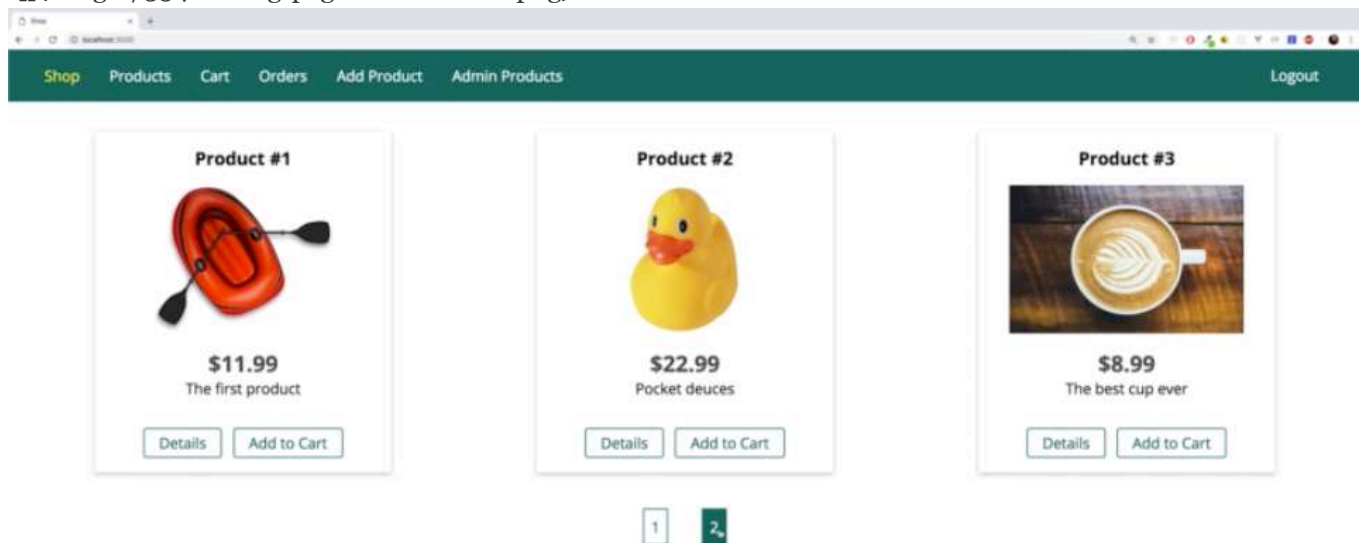


21. Adding Pagination

* Chapter 334: Adding Pagination Links

1. update
 - ./views/shop/index.ejs
 - ./public/css/main.css



```
1 <!--./views/shop/index.ejs-->
2
3 <%= include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/product.css">
5 </head>
6
7 <body>
8   <%= include('../includes/navigation.ejs') %>
9
10  <main>
11    <% if (prods.length > 0) { %>
12      <div class="grid">
13        <% for (let product of prods) { %>
14          <article class="card product-item">
15            <header class="card_header">
16              <h1 class="product_title"><%= product.title %></h1>
17            </header>
18            <div class="card_image">
19              "
21     </div>
22     <div class="card_content">
23         <h2 class="product_price">$<%= product.price %></h2>
24         <p class="product_description"><%= product.description %></p>
25     </div>
26     <div class="card_actions">
27         <a href="/products/<%= product._id %>" class="btn">Details</a>
28         <% if (isAuthenticated) { %>
29             <%= include('../includes/add-to-cart.ejs', {product:
product}) %>
30         <% } %>
31     </div>
32 </article>
33 <% } %>
34 </div>
35 <section class="pagination">
36     <a href="/?page=1">1</a>
37     <a href="/?page=2">2</a>
38 </section>
39 <% } else { %>
40     <h1>No Products Found!</h1>
41 <% } %>
42 </main>
43 <%= include('../includes/end.ejs') %>

```

```

1 /*./public/css/main.css*/
2
3 @import url('https://fonts.googleapis.com/css?family=Open+Sans:400,700');
4
5 * {
6     box-sizing: border-box;
7 }
8
9 body {
10     padding: 0;
11     margin: 0;
12     font-family: 'Open Sans', sans-serif;
13 }
14
15 main {
16     padding: 1rem;
17     margin: auto;
18 }
19
20 form {
21     display: inline;
22 }
23
24 .centered {
25     text-align: center;
26 }
27
28 .image {
29     height: 20rem;
30 }
31

```

```
32 .image img {
33   height: 100%;
34 }
35
36 .main-header {
37   width: 100%;
38   height: 3.5rem;
39   background-color: #00695c;
40   padding: 0 1.5rem;
41   display: flex;
42   align-items: center;
43 }
44
45 .main-header__nav {
46   height: 100%;
47   width: 100%;
48   display: none;
49   align-items: center;
50   justify-content: space-between;
51 }
52
53 .main-header__item-list {
54   list-style: none;
55   margin: 0;
56   padding: 0;
57   display: flex;
58 }
59
60 .main-header__item {
61   margin: 0 1rem;
62   padding: 0;
63 }
64
65 .main-header__item a,
66 .main-header__item button {
67   font: inherit;
68   background: transparent;
69   border: none;
70   text-decoration: none;
71   color: white;
72   cursor: pointer;
73 }
74
75 .main-header__item a:hover,
76 .main-header__item a:active,
77 .main-header__item a.active,
78 .main-header__item button:hover,
79 .main-header__item button:active {
80   color: #ffeb3b;
81 }
82
83 .mobile-nav {
84   width: 30rem;
85   height: 100vh;
86   max-width: 90%;
87   position: fixed;
```

```
88 left: 0;
89 top: 0;
90 background: white;
91 z-index: 10;
92 padding: 2rem 1rem 1rem 2rem;
93 transform: translateX(-100%);
94 transition: transform 0.3s ease-out;
95 }
96
97 .mobile-nav.open {
98   transform: translateX(0);
99 }
100
101 .mobile-nav__item-list {
102   list-style: none;
103   display: flex;
104   flex-direction: column;
105   margin: 0;
106   padding: 0;
107 }
108
109 .mobile-nav__item {
110   margin: 1rem;
111   padding: 0;
112 }
113
114 .mobile-nav__item a,
115 .mobile-nav__item button {
116   font: inherit;
117   text-decoration: none;
118   color: black;
119   font-size: 1.5rem;
120   padding: 0.5rem 2rem;
121   background: transparent;
122   border: none;
123   cursor: pointer;
124 }
125
126 .mobile-nav__item a:active,
127 .mobile-nav__item a:hover,
128 .mobile-nav__item a.active,
129 .mobile-nav__item button:hover,
130 .mobile-nav__item button:active {
131   background: #00695c;
132   color: white;
133   border-radius: 3px;
134 }
135
136 #side-menu-toggle {
137   border: 1px solid white;
138   font: inherit;
139   padding: 0.5rem;
140   display: block;
141   background: transparent;
142   color: white;
143   cursor: pointer;
```

```
144 }
145
146 #side-menu-toggle:focus {
147     outline: none;
148 }
149
150 #side-menu-toggle:active,
151 #side-menu-toggle:hover {
152     color: #ffeb3b;
153     border-color: #ffeb3b;
154 }
155
156 .backdrop {
157     position: fixed;
158     top: 0;
159     left: 0;
160     width: 100%;
161     height: 100vh;
162     background: rgba(0, 0, 0, 0.5);
163     z-index: 5;
164     display: none;
165 }
166
167 .grid {
168     display: flex;
169     flex-wrap: wrap;
170     justify-content: space-around;
171     align-items: stretch;
172 }
173
174 .card {
175     box-shadow: 0 2px 8px rgba(0, 0, 0, 0.26);
176 }
177
178 .card__header,
179 .card__content {
180     padding: 1rem;
181 }
182
183 .card__header h1,
184 .card__content h1,
185 .card__content h2,
186 .card__content p {
187     margin: 0;
188 }
189
190 .card__image {
191     width: 100%;
192 }
193
194 .card__image img {
195     width: 100%;
196 }
197
198 .card__actions {
199     padding: 1rem;
```

```
200     text-align: center;
201 }
202
203 .card__actions button,
204 .card__actions a {
205     margin: 0 0.25rem;
206 }
207
208 .btn {
209     display: inline-block;
210     padding: 0.25rem 1rem;
211     text-decoration: none;
212     font: inherit;
213     border: 1px solid #00695c;
214     color: #00695c;
215     background: white;
216     border-radius: 3px;
217     cursor: pointer;
218 }
219
220 .btn:hover,
221 .btn:active {
222     background-color: #00695c;
223     color: white;
224 }
225
226 .btn.danger {
227     color: red;
228     border-color: red;
229 }
230
231 .btn.danger:hover,
232 .btn.danger:active {
233     background: red;
234     color: white;
235 }
236
237 .user-message {
238     margin: auto;
239     width: 90%;
240     border: 1px solid #4771fa;
241     padding: 0.5rem;
242     border-radius: 3px;
243     background: #b9c9ff;
244     text-align: center;
245 }
246
247 .user-message--error {
248     border-color: red;
249     background: rgb(255, 176, 176);
250     color: red;
251 }
252
253 .pagination {
254     margin-top: 2rem;
255     text-align: center
```

```


256 }
257
258 .pagination a {
259     text-decoration: none;
260     color: #00695c;
261     padding: 0.5rem;
262     border: 1px solid #00695c;
263     margin: 0 1rem;
264 }
265
266 .pagination a:hover,
267 .pagination a:active {
268     background: #00695c;
269     color: white;
270 }
271
272 @media (min-width: 768px) {
273     .main-header__nav {
274         display: flex;
275     }
276
277     #side-menu-toggle {
278         display: none;
279     }
280
281     .user-message {
282         width: 30rem;
283     }
284 }
285

```

* Chapter 335: Retrieving A Chunk Of Data

1. update
 - ./controllers/shop.js

Product #1




\$11.99

The first product

Details

Add to Cart

Product #2




\$22.99

Pocket deuces

Details

Add to Cart

Product #1




\$11.99

The first product

Details

Add to Cart

Product #2

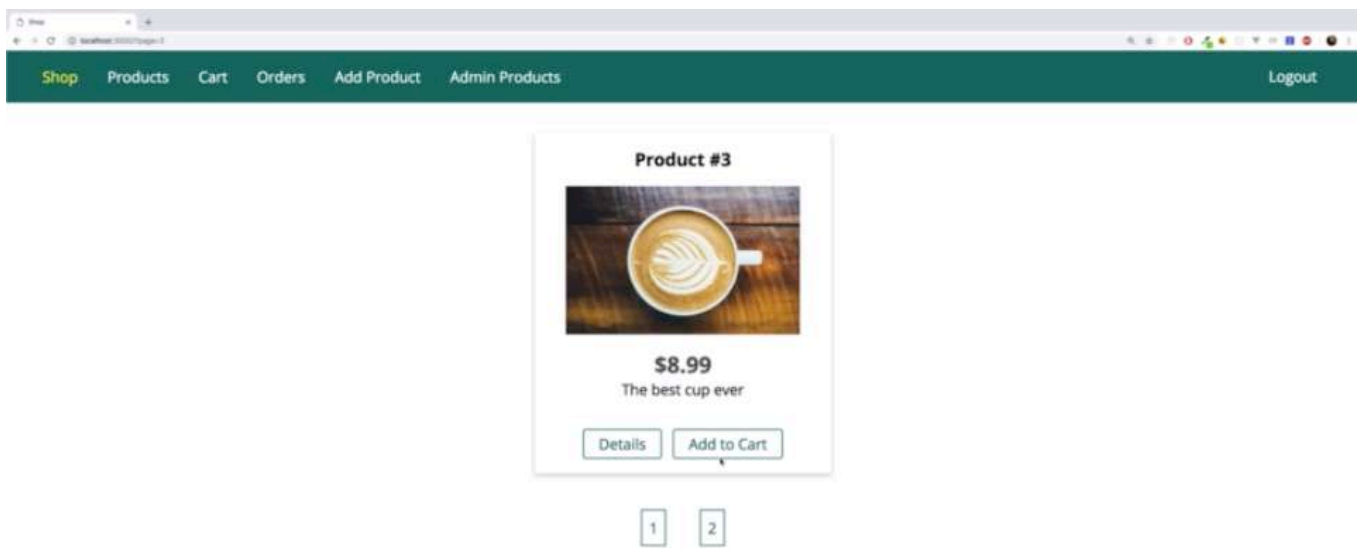


\$22.99

Pocket deuces

Details

Add to Cart





- if i type 'page3' in the end of the URL, 'No Products Found!' we would not find any products because i have no items for this page. but page 1, 2 is fine.

```
1 //./controllers/shop.js
2
3 const fs = require('fs');
4 const path = require('path');
5
6 const PDFDocument= require('pdfkit')
7
8 const Product = require('../models/product');
9 const Order = require('../models/order');
```

```

10
11 const ITEMS_PER_PAGE = 2
12
13 exports.getProducts = (req, res, next) => {
14   Product.find()
15     .then(products => {
16       console.log(products);
17       res.render('shop/product-list', {
18         prods: products,
19         pageTitle: 'All Products',
20         path: '/products'
21       });
22     })
23     .catch(err => {
24       const error = new Error(err);
25       error.httpStatusCode = 500;
26       return next(error);
27     });
28 };
29
30 exports.getProduct = (req, res, next) => {
31   const prodId = req.params.productId;
32   Product.findById(prodId)
33     .then(product => {
34       res.render('shop/product-detail', {
35         product: product,
36         pageTitle: product.title,
37         path: '/products'
38       });
39     })
40     .catch(err => {
41       const error = new Error(err);
42       error.httpStatusCode = 500;
43       return next(error);
44     });
45 };
46
47 exports.getIndex = (req, res, next) => {
48   /**request
49    * and we can get query parameters on the 'query' object
50    * provided by express.js
51    * and i can access page
52    * because i named that query parameter 'page' in the URL
53    */
54   const page = req.query.page
55
56   Product.find()
57   /**there is 'skip()' function
58    * we can skip x amount of results
59    * 'page - 1' is the previous page number
60    *
61    * '(page - 1) * ITEMS_PER_PAGE' means that
62    * if i'm on page 2,
63    * i would skip the first page,
64    * so the first one times ITEMS_PER_PAGE items
65    * so result is that

```

```

66 * i would skip the first one * 2
67 * so the first 2 items
68 */
69 .skip((page - 1) * ITEMS_PER_PAGE)
70 /**'limit()' method limits the amount of data we fetch to the number we specify
71 * and i can pass 'ITEMS_PER_PAGE'
72 * because that is my item limit per page.
73 */
74 .limit(ITEMS_PER_PAGE)
75 .then(products => {
76   res.render('shop/index', {
77     prods: products,
78     pageTitle: 'Shop',
79     path: '/'
80   });
81 })
82 .catch(err => {
83   const error = new Error(err);
84   error.httpStatusCode = 500;
85   return next(error);
86 });
87 };
88
89 exports.getCart = (req, res, next) => {
90   req.user
91     .populate('cart.items.productId')
92     .execPopulate()
93     .then(user => {
94       const products = user.cart.items;
95       res.render('shop/cart', {
96         path: '/cart',
97         pageTitle: 'Your Cart',
98         products: products
99       });
100     })
101     .catch(err => {
102       const error = new Error(err);
103       error.httpStatusCode = 500;
104       return next(error);
105     });
106 };
107
108 exports.postCart = (req, res, next) => {
109   const prodId = req.body.productId;
110   Product.findById(prodId)
111     .then(product => {
112       return req.user.addToCart(product);
113     })
114     .then(result => {
115       console.log(result);
116       res.redirect('/cart');
117     })
118     .catch(err => {
119       const error = new Error(err);
120       error.httpStatusCode = 500;
121       return next(error);

```

```

122     });
123 };
124
125 exports.postCartDeleteProduct = (req, res, next) => {
126     const prodId = req.body.productId;
127     req.user
128         .removeFromCart(prodId)
129         .then(result => {
130             res.redirect('/cart');
131         })
132         .catch(err => {
133             const error = new Error(err);
134             error.httpStatusCode = 500;
135             return next(error);
136         });
137 };
138
139 exports.postOrder = (req, res, next) => {
140     req.user
141         .populate('cart.items.productId')
142         .execPopulate()
143         .then(user => {
144             const products = user.cart.items.map(i => {
145                 return { quantity: i.quantity, product: { ...i.productId._doc } };
146             });
147             const order = new Order({
148                 user: {
149                     email: req.user.email,
150                     userId: req.user
151                 },
152                 products: products
153             });
154             return order.save();
155         })
156         .then(result => {
157             return req.user.clearCart();
158         })
159         .then(() => {
160             res.redirect('/orders');
161         })
162         .catch(err => {
163             const error = new Error(err);
164             error.httpStatusCode = 500;
165             return next(error);
166         });
167 };
168
169 exports.getOrders = (req, res, next) => {
170     Order.find({ 'user.userId': req.user._id })
171         .then(orders => {
172             res.render('shop/orders', {
173                 path: '/orders',
174                 pageTitle: 'Your Orders',
175                 orders: orders
176             });
177         })

```

```

178     .catch(err => {
179         const error = new Error(err);
180         error.httpStatusCode = 500;
181         return next(error);
182     });
183 };
184
185 exports.getInvoice = (req, res, next) => {
186     const orderId = req.params.orderId;
187     Order.findById(orderId)
188         .then(order => {
189             if (!order) {
190                 return next(new Error('No order found.'));
191             }
192             if (order.user.userId.toString() !== req.user._id.toString()) {
193                 return next(new Error('Unauthorized'));
194             }
195             const invoiceName = 'invoice-' + orderId + '.pdf';
196             const invoicePath = path.join('data', 'invoices', invoiceName);
197
198             const pdfDoc = new PDFDocument()
199             res.setHeader('Content-Type', 'application/pdf');
200             res.setHeader(
201                 'Content-Disposition',
202                 'inline; filename="' + invoiceName + '"'
203             )
204             pdfDoc.pipe(fs.createWriteStream(invoicePath))
205             pdfDoc.pipe(res)
206
207             pdfDoc.fontSize(26).text('Invoice', {
208                 underline: true
209             })
210
211             pdfDoc.text('-----')
212             let totalPrice = 0
213             /**'products' is an array
214              * because we store this products in a database as an array.
215              */
216             order.products.forEach(prod => {
217                 totalPrice += prod.quantity * prod.product.price
218                 pdfDoc
219                     .fontSize(14)
220                     .text(
221                         prod.product.title +
222                         ' - ' +
223                         prod.quantity +
224                         ' x ' +
225                         '$' +
226                         prod.product.price)
227             })
228             pdfDoc.text('---')
229             pdfDoc.fontSize(20).text('Total Price: $' + totalPrice)
230
231             pdfDoc.end()
232             //fs.readFile(invoicePath, (err, data) => {
233             //    if (err) {

```

```

234 // return next(err);
235 // }
236 // res.setHeader('Content-Type', 'application/pdf');
237 // res.setHeader(
238 //   'Content-Disposition',
239 //   'inline; filename="' + invoiceName + '"'
240 // );
241 // res.send(data);
242 //});
243
244 //const file = fs.createReadStream(invoicePath)
245 //file.pipe(res)
246 })
247 .catch(err => next(err));
248 };
249

```

* Chapter 337: Preparing Pagination Data On The Server

1. update

- ./controllers/shop.js

```

1 //./controllers/shop.js
2
3 const fs = require('fs');
4 const path = require('path');
5
6 const PDFDocument= require('pdfkit')
7
8 const Product = require('../models/product');
9 const Order = require('../models/order');
10
11 const ITEMS_PER_PAGE = 2
12
13 exports.getProducts = (req, res, next) => {
14   Product.find()
15     .then(products => {
16       console.log(products);
17       res.render('shop/product-list', {
18         prods: products,
19         pageTitle: 'All Products',
20         path: '/products'
21       });
22     })
23     .catch(err => {
24       const error = new Error(err);
25       error.httpStatusCode = 500;
26       return next(error);
27     });
28 };
29
30 exports.getProduct = (req, res, next) => {
31   const prodId = req.params.productId;

```

```

32 Product.findById(prodId)
33   .then(product => {
34     res.render('shop/product-detail', {
35       product: product,
36       pageTitle: product.title,
37       path: '/products'
38     });
39   })
40   .catch(err => {
41     const error = new Error(err);
42     error.httpStatusCode = 500;
43     return next(error);
44   });
45 };
46
47 exports.getIndex = (req, res, next) => {
48   const page = req.query.page
49   let totalItems
50
51   Product
52     .find()
53     .count()
54     .then(numProducts => {
55       totalItems = numProducts
56       return Product
57         .find()
58         .skip((page - 1) * ITEMS_PER_PAGE)
59         .limit(ITEMS_PER_PAGE)
60     })
61     .then(products => {
62       res.render('shop/index', {
63         prods: products,
64         pageTitle: 'Shop',
65         path: '/',
66         totalProducts: totalItems,
67         /**'hasNextPage' is only be the case
68          * if the total number of items is greater than the page we are on.
69          * times ITEMS_PER_PAGE
70          */
71         hasNextPage: ITEMS_PER_PAGE * page < totalItems,
72         hasPreviousPage: page > 1,
73         nextPage: page + 1,
74         previousPage: page - 1,
75         lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
76       });
77     })
78     .catch(err => {
79       const error = new Error(err);
80       error.httpStatusCode = 500;
81       return next(error);
82     });
83 };
84
85 exports.getCart = (req, res, next) => {
86   req.user
87     .populate('cart.items.productId')

```

```

88     .execPopulate()
89     .then(user => {
90         const products = user.cart.items;
91         res.render('shop/cart', {
92             path: '/cart',
93             pageTitle: 'Your Cart',
94             products: products
95         });
96     })
97     .catch(err => {
98         const error = new Error(err);
99         error.httpStatusCode = 500;
100         return next(error);
101     });
102 };
103
104 exports.postCart = (req, res, next) => {
105     const prodId = req.body.productId;
106     Product.findById(prodId)
107     .then(product => {
108         return req.user.addToCart(product);
109     })
110     .then(result => {
111         console.log(result);
112         res.redirect('/cart');
113     })
114     .catch(err => {
115         const error = new Error(err);
116         error.httpStatusCode = 500;
117         return next(error);
118     });
119 };
120
121 exports.postCartDeleteProduct = (req, res, next) => {
122     const prodId = req.body.productId;
123     req.user
124     .removeFromCart(prodId)
125     .then(result => {
126         res.redirect('/cart');
127     })
128     .catch(err => {
129         const error = new Error(err);
130         error.httpStatusCode = 500;
131         return next(error);
132     });
133 };
134
135 exports.postOrder = (req, res, next) => {
136     req.user
137     .populate('cart.items.productId')
138     .execPopulate()
139     .then(user => {
140         const products = user.cart.items.map(i => {
141             return { quantity: i.quantity, product: { ...i.productId._doc } };
142         });
143         const order = new Order({

```



```

144     user: {
145       email: req.user.email,
146       userId: req.user
147     },
148     products: products
149   });
150   return order.save();
151 })
152 .then(result => {
153   return req.user.clearCart();
154 })
155 .then(() => {
156   res.redirect('/orders');
157 })
158 .catch(err => {
159   const error = new Error(err);
160   error.httpStatusCode = 500;
161   return next(error);
162 });
163 };
164
165 exports.getOrders = (req, res, next) => {
166   Order.find({ 'user.userId': req.user._id })
167     .then(orders => {
168       res.render('shop/orders', {
169         path: '/orders',
170         pageTitle: 'Your Orders',
171         orders: orders
172       });
173     })
174     .catch(err => {
175       const error = new Error(err);
176       error.httpStatusCode = 500;
177       return next(error);
178     });
179 };
180
181 exports.getInvoice = (req, res, next) => {
182   const orderId = req.params.orderId;
183   Order.findById(orderId)
184     .then(order => {
185       if (!order) {
186         return next(new Error('No order found.'));
187       }
188       if (order.user.userId.toString() !== req.user._id.toString()) {
189         return next(new Error('Unauthorized'));
190       }
191       const invoiceName = 'invoice-' + orderId + '.pdf';
192       const invoicePath = path.join('data', 'invoices', invoiceName);
193
194       const pdfDoc = new PDFDocument()
195       res.setHeader('Content-Type', 'application/pdf');
196       res.setHeader(
197         'Content-Disposition',
198         'inline; filename="' + invoiceName + '"'
199       )

```

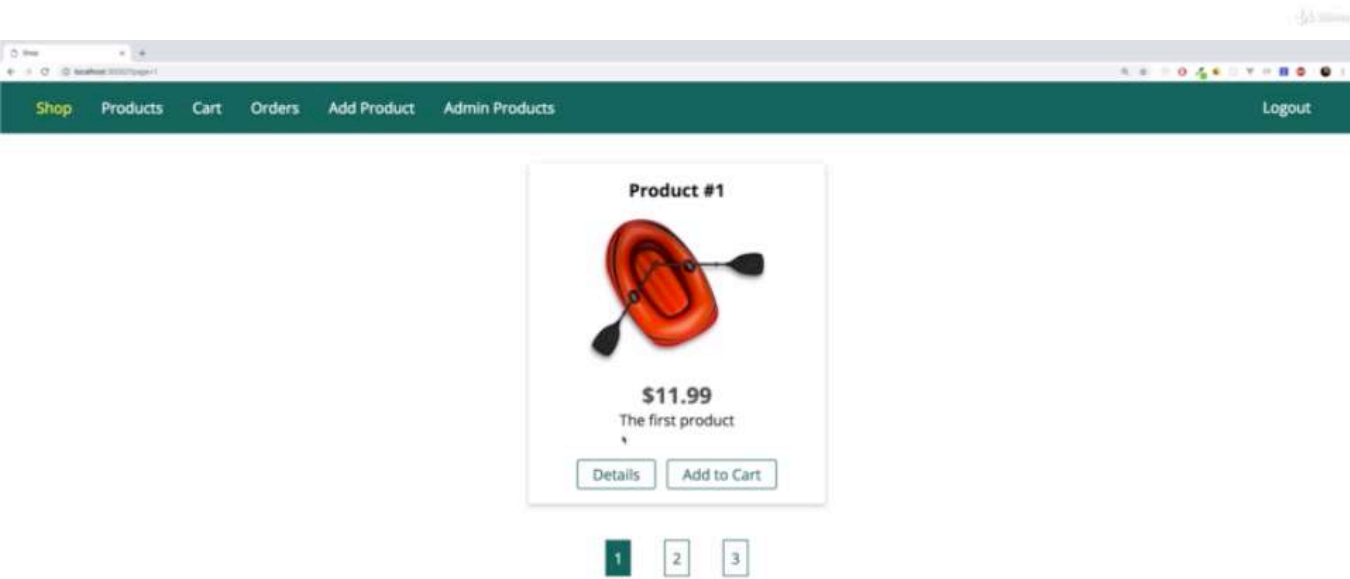
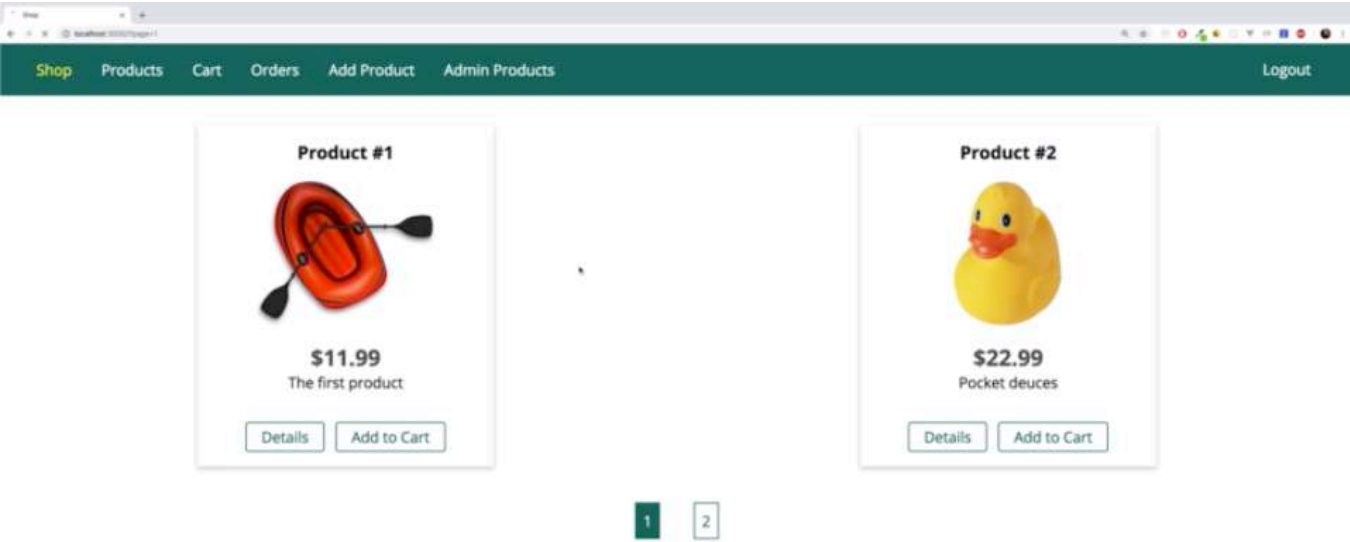
```

200 pdfDoc.pipe(fs.createWriteStream(invoicePath))
201 pdfDoc.pipe(res)
202
203 pdfDoc.fontSize(26).text('Invoice', {
204   underline: true
205 })
206
207 pdfDoc.text('-----')
208 let totalPrice = 0
209 order.products.forEach(prod => {
210   totalPrice += prod.quantity * prod.product.price
211   pdfDoc
212     .fontSize(14)
213     .text(
214       prod.product.title +
215       ' - ' +
216       prod.quantity +
217       ' x ' +
218       '$' +
219       prod.product.price)
220 })
221 pdfDoc.text('---')
222 pdfDoc.fontSize(20).text('Total Price: $' + totalPrice)
223
224 pdfDoc.end()
225 //fs.readFile(invoicePath, (err, data) => {
226 //  if (err) {
227 //    return next(err);
228 //  }
229 //  res.setHeader('Content-Type', 'application/pdf');
230 //  res.setHeader(
231 //    'Content-Disposition',
232 //    'inline; filename="' + invoiceName + '"'
233 //  );
234 //  res.send(data);
235 //});
236
237 //const file = fs.createReadStream(invoicePath)
238 //file.pipe(res)
239 })
240 .catch(err => next(err));
241 };
242


```

* Chapter 338: Adding Dynamic Pagination Buttons

1. update
 - ./controllers/shop.js
 - ./views/shop/index.ejs
 - ./public/css/main.css



Product #2



\$22.99

Pocket deuces

Details


Add to Cart

1

2

3

Product #1



\$11.99

The first product

Details


Add to Cart

1

2

3

Product #3



\$8.99

The best cup ever

Details

Add to Cart

1

2

3

Title

Product 4

Image

Choose file duck.jpg

Price


22

Description

fasdf

Add Product

Product #1



\$11.99


The first product

Details

Add to Cart

- 1
- 2
- 4

Product 4



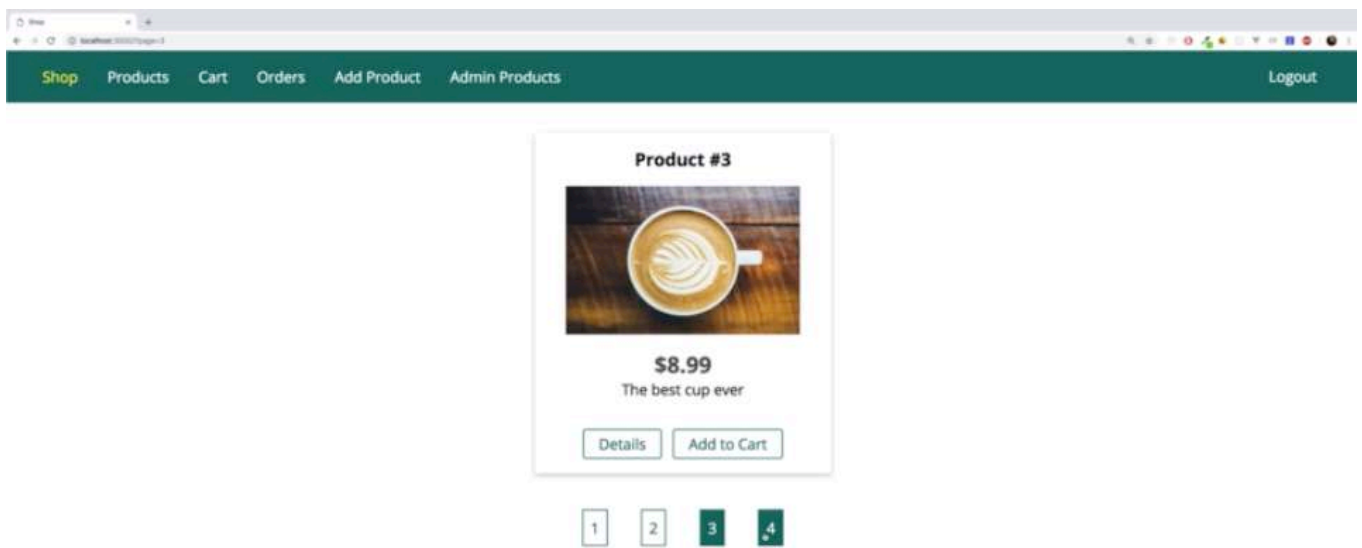
\$22

fasdf

Details

Add to Cart

- 1
- 3
- 4



```
1 //./controllers/shop.js
2
3 const fs = require('fs');
4 const path = require('path');
5
6 const PDFDocument = require('pdfkit');
7
8 const Product = require('../models/product');
9 const Order = require('../models/order');
10
11 const ITEMS_PER_PAGE = 2;
12
13 exports.getProducts = (req, res, next) => {
14   Product.find()
15     .then(products => {
16       console.log(products);
17       res.render('shop/product-list', {
18         prods: products,
19         pageTitle: 'All Products',
20         path: '/products'
21       });
22     })
23     .catch(err => {
24       const error = new Error(err);
25       error.httpStatusCode = 500;
26       return next(error);
27     });
28 };
29
30 exports.getProduct = (req, res, next) => {
31   const prodId = req.params.productId;
32   Product.findById(prodId)
33     .then(product => {
34       res.render('shop/product-detail', {
35         product: product,
```

```

36     pageTitle: product.title,
37     path: '/products'
38   });
39 })
40 .catch(err => {
41   const error = new Error(err);
42   error.httpStatusCode = 500;
43   return next(error);
44 });
45 };
46
47 exports.getIndex = (req, res, next) => {
48   const page = req.query.page;
49   let totalItems;
50
51   Product.find()
52     .countDocuments()
53     .then(numProducts => {
54       totalItems = numProducts;
55       return Product.find()
56         .skip((page - 1) * ITEMS_PER_PAGE)
57         .limit(ITEMS_PER_PAGE);
58     })
59     .then(products => {
60       res.render('shop/index', {
61         prods: products,
62         pageTitle: 'Shop',
63         path: '/',
64         totalProducts: totalItems,
65         hasNextPage: ITEMS_PER_PAGE * page < totalItems,
66         hasPreviousPage: page > 1,
67         nextPage: page + 1,
68         previousPage: page - 1,
69         lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
70       });
71     })
72     .catch(err => {
73       const error = new Error(err);
74       error.httpStatusCode = 500;
75       return next(error);
76     });
77 };
78
79 exports.getCart = (req, res, next) => {
80   req.user
81     .populate('cart.items.productId')
82     .execPopulate()
83     .then(user => {
84       const products = user.cart.items;
85       res.render('shop/cart', {
86         path: '/cart',
87         pageTitle: 'Your Cart',
88         products: products
89       });
90     })
91     .catch(err => {

```



```

92     const error = new Error(err);
93     error.httpStatusCode = 500;
94     return next(error);
95 });
96 };
97
98 exports.postCart = (req, res, next) => {
99     const prodId = req.body.productId;
100    Product.findById(prodId)
101        .then(product => {
102            return req.user.addToCart(product);
103        })
104        .then(result => {
105            console.log(result);
106            res.redirect('/cart');
107        })
108        .catch(err => {
109            const error = new Error(err);
110            error.httpStatusCode = 500;
111            return next(error);
112        });
113 };
114
115 exports.postCartDeleteProduct = (req, res, next) => {
116     const prodId = req.body.productId;
117     req.user
118         .removeFromCart(prodId)
119         .then(result => {
120             res.redirect('/cart');
121         })
122         .catch(err => {
123             const error = new Error(err);
124             error.httpStatusCode = 500;
125             return next(error);
126         });
127 };
128
129 exports.postOrder = (req, res, next) => {
130     req.user
131         .populate('cart.items.productId')
132         .execPopulate()
133         .then(user => {
134             const products = user.cart.items.map(i => {
135                 return { quantity: i.quantity, product: { ...i.productId._doc } };
136             });
137             const order = new Order({
138                 user: {
139                     email: req.user.email,
140                     userId: req.user
141                 },
142                 products: products
143             });
144             return order.save();
145         })
146         .then(result => {
147             return req.user.clearCart();

```

```

148     })
149     .then(() => {
150         res.redirect('/orders');
151     })
152     .catch(err => {
153         const error = new Error(err);
154         error.httpStatusCode = 500;
155         return next(error);
156     });
157 };
158
159 exports.getOrders = (req, res, next) => {
160     Order.find({ 'user.userId': req.user._id })
161     .then(orders => {
162         res.render('shop/orders', {
163             path: '/orders',
164             pageTitle: 'Your Orders',
165             orders: orders
166         });
167     })
168     .catch(err => {
169         const error = new Error(err);
170         error.httpStatusCode = 500;
171         return next(error);
172     });
173 };
174
175 exports.getInvoice = (req, res, next) => {
176     const orderId = req.params.orderId;
177     Order.findById(orderId)
178     .then(order => {
179         if (!order) {
180             return next(new Error('No order found.'));
181         }
182         if (order.user.userId.toString() !== req.user._id.toString()) {
183             return next(new Error('Unauthorized'));
184         }
185         const invoiceName = 'invoice-' + orderId + '.pdf';
186         const invoicePath = path.join('data', 'invoices', invoiceName);
187
188         const pdfDoc = new PDFDocument();
189         res.setHeader('Content-Type', 'application/pdf');
190         res.setHeader(
191             'Content-Disposition',
192             'inline; filename="' + invoiceName + '"'
193         );
194         pdfDoc.pipe(fs.createWriteStream(invoicePath));
195         pdfDoc.pipe(res);
196
197         pdfDoc.fontSize(26).text('Invoice', {
198             underline: true
199         });
200         pdfDoc.text('-----');
201         let totalPrice = 0;
202         order.products.forEach(prod => {
203             totalPrice += prod.quantity * prod.product.price;

```

```

204     pdfDoc
205       .fontSize(14)
206       .text(
207         prod.product.title +
208         ' - ' +
209         prod.quantity +
210         ' x ' +
211         '$' +
212         prod.product.price
213       );
214   });
215   pdfDoc.text('---');
216   pdfDoc.fontSize(20).text('Total Price: $' + totalPrice);
217
218   pdfDoc.end();
219   // fs.readFile(invoicePath, (err, data) => {
220   //   if (err) {
221   //     return next(err);
222   //   }
223   //   res.setHeader('Content-Type', 'application/pdf');
224   //   res.setHeader(
225   //     'Content-Disposition',
226   //     'inline; filename="' + invoiceName + '"'
227   //   );
228   //   res.send(data);
229   // });
230   // const file = fs.createReadStream(invoicePath);
231
232   // file.pipe(res);
233 })
234 .catch(err => next(err));
235 };
236

```

```

1 <!--./views/shop/index.ejs-->
2
3 <%= include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/product.css">
5 </head>
6
7 <body>
8   <%= include('../includes/navigation.ejs') %>
9
10  <main>
11    <% if (prods.length > 0) { %>
12      <div class="grid">
13        <% for (let product of prods) { %>
14          <article class="card product-item">
15            <header class="card__header">
16              <h1 class="product__title"><%= product.title %></h1>
17            </header>
18            <div class="card__image">
19              "
21            </div>
22            <div class="card__content">
23              <h2 class="product__price"><%= product.price %></h2>

```

```

24         <p class="product_description"><%= product.description %></p>
25     </div>
26     <div class="card_actions">
27         <a href="/products/<%= product._id %>" class="btn">Details</a>
28         <% if (isAuthenticated) { %>
29             <%= include('../includes/add-to-cart.ejs', {product:
product}) %>
30         <% } %>
31     </div>
32 </article>
33 <% } %>
34 </div>
35 <section class="pagination">
36     <% if (currentPage !== 1 && previousPage !== 1) { %>
37         <a href="/?page=1">1</a>
38     <% } %>
39     <% if (hasPreviousPage) { %>
40         <a href="/?page=<%= previousPage %>"><%= previousPage %></a>
41     <% } %>
42     <a href="/?page=<%= currentPage %>" class="active"><%= currentPage %>
</a>
43     <% if (hasNextPage) { %>
44         <a href="/?page=<%= nextPage %>"><%= nextPage %></a>
45     <% } %>
46     <% if (lastPage !== currentPage && nextPage !== lastPage) { %>
47         <a href="/?page=<%= lastPage %>"><%= lastPage %></a>
48     <% } %>
49 </section>
50 <% } else { %>
51     <h1>No Products Found!</h1>
52 <% } %>
53 </main>
54 <%= include('../includes/end.ejs') %>

```

```

1 /*./public/css/main.css*/
2
3 @import url('https://fonts.googleapis.com/css?family=Open+Sans:400,700');
4
5 * {
6     box-sizing: border-box;
7 }
8
9 body {
10     padding: 0;
11     margin: 0;
12     font-family: 'Open Sans', sans-serif;
13 }
14
15 main {
16     padding: 1rem;
17     margin: auto;
18 }
19
20 form {
21     display: inline;
22 }
23

```

```
24 .centered {
25   text-align: center;
26 }
27
28 .image {
29   height: 20rem;
30 }
31
32 .image img {
33   height: 100%;
34 }
35
36 .main-header {
37   width: 100%;
38   height: 3.5rem;
39   background-color: #00695c;
40   padding: 0 1.5rem;
41   display: flex;
42   align-items: center;
43 }
44
45 .main-header__nav {
46   height: 100%;
47   width: 100%;
48   display: none;
49   align-items: center;
50   justify-content: space-between;
51 }
52
53 .main-header__item-list {
54   list-style: none;
55   margin: 0;
56   padding: 0;
57   display: flex;
58 }
59
60 .main-header__item {
61   margin: 0 1rem;
62   padding: 0;
63 }
64
65 .main-header__item a,
66 .main-header__item button {
67   font: inherit;
68   background: transparent;
69   border: none;
70   text-decoration: none;
71   color: white;
72   cursor: pointer;
73 }
74
75 .main-header__item a:hover,
76 .main-header__item a:active,
77 .main-header__item a.active,
78 .main-header__item button:hover,
79 .main-header__item button:active {
```

```
80   color: #ffeb3b;
81 }
82
83 .mobile-nav {
84   width: 30rem;
85   height: 100vh;
86   max-width: 90%;
87   position: fixed;
88   left: 0;
89   top: 0;
90   background: white;
91   z-index: 10;
92   padding: 2rem 1rem 1rem 2rem;
93   transform: translateX(-100%);
94   transition: transform 0.3s ease-out;
95 }
96
97 .mobile-nav.open {
98   transform: translateX(0);
99 }
100
101 .mobile-nav__item-list {
102   list-style: none;
103   display: flex;
104   flex-direction: column;
105   margin: 0;
106   padding: 0;
107 }
108
109 .mobile-nav__item {
110   margin: 1rem;
111   padding: 0;
112 }
113
114 .mobile-nav__item a,
115 .mobile-nav__item button {
116   font: inherit;
117   text-decoration: none;
118   color: black;
119   font-size: 1.5rem;
120   padding: 0.5rem 2rem;
121   background: transparent;
122   border: none;
123   cursor: pointer;
124 }
125
126 .mobile-nav__item a:active,
127 .mobile-nav__item a:hover,
128 .mobile-nav__item a.active,
129 .mobile-nav__item button:hover,
130 .mobile-nav__item button:active {
131   background: #00695c;
132   color: white;
133   border-radius: 3px;
134 }
135
```

```
136 #side-menu-toggle {
137   border: 1px solid white;
138   font: inherit;
139   padding: 0.5rem;
140   display: block;
141   background: transparent;
142   color: white;
143   cursor: pointer;
144 }
145
146 #side-menu-toggle:focus {
147   outline: none;
148 }
149
150 #side-menu-toggle:active,
151 #side-menu-toggle:hover {
152   color: #ffeb3b;
153   border-color: #ffeb3b;
154 }
155
156 .backdrop {
157   position: fixed;
158   top: 0;
159   left: 0;
160   width: 100%;
161   height: 100vh;
162   background: rgba(0, 0, 0, 0.5);
163   z-index: 5;
164   display: none;
165 }
166
167 .grid {
168   display: flex;
169   flex-wrap: wrap;
170   justify-content: space-around;
171   align-items: stretch;
172 }
173
174 .card {
175   box-shadow: 0 2px 8px rgba(0, 0, 0, 0.26);
176 }
177
178 .card__header,
179 .card__content {
180   padding: 1rem;
181 }
182
183 .card__header h1,
184 .card__content h1,
185 .card__content h2,
186 .card__content p {
187   margin: 0;
188 }
189
190 .card__image {
191   width: 100%;
```

```
192 }
193
194 .card__image img {
195     width: 100%;
196 }
197
198 .card__actions {
199     padding: 1rem;
200     text-align: center;
201 }
202
203 .card__actions button,
204 .card__actions a {
205     margin: 0 0.25rem;
206 }
207
208 .btn {
209     display: inline-block;
210     padding: 0.25rem 1rem;
211     text-decoration: none;
212     font: inherit;
213     border: 1px solid #00695c;
214     color: #00695c;
215     background: white;
216     border-radius: 3px;
217     cursor: pointer;
218 }
219
220 .btn:hover,
221 .btn:active {
222     background-color: #00695c;
223     color: white;
224 }
225
226 .btn.danger {
227     color: red;
228     border-color: red;
229 }
230
231 .btn.danger:hover,
232 .btn.danger:active {
233     background: red;
234     color: white;
235 }
236
237 .user-message {
238     margin: auto;
239     width: 90%;
240     border: 1px solid #4771fa;
241     padding: 0.5rem;
242     border-radius: 3px;
243     background: #b9c9ff;
244     text-align: center;
245 }
246
247 .user-message--error {
```



```

248 border-color: red;
249 background: rgb(255, 176, 176);
250 color: red;
251 }
252
253 .pagination {
254     margin-top: 2rem;
255     text-align: center
256 }
257
258 .pagination a {
259     text-decoration: none;
260     color: #00695c;
261     padding: 0.5rem;
262     border: 1px solid #00695c;
263     margin: 0 1rem;
264 }
265
266 .pagination a:hover,
267 .pagination a:active,
268 .pagination a.active {
269     background: #00695c;
270     color: white;
271 }
272
273 @media (min-width: 768px) {
274     .main-header__nav {
275         display: flex;
276     }
277
278     #side-menu-toggle {
279         display: none;
280     }
281
282     .user-message {
283         width: 30rem;
284     }
285 }
286

```

* Chapter 339: Re-Using The Pagination Logic & Controls

1. update

- ./views/shop/index.ejs
- ./views/includes/pagination.ejs
- ./views/shop/product-list.ejs
- ./controllers/shop.js

Product #1



\$ 11.99

The first product

Details

Add to Cart

- 1
- 2
- 4

Product #2



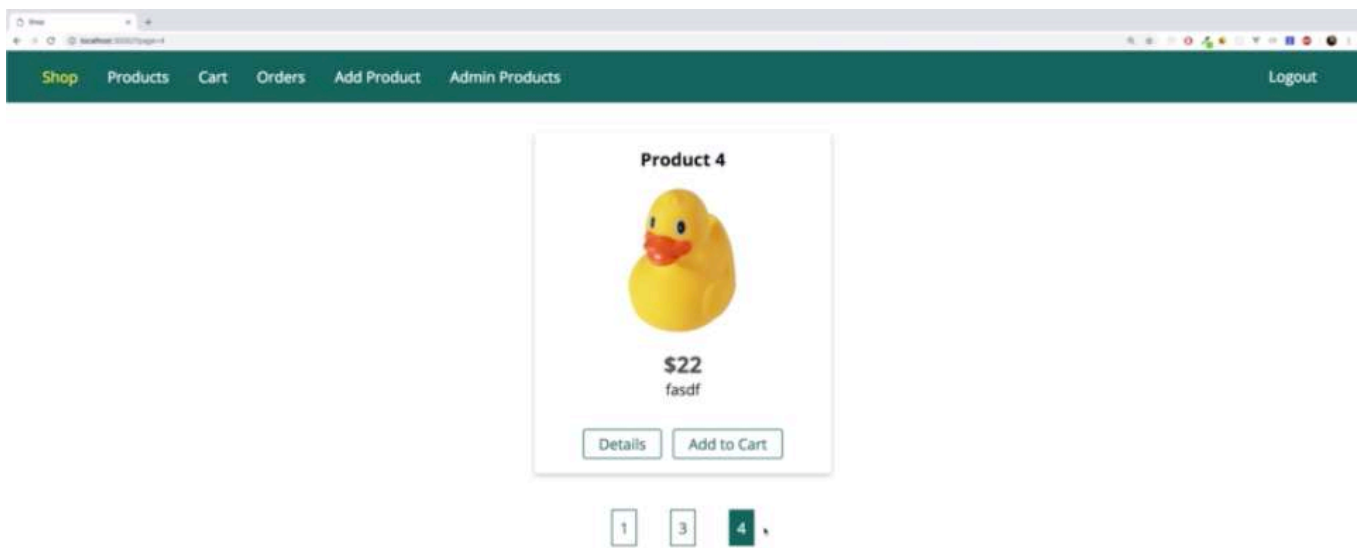
\$22.99

Pocket deuces

Details

Add to Cart

- 1
- 2
- 3
- 4



Shreyas

```
1 //./controllers/shop.js
2
3 const fs = require('fs');
4 const path = require('path');
5
6 const PDFDocument = require('pdfkit');
7
8 const Product = require('../models/product');
9 const Order = require('../models/order');
10
11 const ITEMS_PER_PAGE = 2;
12
13 exports.getProducts = (req, res, next) => {
14   const page = +req.query.page || 1;
15   let totalItems;
16
17   Product.find()
18     .countDocuments()
19     .then(numProducts => {
20       totalItems = numProducts;
21       return Product.find()
22         .skip((page - 1) * ITEMS_PER_PAGE)
23         .limit(ITEMS_PER_PAGE);
24     })
25     .then(products => {
26       res.render('shop/product-list', {
27         prods: products,
28         pageTitle: 'Products',
29         path: '/products',
30         currentPage: page,
31         hasNextPage: ITEMS_PER_PAGE * page < totalItems,
32         hasPreviousPage: page > 1,
33         nextPage: page + 1,
34         previousPage: page - 1,
35         lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
```

```

36     });
37   })
38   .catch(err => {
39     const error = new Error(err);
40     error.httpStatusCode = 500;
41     return next(error);
42   });
43 };
44
45 exports.getProduct = (req, res, next) => {
46   const prodId = req.params.productId;
47   Product.findById(prodId)
48     .then(product => {
49       res.render('shop/product-detail', {
50         product: product,
51         pageTitle: product.title,
52         path: '/products'
53       });
54     })
55     .catch(err => {
56       const error = new Error(err);
57       error.httpStatusCode = 500;
58       return next(error);
59     });
60 };
61
62 exports.getIndex = (req, res, next) => {
63   const page = +req.query.page || 1;
64   let totalItems;
65
66   Product.find()
67     .countDocuments()
68     .then(numProducts => {
69       totalItems = numProducts;
70       return Product.find()
71         .skip((page - 1) * ITEMS_PER_PAGE)
72         .limit(ITEMS_PER_PAGE);
73     })
74     .then(products => {
75       res.render('shop/index', {
76         prods: products,
77         pageTitle: 'Shop',
78         path: '/',
79         currentPage: page,
80         hasNextPage: ITEMS_PER_PAGE * page < totalItems,
81         hasPreviousPage: page > 1,
82         nextPage: page + 1,
83         previousPage: page - 1,
84         lastPage: Math.ceil(totalItems / ITEMS_PER_PAGE)
85       });
86     })
87     .catch(err => {
88       const error = new Error(err);
89       error.httpStatusCode = 500;
90       return next(error);
91     });

```

```

92 };
93
94 exports.getCart = (req, res, next) => {
95   req.user
96     .populate('cart.items.productId')
97     .execPopulate()
98     .then(user => {
99       const products = user.cart.items;
100       res.render('shop/cart', {
101         path: '/cart',
102         pageTitle: 'Your Cart',
103         products: products
104       });
105     })
106     .catch(err => {
107       const error = new Error(err);
108       error.httpStatusCode = 500;
109       return next(error);
110     });
111 };
112
113 exports.postCart = (req, res, next) => {
114   const prodId = req.body.productId;
115   Product.findById(prodId)
116     .then(product => {
117       return req.user.addToCart(product);
118     })
119     .then(result => {
120       console.log(result);
121       res.redirect('/cart');
122     })
123     .catch(err => {
124       const error = new Error(err);
125       error.httpStatusCode = 500;
126       return next(error);
127     });
128 };
129
130 exports.postCartDeleteProduct = (req, res, next) => {
131   const prodId = req.body.productId;
132   req.user
133     .removeFromCart(prodId)
134     .then(result => {
135       res.redirect('/cart');
136     })
137     .catch(err => {
138       const error = new Error(err);
139       error.httpStatusCode = 500;
140       return next(error);
141     });
142 };
143
144 exports.postOrder = (req, res, next) => {
145   req.user
146     .populate('cart.items.productId')
147     .execPopulate()

```

```

148 .then(user => {
149     const products = user.cart.items.map(i => {
150         return { quantity: i.quantity, product: { ...i.productId._doc } };
151     });
152     const order = new Order({
153         user: {
154             email: req.user.email,
155             userId: req.user
156         },
157         products: products
158     });
159     return order.save();
160 })
161 .then(result => {
162     return req.user.clearCart();
163 })
164 .then(() => {
165     res.redirect('/orders');
166 })
167 .catch(err => {
168     const error = new Error(err);
169     error.httpStatusCode = 500;
170     return next(error);
171 });
172 };
173
174 exports.getOrders = (req, res, next) => {
175     Order.find({ 'user.userId': req.user._id })
176     .then(orders => {
177         res.render('shop/orders', {
178             path: '/orders',
179             pageTitle: 'Your Orders',
180             orders: orders
181         });
182     })
183     .catch(err => {
184         const error = new Error(err);
185         error.httpStatusCode = 500;
186         return next(error);
187     });
188 };
189
190 exports.getInvoice = (req, res, next) => {
191     const orderId = req.params.orderId;
192     Order.findById(orderId)
193     .then(order => {
194         if (!order) {
195             return next(new Error('No order found.'));
196         }
197         if (order.user.userId.toString() !== req.user._id.toString()) {
198             return next(new Error('Unauthorized'));
199         }
200         const invoiceName = 'invoice-' + orderId + '.pdf';
201         const invoicePath = path.join('data', 'invoices', invoiceName);
202
203         const pdfDoc = new PDFDocument();

```

```

204 res.setHeader('Content-Type', 'application/pdf');
205 res.setHeader(
206   'Content-Disposition',
207   'inline; filename="' + invoiceName + '"'
208 );
209 pdfDoc.pipe(fs.createWriteStream(invoicePath));
210 pdfDoc.pipe(res);
211
212 pdfDoc.fontSize(26).text('Invoice', {
213   underline: true
214 });
215 pdfDoc.text('-----');
216 let totalPrice = 0;
217 order.products.forEach(prod => {
218   totalPrice += prod.quantity * prod.product.price;
219   pdfDoc
220     .fontSize(14)
221     .text(
222       prod.product.title +
223       ' - ' +
224       prod.quantity +
225       ' x ' +
226       '$' +
227       prod.product.price
228     );
229 });
230 pdfDoc.text('---');
231 pdfDoc.fontSize(20).text('Total Price: $' + totalPrice);
232
233 pdfDoc.end();
234 // fs.readFile(invoicePath, (err, data) => {
235 //   if (err) {
236 //     return next(err);
237 //   }
238 //   res.setHeader('Content-Type', 'application/pdf');
239 //   res.setHeader(
240 //     'Content-Disposition',
241 //     'inline; filename="' + invoiceName + '"'
242 //   );
243 //   res.send(data);
244 // });
245 // const file = fs.createReadStream(invoicePath);
246
247 // file.pipe(res);
248 })
249 .catch(err => next(err));
250 };
251

```

```

1 <!--./views/shop/index.ejs-->
2
3 <%- include('../includes/head.ejs') %>
4   <link rel="stylesheet" href="/css/product.css">
5 </head>
6
7 <body>
8   <%- include('../includes/navigation.ejs') %>

```

```

9
10 <main>
11   <% if (prods.length > 0) { %>
12     <div class="grid">
13       <% for (let product of prods) { %>
14         <article class="card product-item">
15           <header class="card_header">
16             <h1 class="product_title"><%= product.title %></h1>
17           </header>
18           <div class="card_image">
19             "
21             </div>
22           <div class="card_content">
23             <h2 class="product_price">$<%= product.price %></h2>
24             <p class="product_description"><%= product.description %></p>
25           </div>
26           <div class="card_actions">
27             <a href="/products/<%= product._id %>" class="btn">Details</a>
28             <% if (isAuthenticated) { %>
29               <%= include('../includes/add-to-cart.ejs', {product:
product}) %>
30             <% } %>
31           </div>
32         </article>
33       <% } %>
34     </div>
35     <%= include('../includes/pagination.ejs', {currentPage: currentPage, nextPage:
nextPage, previousPage: previousPage, lastPage: lastPage, hasNextPage: hasNextPage,
hasPreviousPage: hasPreviousPage}) %>
36   <% } else { %>
37     <h1>No Products Found!</h1>
38   <% } %>
39 </main>
40 <%= include('../includes/end.ejs') %>

```

```

1 <!--./views/includes/pagination.ejs-->
2
3 <section class="pagination">
4   <% if (currentPage !== 1 && previousPage !== 1) { %>
5     <a href="?page=1">1</a>
6   <% } %>
7   <% if (hasPreviousPage) { %>
8     <a href="?page=<%= previousPage %>"><%= previousPage %></a>
9   <% } %>
10  <a href="?page=<%= currentPage %>" class="active"><%= currentPage %></a>
11  <% if (hasNextPage) { %>
12    <a href="?page=<%= nextPage %>"><%= nextPage %></a>
13  <% } %>
14  <% if (lastPage !== currentPage && nextPage !== lastPage) { %>
15    <a href="?page=<%= lastPage %>"><%= lastPage %></a>
16  <% } %>
17 </section>

```

```

1 <!--./views/shop/product-list.ejs-->
2
3 <%= include('../includes/head.ejs') %>

```



```

4 <link rel="stylesheet" href="/css/product.css">
5 </head>
6
7 <body>
8 <%= include('../includes/navigation.ejs') %>
9
10 <main>
11 <% if (prods.length > 0) { %>
12 <div class="grid">
13 <% for (let product of prods) { %>
14 <article class="card product-item">
15 <header class="card__header">
16 <h1 class="product__title">
17 <%= product.title %>
18 </h1>
19 </header>
20 <div class="card__image">
21 ">
22 </div>
23 <div class="card__content">
24 <h2 class="product__price">$
25 <%= product.price %>
26 </h2>
27 <p class="product__description">
28 <%= product.description %>
29 </p>
30 </div>
31 <div class="card__actions">
32 <a href="/products/<%= product.id %>"
class="btn">Details</a>
33 <% if (isAuthenticated) { %>
34 <%= include('../includes/add-to-cart.ejs', {product:
product}) %>
35 <% } %>
36 </div>
37 </article>
38 <% } %>
39 </div>
40 <%= include('../includes/pagination.ejs', {currentPage: currentPage,
nextPage: nextPage, previousPage: previousPage, lastPage: lastPage, hasNextPage:
hasNextPage, hasPreviousPage: hasPreviousPage}) %>
41 <% } else { %>
42 <h1>No Products Found!</h1>
43 <% } %>
44 </main>
45 <%= include('../includes/end.ejs') %>

```