”
- “/views/product.ejs”

</talentlabs>

- “/views/manufacturers.ejs”
- “/views/manufacturer.ejs”

Task - Set up the basic HTML template.

Copy and paste the following code to **each of the .ejs template files created above**. Below a **starter code that involves the UI library Bootstrap**:

```
<!DOCTYPE html>
<html>

<head>
  <title>
    <%= title %>
  </title>
  <!-- Latest compiled and minified CSS -->
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"

integrity="sha384-HSMxcRTRxN+Bdg0JdbxYKrThec0KuH5zCYotlSAcp1+c8xmyTe9GYg1l9a69psu"
crossorigin="anonymous">

  <!-- Latest compiled and minified JavaScript -->
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"

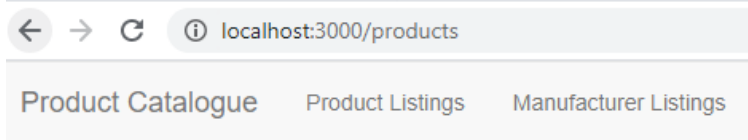
integrity="sha384-aJ210j1MXNL5UyIl/XNwTMqvzeRMZH2w8c5cRVpzpU8Y5bApTppSuUkhZXN0VxHd"
crossorigin="anonymous"></script>
</head>

<body>
  <nav class="navbar navbar-default">
    <div class="container-fluid">
      <div class="navbar-header">
        <a class="navbar-brand" href="/">
          Product Catalogue
        </a>
      </div>
      <div class="nav navbar-nav">
        <li><a href="/products">Product Listings</a></li>
        <li><a href="/manufacturers">Manufacturer Listings</a></li>
      </ul>
    </div>
  </nav>
  <div class="container-fluid">
    <p>
      <%= description %>
    </p>
  </div>
</body>

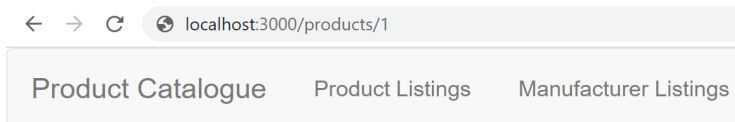
</html>
```

</talentlabs>

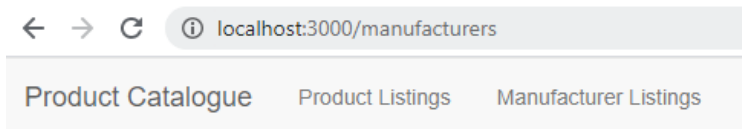
After this please make sure you can see the following **4 pages**:



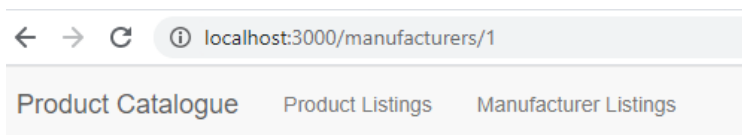
This page shows a list of products.



This page shows the details of a product



This page shows a list of manufacturers.



This page shows a list of products from this manufacturer.

Task - Add the data

In this project, the data will be **split into 2 arrays (manufacturers and products)**. Given a product object, we can **find out its manufacturer details using the manufacturer_id**.

Note that the imageUrl in the data is just a placeholder image, you will need to customize it later.

Please add the following data to **"/routes/index.js" just before all the routes.**

```
var manufacturers = [
  {
    id: 1,
    name: "Lego",
    imageUrl: "https://via.placeholder.com/150",
  },
  {
    id: 2,
    name: "Disney",
    imageUrl: "https://via.placeholder.com/150",
  },
];

var products = [
  {
    id: 1,
    name: "Lego City 2824: Advent Calendar 2010",
    price: 3.42,
    imageUrl: "https://via.placeholder.com/150",
    description: "description placeholder",
    manufacturerId: 1,
  },
  {
    id: 2,
    name: "LEGO Friends 41016: Advent Calendar",
    price: 24.95,
    imageUrl: "https://via.placeholder.com/150",
    description: "description placeholder",
    manufacturerId: 1,
  },
  {
    id: 3,
    name: "LEGO Star Wars 75018: Jek-14's Stealth Starfighter",
    price: 68.87,
    imageUrl: "https://via.placeholder.com/150",
    description: "description placeholder",
    manufacturerId: 1,
  },
  {
    id: 4,
```

</talentlabs>

```
name: "Disney Phineas and Ferb 8 Ferb Plush, soft, cuddle doll toy",
price: 19.99,
imageUrl: "https://via.placeholder.com/150",
description: "description placeholder",
manufacturerId: 2,
},
{
  id: 5,
  name:
    "DESPICABLE ME 2 - Minion cuddly Soft Toy - Plush Figures Banana 28-33 cm, Minion
Typ:Bob",
  price: 19.99,
  imageUrl: "https://via.placeholder.com/150",
  description: "description placeholder",
  manufacturerId: 2,
},
];
```

Task - Product Listing Page

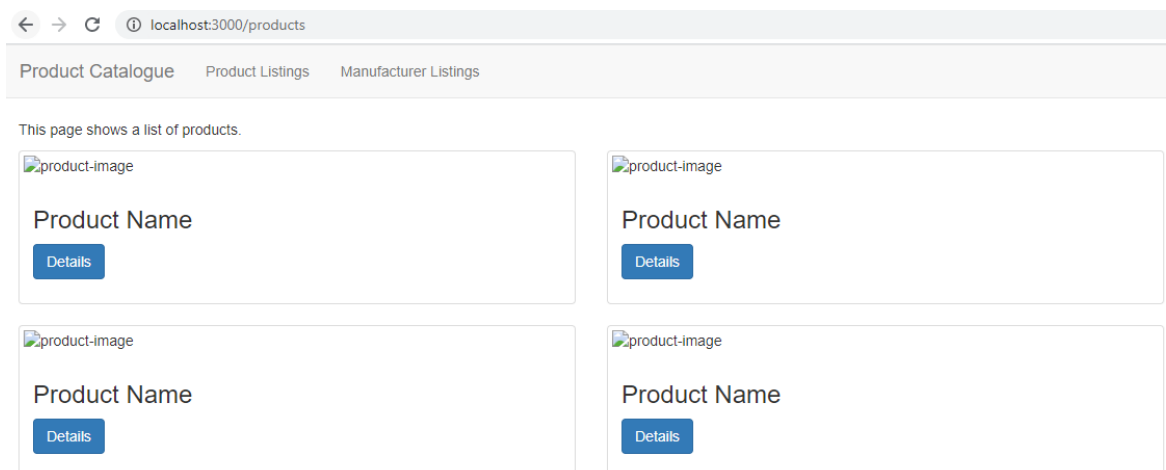
Please take a look at Lecture 3 EJS HTML Template for this task.

Firstly, add the products variable as a **rendering context** for the template "products.ejs" in **the products listing page route**.

Next, Add the following starter code to **"/views/products.ejs"**:

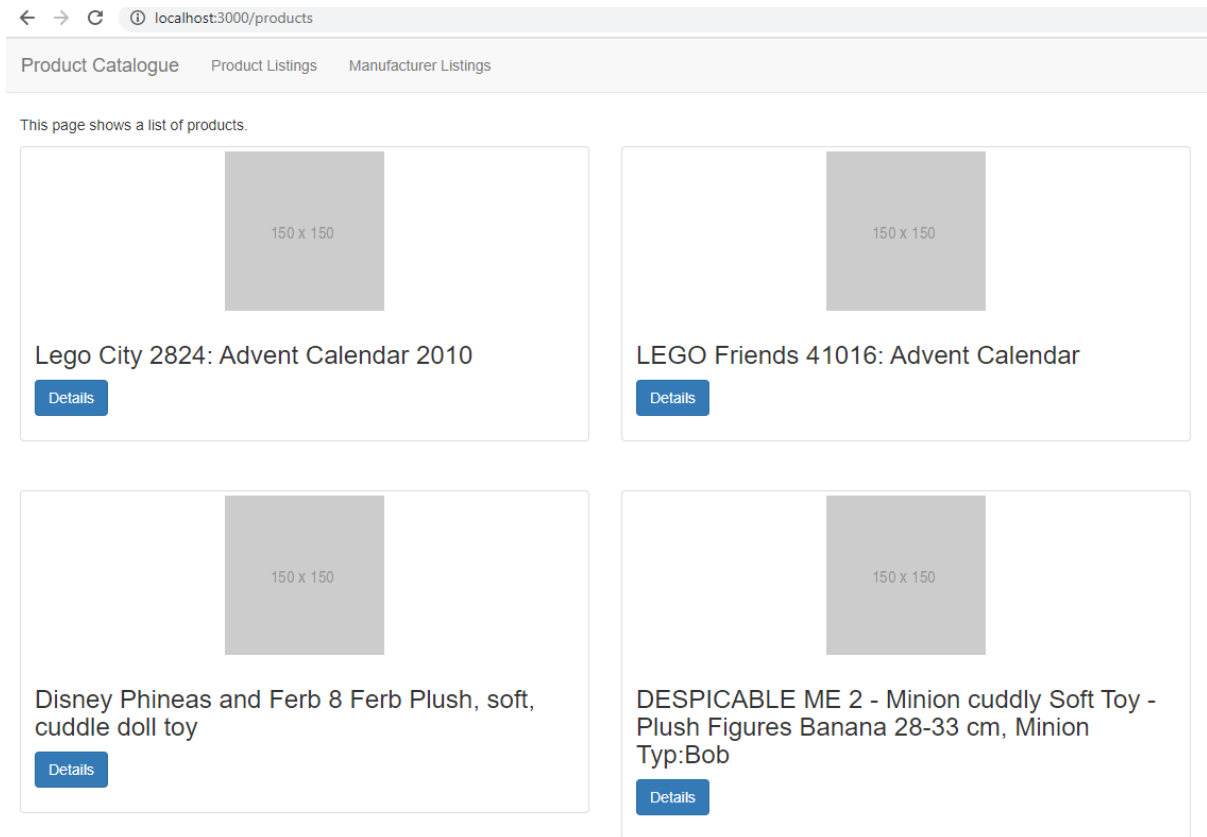
```
<p>
  <%= description%>
</p>
<div class="row">
  <% products.forEach(function(p){ %>
    <div class="col-sm-6 col-md-4">
      <div class="thumbnail">
        
        <div class="caption">
          <h3>Product Title</h3>
          <p>
            <a href="#" class="btn btn-primary">Details</a>
          </p>
        </div>
      </div>
    </div>
  <% }); %>
</div>
```

You should see this at this point:



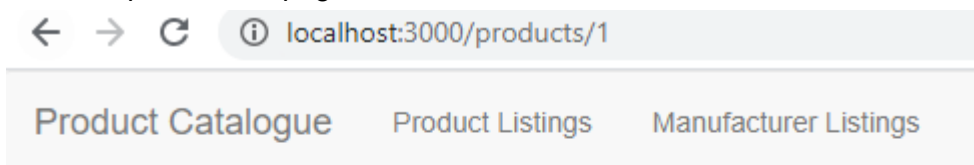
</talentlabs>

Please **update the template file** to make the page look like this:



Hints: you can use the EJS placeholder, if you want to replace a string inside a HTML attribute, you can do this: **href="/products/<%= p.id %>"**

Also, when the **details button is clicked, it should bring the user to the corresponding product page**. For example, when the product with id =1 is clicked, it will bring the user to visit the “/products/1” page.



Task - Product Page

First, we need to update the route for the product page. Please take a look at Lecture 4 Introduction to Express Routing for this task.

Replace the product page route with the following code, please **update the code to accept the request parameter correctly.**

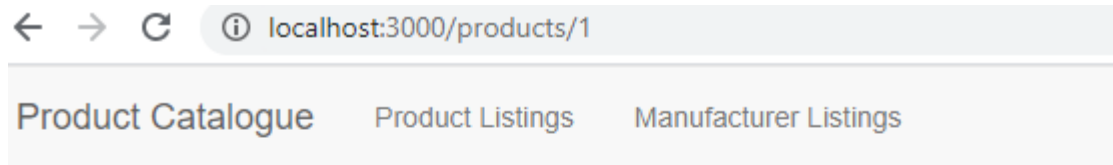
```
router.get("/products/:id", function (req, res, next) {  
  // Fill in the code: get the parameter  
  var requestedId = -1;  
  
  // Get the requested product from the product list  
  var requestedProduct = products.filter(function (product) {  
    return product.id === requestedId;  
  });  
  
  // Check if the requested product id exist  
  if (requestedProduct.length > 0) {  
    res.render("product", {  
      title: "Product Page",  
      product: requestedProduct[0],  
      description: "This page shows the details of a product",  
    });  
  } else {  
    // 404 Product not found  
    res.status(404).send("Product not found");  
  }  
});
```

Next, Add the following starter code to **"/views/product.ejs"**:

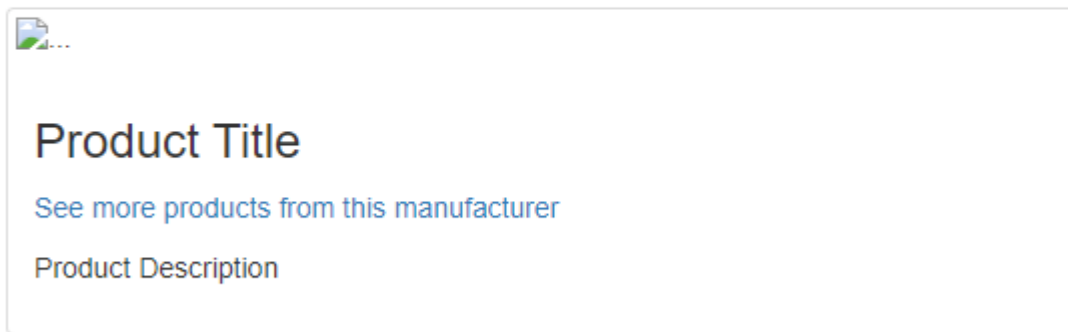
```
<p>  
  <%= description %>  
</p>  
<div class="row">  
  <div class="col-sm-6 col-md-4">  
    <div class="thumbnail">  
        
      <div class="caption">  
        <h3>Product Title</h3>  
        <p>  
          <a href="">See more products from this manufacturer</a>  
        <p>  
        <p>Product Description</p>  
      </div>  
    </div>  
  </div>  
</div>
```

</talentlabs>

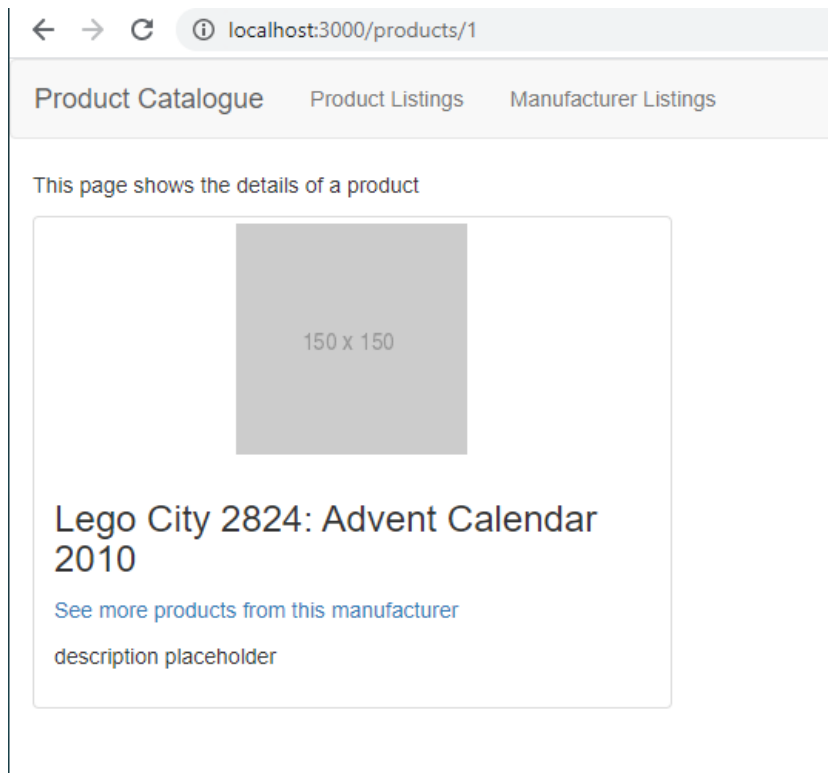
You should see this at this point:



This page shows the details of a product



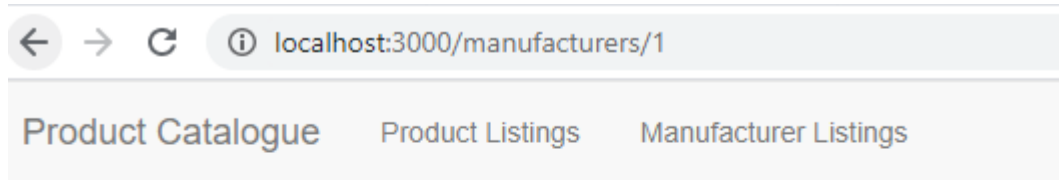
Please **update the template file** to make the page look like this:



</talentlabs>

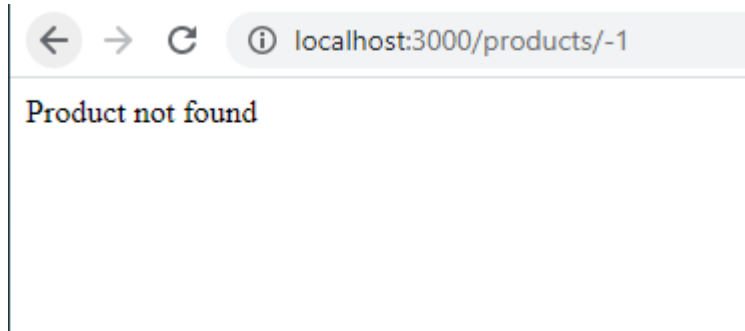
Hints: you can use the EJS placeholder, if you want to replace a string inside a HTML attribute, you can do this: **href="/products/<%= p.id %>"**

Also, when the **"see more"** is clicked, it should bring the user to the corresponding **manufacturer page**. For example, when the manufacturer with id =1 is clicked, it will bring the user to visit the **"/manufacturers/1"** page.



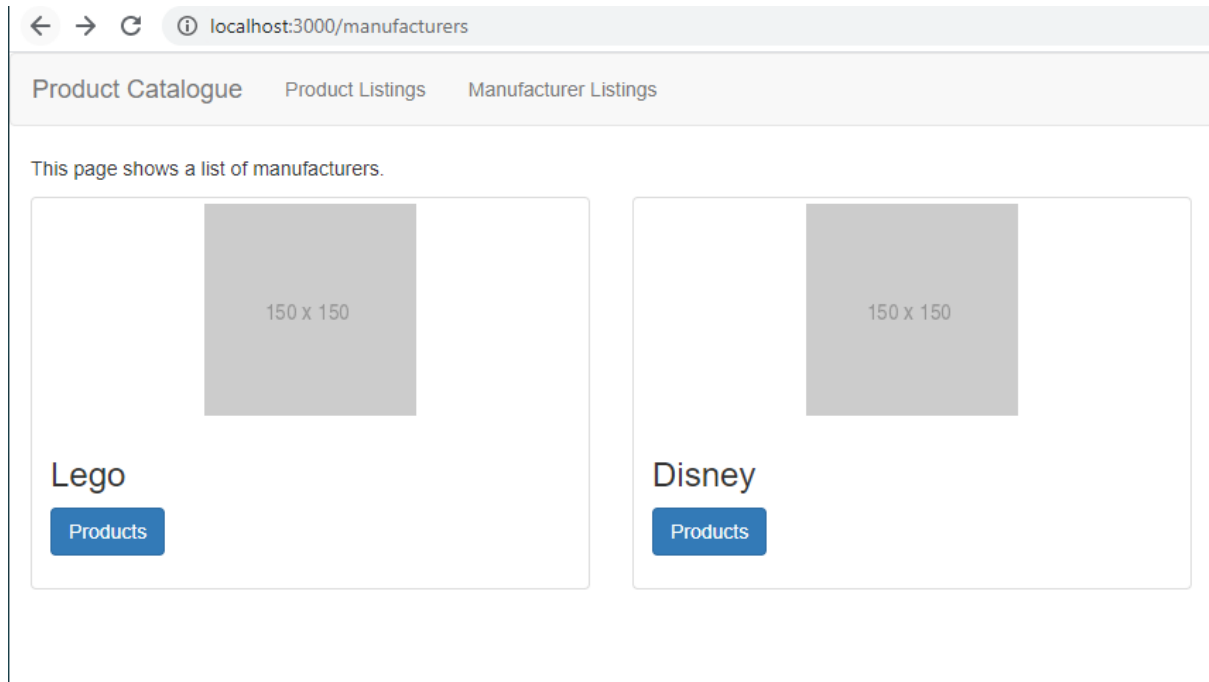
This page shows a list of products from this manufacturer.

If the requested **product id is not valid, the page should show an error:**



Task - Manufacturer Listing Page

Update the route and template file to produce the following result, the implementations should be similar to the previous tasks.



Task - Manufacturer Page

First, we need to update the route for the manufacturer page.

Replace the manufacturer page route with the following code, please **update the code to accept the request parameter correctly.**

```
router.get("/manufacturers/:id", function (req, res, next) {
  // Fill in the code: get the parameter
  var requestedId = -1;

  // Get the requested product from the product list
  var requestedProducts = products.filter(function (product) {
    return product.manufacturerId == requestedId;
  });

  // Check if the requested product id exist
  res.render("products", {
    title: "Manufacturer Page",
    products: requestedProducts,
```

</talentlabs>

```
description: "This page shows a list of products from this manufacturer.",
});
});0
```

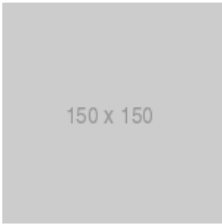
Notice that **we are using template "products.ejs" now**. It is because we can actually **reuse that template for listing all products of a manufacturer**.

After this task, you should see this for a manufacturer:

[←](#) [→](#) [↻](#) [localhost:3000/manufacturers/1](#)


[Product Catalogue](#) [Product Listings](#) [Manufacturer Listings](#)

This page shows a list of products from this manufacturer.



Lego City 2824: Advent Calendar 2010

[Details](#)



LEGO Friends 41016: Advent Calendar

[Details](#)

</talentlabs>

Task - Replace the fake data with real data

Visit any existing online store and replace the manufacturers and products arrays with real world data.

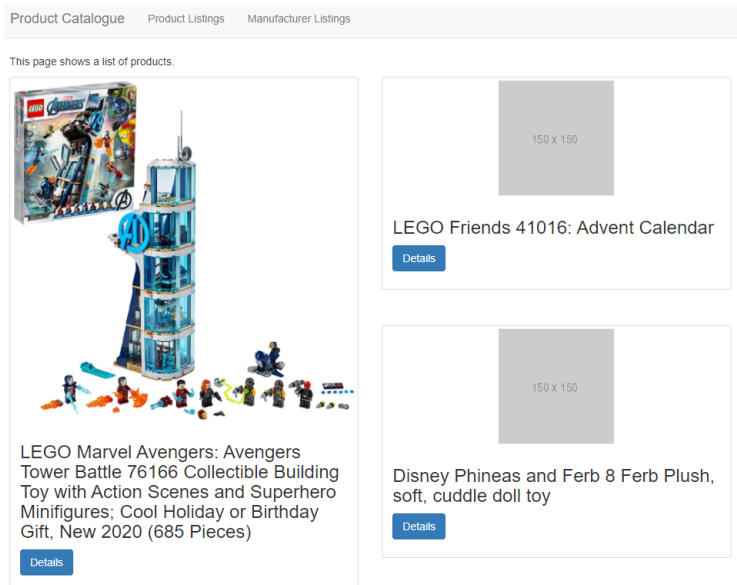
For example for this [product](#), **Steps**:

1. Copy the product name.
2. Copy part of the product description
3. Copy the product price
4. Download the image to the **/public/images** folder.

The product object becomes this

```
{
  id: 1,
  name: "LEGO Marvel Avengers: Avengers Tower Battle 76166 Collectible Building Toy with Action Scenes and Superhero Minifigures; Cool Holiday or Birthday Gift, New 2020 (685 Pieces)",
  price: 89.95,
  imageUrl: "/images/product_1.jpg",
  description: "With 5 floors and 7 feature-packed rooms, the LEGO Marvel Avengers: Avengers Tower Battle (76166) is bursting with role-play possibilities with classic Marvel characters",
  manufacturerId: 1,
}
```

The result should look similar to this:



Please update and add more real world products and brands to manufacturers and products arrays and **make the website realistic!**