

# Lab: E-Commerce Data Rendering using React Toolkit

**Estimated time needed:** 60 minutes

## Introduction

In this lab, you will learn how to use Redux Toolkit to manage state throughout your entire application in such a way that it can be accessed by any component without following the hierarchy between the components. You will build a simple E-Commerce application using React and Redux where you will display a list of products with an "Add to Cart" button for each product, display the items added to the cart, and allow users to remove items from the cart. All this information is going to be globally available throughout the application using Redux Toolkit.

## Learning objectives

After completing this lab, you will be able to:

- Integrate React components with Redux for state management
- Implement basic E-Commerce features such as adding items to the cart, removing items from the cart, and clearing the cart
- Practice composing multiple React components to build a cohesive user interface

## Prerequisites

- Basic knowledge of React composition of components
- Intermediate knowledge of JavaScript
- Knowledge of React functional component, state management using useState hook, and Redux Toolkit

## Step 1: Setting up the environment

1. From the menu on top of the lab, click the **Terminal** tab at the top-right of the window shown at number 1 in the given screenshot, and then click **New Terminal** as shown at number 2.

2. Now, write the following command in the terminal to clone the boiler template for this React application and hit Enter.

```
1. 1
1. git clone https://github.com/ibm-developer-skills-network/e-commerce_rtk.git
```

Copied!

3. The above command will create a folder, "e-commerce\_rtk" under the "Project" folder. You can see the structure in the screenshot.

4. Next, you need to make sure that the path of your terminal should have cloned folder to perform certain operations for this react application. Use the below command to navigate to the **e-commerce\_rtk** folder in the terminal.

```
1. 1
1. cd e-commerce_rtk
```

Copied!

5. To ensure the code you have cloned is working correctly, you need to perform the following steps:

- Write the given command in the terminal and hit Enter. This command will install all the necessary packages to execute the application.

```
1. 1
1. npm install
```

Copied!

- Then execute the following command to run the application and this will provide you with port number 4173.

```
1. 1
1. npm run preview
```

Copied!

6. To view your React application, click the Skills Network icon on the left (refer to number 1). This action will open the **SKILLS NETWORK TOOLBOX**. Next, click **Launch Application** (refer to number 2). Enter the port number **4173** in **Application Port** (refer to number 3) and click .

7. The output will display as shown in the given screenshot.

8. You can preserve your latest work on this lab by adding, committing, and pushing it to your GitHub repository. This ensures that even if you're not working on the task continuously, your progress will be saved, allowing you to resume from where you left off.

*Note: Step 8 is optional. We will recommend you to complete task in one sitting*

## Step 2: Implementing ProductList component

1. Next, navigate to the **ProductList.jsx** file located in the **Components** folder of the **src** directory in your cloned **e-commerce\_rtk** folder.
2. The basic structure of this component will be as shown in the screenshot.
3. Now you need to show the product list in the front end. For this you need to apply the map method within the `<ul>` tag with class name "product-list-items".

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8

1. {products.map(product => (
2.   <li key={product.id} className="product-list-item">
3.     <span>{product.name} - ${product.price}</span>
4.     <button>
5.       Add to Cart
6.     </button>
7.   </li>
8. ))}
```

**Copied!**

- Check the output and it will be displayed as shown in the given screenshot.

## Step 3: Implement Redux logic

- You need to apply logic for Redux toolkit to ensure that when you click the Add Product button to add a product to the cart, the information of product quantity entered by you should be available globally to any component.
- Now navigate to the **CartSlice.jsx** file located in the **Components** folder of the **src** directory in your cloned **ecommerce\_rtk** folder.
- You will see the structure as given below:

```

1. 1
2. 2
3. 3

1. const CartSlice = ({
```

**Copied!**

- First, initialize an empty array named **cartItems** outside **CartSlice**.

```

1. 1
2. 2
3. 3

1. const initialState = {
2.   cartItems: [],
3. };
```

**Copied!**

- Now initialize **CartSlice** with one **createSlice** Redux Toolkit function. You need to install **@reduxjs/toolkit** and **react-redux** as a third-party module. For this lab, you do not need to install it separately as it is already provided in the **package.json** file and **createSlice** is a utility function provided by Redux Toolkit, a library built on top of Redux. It simplifies the process of creating Redux slices, which are portions of the Redux state, along with associated action creators and reducers.

```

1. 1
2. 2
3. 3

1. const CartSlice = createSlice({
```

**Copied!**

- Ensure that **createSlice** should be imported at the top of this component.

```

1. 1
1. import { createSlice } from '@reduxjs/toolkit';
```

**Copied!**

## Step 4: Actions and reducers creation

- Slice Creation:
  - You call **createSlice** with an object containing configuration options for your slice.
  - The configuration options include:
    - name**: A string value representing the name of your slice. It's used internally by Redux Toolkit for action type prefixing and other purposes.
    - initialState**: An object representing the initial state of your slice.
    - reducers**: An object containing reducer functions. Each key-value pair represents a single reducer, where the key is the name of the action and the value is the reducer function.

```

1. 1
2. 2
3. 3
4. 4
```

```

5. 5
6. 6
1. const CartSlice = createSlice({
2.   name: 'cart',
3.   initialState,
4.   reducers: {
5.     }
6. });

```

**Copied!**

## Step 5: Reducers creation and export actions

- Inside the **reducer** object, you need to create five functions out of which two are used to handle addition and removal of products in the shopping cart, one to clear all the items at once, and other two are to increase and decrease the quantity.

- o **addItemToCart**:

- This reducer function handles the action of adding an item to the cart.
- It takes two parameters: **state** (current state of the slice) and **action** (the dispatched action containing the payload).
- It first checks if the item already exists in the cart by searching for its ID within **state.cartItems**.
- If the item exists (**existingItem** is true), it increases the quantity of the existing item in the cart by 1.
- If the item doesn't exist in the cart, it adds the item to the **cartItems** array with a quantity of 1.

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
1. addItemToCart(state, action) {
2.   const existingItem = state.cartItems.find(item => item.id === action.payload.id);
3.   if (existingItem) {
4.     existingItem.quantity += 1;
5.   } else {
6.     state.cartItems.push({ ...action.payload, quantity: 1 });
7.   }
8. },

```

**Copied!**

- o **removeItemFromCart**:

- This reducer function handles the action of removing an item from the cart.
- It takes two parameters: **state** and **action**.
- It updates the **cartItems** array by filtering out the item with the ID provided in the **action payload**.

```

1. 1
2. 2
3. 3
1. removeItemFromCart(state, action) {
2.   state.cartItems = state.cartItems.filter(item => item.id !== action.payload);
3. },

```

**Copied!**

- o **clearCart**:

- This reducer function handles the action of clearing the entire cart.
- It takes only the state parameter.
- It sets the **cartItems** array to an empty array, effectively clearing all items from the cart.

```

1. 1
2. 2
3. 3
1. clearCart(state) {
2.   state.cartItems = [];
3. },

```

**Copied!**

- o **increaseItemQuantity**:

- This reducer function handles the action of increasing the quantity of a specific item in the cart.
- It takes two parameters: **state** and **action**.
  - **state**: This represents the current state of the Redux store. It typically includes the data relevant to the application.
  - **action**: This is an object that describes the action that occurred. Redux actions are plain JavaScript objects that must have a type property indicating the type of action being performed. Additionally, they may contain additional data necessary to carry out the action. In this case, **action.payload** likely contains the identifier (id) of the item whose quantity needs to be increased.

- The function logic:

- It finds the item in the shopping cart whose identifier (id) matches the identifier passed in the action payload.
- If the item is found (itemToIncrease is not undefined), it increments the quantity property of that item by 1.

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
1. increaseItemQuantity(state, action) {
2.   const itemToIncrease = state.cartItems.find(item => item.id === action.payload);
3.   if (itemToIncrease) {
4.     itemToIncrease.quantity += 1;
5.   }
6. },

```

**Copied!**

- o **decreaseItemQuantity:**

- This reducer function handles the action of decreasing the quantity of a specific item in the cart.
- It takes two parameters: **state** and **action**.
  - **state:** This represents the current state of the Redux store, typically containing all the data relevant to the application.
  - **action:** Similar to the previous function, it's an object describing the action being performed. It's expected to have a type property indicating the action type and may include additional data needed to carry out the action. Here, **action.payload** likely holds the identifier (id) of the item whose quantity needs to be decreased.
- The function logic:
  - It attempts to find the item in the shopping cart whose identifier (id) matches the identifier provided in the action payload.
  - If the item is found (**itemToDecrease** is not undefined) and its quantity is greater than 1, it decrements the quantity property of that item by 1.

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46
47. 47
48. 48
49. 49
50. 50
51. 51
52. 52
53. 53
54. 54
55. 55
56. 56
57. 57
58. 58
59. 59
60. 60
61. 61
62. 62
63. 63
64. 64
65. 65
66. 66
67. 67
68. 68
69. 69
70. 70
71. 71
72. 72
73. 73
74. 74
75. 75
76. 76
77. 77
78. 78
79. 79
80. 80
81. 81
82. 82
83. 83
84. 84
85. 85
86. 86
87. 87
88. 88
89. 89
90. 90
91. 91
92. 92
93. 93
94. 94
95. 95
96. 96
97. 97
98. 98
99. 99
100. 100
101. 101
102. 102
103. 103
104. 104
105. 105
106. 106
107. 107
108. 108
109. 109
110. 110
111. 111
112. 112
113. 113
114. 114
115. 115
116. 116
117. 117
118. 118
119. 119
120. 120
121. 121
122. 122
123. 123
124. 124
125. 125
126. 126
127. 127
128. 128
129. 129
130. 130
131. 131
132. 132
133. 133
134. 134
135. 135
136. 136
137. 137
138. 138
139. 139
140. 140
141. 141
142. 142
143. 143
144. 144
145. 145
146. 146
147. 147
148. 148
149. 149
150. 150
151. 151
152. 152
153. 153
154. 154
155. 155
156. 156
157. 157
158. 158
159. 159
160. 160
161. 161
162. 162
163. 163
164. 164
165. 165
166. 166
167. 167
168. 168
169. 169
170. 170
171. 171
172. 172
173. 173
174. 174
175. 175
176. 176
177. 177
178. 178
179. 179
180. 180
181. 181
182. 182
183. 183
184. 184
185. 185
186. 186
187. 187
188. 188
189. 189
190. 190
191. 191
192. 192
193. 193
194. 194
195. 195
196. 196
197. 197
198. 198
199. 199
200. 200
201. 201
202. 202
203. 203
204. 204
205. 205
206. 206
207. 207
208. 208
209. 209
210. 210
211. 211
212. 212
213. 213
214. 214
215. 215
216. 216
217. 217
218. 218
219. 219
220. 220
221. 221
222. 222
223. 223
224. 224
225. 225
226. 226
227. 227
228. 228
229. 229
230. 230
231. 231
232. 232
233. 233
234. 234
235. 235
236. 236
237. 237
238. 238
239. 239
240. 240
241. 241
242. 242
243. 243
244. 244
245. 245
246. 246
247. 247
248. 248
249. 249
250. 250
251. 251
252. 252
253. 253
254. 254
255. 255
256. 256
257. 257
258. 258
259. 259
260. 260
261. 261
262. 262
263. 263
264. 264
265. 265
266. 266
267. 267
268. 268
269. 269
270. 270
271. 271
272. 272
273. 273
274. 274
275. 275
276. 276
277. 277
278. 278
279. 279
280. 280
281. 281
282. 282
283. 283
284. 284
285. 285
286. 286
287. 287
288. 288
289. 289
290. 290
291. 291
292. 292
293. 293
294. 294
295. 295
296. 296
297. 297
298. 298
299. 299
300. 300
301. 301
302. 302
303. 303
304. 304
305. 305
306. 306
307. 307
308. 308
309. 309
310. 310
311. 311
312. 312
313. 313
314. 314
315. 315
316. 316
317. 317
318. 318
319. 319
320. 320
321. 321
322. 322
323. 323
324. 324
325. 325
326. 326
327. 327
328. 328
329. 329
330. 330
331. 331
332. 332
333. 333
334. 334
335. 335
336. 336
337. 337
338. 338
339. 339
340. 340
341. 341
342. 342
343. 343
344. 344
345. 345
346. 346
347. 347
348. 348
349. 349
350. 350
351. 351
352. 352
353. 353
354. 354
355. 355
356. 356
357. 357
358. 358
359. 359
360. 360
361. 361
362. 362
363. 363
364. 364
365. 365
366. 366
367. 367
368. 368
369. 369
370. 370
371. 371
372. 372
373. 373
374. 374
375. 375
376. 376
377. 377
378. 378
379. 379
380. 380
381. 381
382. 382
383. 383
384. 384
385. 385
386. 386
387. 387
388. 388
389. 389
390. 390
391. 391
392. 392
393. 393
394. 394
395. 395
396. 396
397. 397
398. 398
399. 399
400. 400
401. 401
402. 402
403. 403
404. 404
405. 405
406. 406
407. 407
408. 408
409. 409
410. 410
411. 411
412. 412
413. 413
414. 414
415. 415
416. 416
417. 417
418. 418
419. 419
420. 420
421. 421
422. 422
423. 423
424. 424
425. 425
426. 426
427. 427
428. 428
429. 429
430. 430
431. 431
432. 432
433. 433
434. 434
435. 435
436. 436
437. 437
438. 438
439. 439
440. 440
441. 441
442. 442
443. 443
444. 444
445. 445
446. 446
447. 447
448. 448
449. 449
450. 450
451. 451
452. 452
453. 453
454. 454
455. 455
456. 456
457. 457
458. 458
459. 459
460. 460
461. 461
462. 462
463. 463
464. 464
465. 465
466. 466
467. 467
468. 468
469. 469
470. 470
471. 471
472. 472
473. 473
474. 474
475. 475
476. 476
477. 477
478. 478
479. 479
480. 480
481. 481
482. 482
483. 483
484. 484
485. 485
486. 486
487. 487
488. 488
489. 489
490. 490
491. 491
492. 492
493. 493
494. 494
495. 495
496. 496
497. 497
498. 498
499. 499
500. 500
501. 501
502. 502
503. 503
504. 504
505. 505
506. 506
507. 507
508. 508
509. 509
510. 510
511. 511
512. 512
513. 513
514. 514
515. 515
516. 516
517. 517
518. 518
519. 519
520. 520
521. 521
522. 522
523. 523
524. 524
525. 525
526. 526
527. 527
528. 528
529. 529
530. 530
531. 531
532. 532
533. 533
534. 534
535. 535
536. 536
537. 537
538. 538
539. 539
540. 540
541. 541
542. 542
543. 543
544. 544
545. 545
546. 546
547. 547
548. 548
549. 549
550. 550
551. 551
552. 552
553. 553
554. 554
555. 555
556. 556
557. 557
558. 558
559. 559
560. 560
561. 561
562. 562
563. 563
564. 564
565. 565
566. 566
567. 567
568. 568
569. 569
570. 570
571. 571
572. 572
573. 573
574. 574
575. 575
576. 576
577. 577
578. 578
579. 579
580. 580
581. 581
582. 582
583. 583
584. 584
585. 585
586. 586
587. 587
588. 588
589. 589
590. 590
591. 591
592. 592
593. 593
594. 594
595. 595
596. 596
597. 597
598. 598
599. 599
600. 600
601. 601
602. 602
603. 603
604. 604
605. 605
606. 606
607. 607
608. 608
609. 609
610. 610
611. 611
612. 612
613. 613
614. 614
615. 615
616. 616
617. 617
618. 618
619. 619
620. 620
621. 621
622. 622
623. 623
624. 624
625. 625
626. 626
627. 627
628. 628
629. 629
630. 630
631. 631
632. 632
633. 633
634. 634
635. 635
636. 636
637. 637
638. 638
639. 639
640. 640
641. 641
642. 642
643. 643
644. 644
645. 645
646. 646
647. 647
648. 648
649. 649
650. 650
651. 651
652. 652
653. 653
654. 654
655. 655
656. 656
657. 657
658. 658
659. 659
660. 660
661. 661
662. 662
663. 663
664. 664
665. 665
666. 666
667. 667
668. 668
669. 669
670. 670
671. 671
672. 672
673. 673
674. 674
675. 675
676. 676
677. 677
678. 678
679. 679
680. 680
681. 681
682. 682
683. 683
684. 684
685. 685
686. 686
687. 687
688. 688
689. 689
690. 690
691. 691
692. 692
693. 693
694. 694
695. 695
696. 696
697. 697
698. 698
699. 699
700. 700
701. 701
702. 702
703. 703
704. 704
705. 705
706. 706
707. 707
708. 708
709. 709
710. 710
711. 711
712. 712
713. 713
714. 714
715. 715
716. 716
717. 717
718. 718
719. 719
720. 720
721. 721
722. 722
723. 723
724. 724
725. 725
726. 726
727. 727
728. 728
729. 729
730. 730
731. 731
732. 732
733. 733
734. 734
735. 735
736. 736
737. 737
738. 738
739. 739
740. 740
741. 741
742. 742
743. 743
744. 744
745. 745
746. 746
747. 747
748. 748
749. 749
750. 750
751. 751
752. 752
753. 753
754. 754
755. 755
756. 756
757. 757
758. 758
759. 759
760. 760
761. 761
762. 762
763. 763
764. 764
765. 765
766. 766
767. 767
768. 768
769. 769
770. 770
771. 771
772. 772
773. 773
774. 774
775. 775
776. 776
777. 777
778. 778
779. 779
780. 780
781. 781
782. 782
783. 783
784. 784
785. 785
786. 786
787. 787
788. 788
789. 789
790. 790
791. 791
792. 792
793. 793
794. 794
795. 795
796. 796
797. 797
798. 798
799. 799
800. 800
801. 801
802. 802
803. 803
804. 804
805. 805
806. 806
807. 807
808. 808
809. 809
810. 810
811. 811
812. 812
813. 813
814. 814
815. 815
816. 816
817. 817
818. 818
819. 819
820. 820
821. 821
822. 822
823. 823
824. 824
825. 825
826. 826
827. 827
828. 828
829. 829
830. 830
831. 831
832. 832
833. 833
834. 834
835. 835
836. 836
837. 837
838. 838
839. 839
840. 840
841. 841
842. 842
843. 843
844. 844
845. 845
846. 846
847. 847
848. 848
849. 849
850. 850
851. 851
852. 852
853. 853
854. 854
855. 855
856. 856
857. 857
858. 858
859. 859
860. 860
861. 861
862. 862
863. 863
864. 864
865. 865
866. 866
867. 867
868. 868
869. 869
870. 870
871. 871
872. 872
873. 873
874. 874
875. 875
876. 876
877. 877
878. 878
879. 879
880. 880
881. 881
882. 882
883. 883
884. 884
885. 885
886. 886
887. 887
888. 888
889. 889
890. 890
891. 891
892. 892
893. 893
894. 894
895. 895
896. 896
897. 897
898. 898
899. 899
900. 900
901. 901
902. 902
903. 903
904. 904
905. 905
906. 906
907. 907
908. 908
909. 909
910. 910
911. 911
912. 912
913. 913
914. 914
915. 915
916. 916
917. 917
918. 918
919. 919
920. 920
921. 921
922. 922
923. 923
924. 924
925. 925
926. 926
927. 927
928. 928
929. 929
930. 930
931. 931
932. 932
933. 933
934. 934
935. 935
936. 936
937. 937
938. 938
939. 939
940. 940
941. 941
942. 942
943. 943
944. 944
945. 945
946. 946
947. 947
948. 948
949. 949
950. 950
951. 951
952. 952
953. 953
954. 954
955. 955
956. 956
957. 957
958. 958
959. 959
960. 960
961. 961
962. 962
963. 963
964. 964
965. 965
966. 966
967. 967
968. 968
969. 969
970. 970
971. 971
972. 972
973. 973
974. 974
975. 975
976. 976
977. 977
978. 978
979. 979
980. 980
981. 981
982. 982
983. 983
984. 984
985. 985
986. 986
987. 987
988. 988
989. 989
990. 990
991. 991
992. 992
993. 993
994. 994
995. 995
996. 996
997. 997
998. 998
999. 999
1000. 1000
1001. 1001
1002. 1002
1003. 1003
1004. 1004
1005. 1005
1006. 1006
1007. 1007
1008. 1008
1009. 1009
1010. 1010
1011. 1011
1012. 1012
1013. 1013
1014. 1014
1015. 1015
1016. 1016
1017. 1017
1018. 1018
1019. 1019
1020. 1020
1021. 1021
1022. 1022
1023. 1023
1024. 1024
1025. 1025
1026. 1026
1027. 1027
1028. 1028
1029. 1029
1030. 1030
1031. 1031
1032. 1032
1033. 1033
1034. 1034
1035. 1035
1036. 1036
1037. 1037
1038. 1038
1039. 1039
1040. 1040
1041. 1041
1042. 1042
1043. 1043
1044. 1044
1045. 1045
1046. 1046
1047. 1047
1048. 1048
1049. 1049
1050. 1050
1051. 1051
1052. 1052
1053. 1053
1054. 1054
1055. 1055
1056. 1056
1057. 1057
1058. 1058
1059. 1059
1060. 1060
1061. 1061
1062. 1062
1063. 1063
1064. 1064
1065. 1065
1066. 1066
1067. 1067
1068. 1068
1069. 1069
1070. 1070
1071. 1071
1072. 1072
1073. 1073
1074. 1074
1075. 1075
1076. 1076
1077. 1077
1078. 1078
1079. 1079
1080. 1080
1081. 1081
1082. 1082
1083. 1083
1084. 1084
1085. 1085
1086. 1086
1087. 1087
1088. 1088
1089. 1089
1090. 1090
1091. 1091
1092. 1092
1093. 1093
1094. 1094
1095. 1095
1096. 1096
1097. 1097
1098. 1098
1099. 1099
1100. 1100
1101. 1101
1102. 1102
1103. 1103
1104. 1104
1105. 1105
1106. 1106
1107. 1107
1108. 1108
1109. 1109
1110. 1110
1111. 1111
1112. 1112
1113. 1113
1114. 1114
1115. 1115
1116. 1116
1117. 1117
1118. 1118
1119. 1119
1120. 1120
1121. 1121
1122. 1122
1123. 1123
1124. 1124
1125. 1125
1126. 1126
1127. 1127
1128. 1128
1129. 1129
1130. 1130
1131. 1131
1132. 1132
1133. 1133
1134. 1134
1135. 1135
1136. 1136
1137. 1137
1138. 1138
1139. 1139
1140. 1140
1141. 1141
1142. 1142
1143. 1143
1144. 1144
1145. 1145
1146. 1146
1147. 1147
1148. 1148
1149. 1149
1150. 1150
1151. 1151
1152. 1152
1153. 1153
1154. 1154
1155. 1155
1156. 1156
1157. 1157
1158. 1158
1159. 1159
1160. 1160
1161. 1161
1162. 1162
1163. 1163
1164. 1164
1165. 1165
1166. 1166
1167. 1167
1168. 1168
1169. 1169
1170. 1170
1171. 1171
1172. 1172
1173. 1173
1174. 1174
1175. 
```

```

2. import cartReducer from './Components/CartSlice';
3. const store = configureStore({
4.   reducer: {
5.     cart: cartReducer,
6.   },
7. });
8.
9. export default store;

```

**Copied!**

4. Now, to make this data available globally for any component in the application, you need to import the data in **main.jsx** component. For this navigate to **main.jsx** file and paste the below code in the file.

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
1. import React from 'react'
2. import ReactDOM from 'react-dom/client'
3. import App from './App.jsx'
4. import './index.css'
5. import { Provider } from 'react-redux'
6. import store from './store.js'
7. ReactDOM.createRoot(document.getElementById('root')).render(
8.   <React.StrictMode>
9.     <Provider store={store}>
10.       <App />
11.     </Provider>
12.   </React.StrictMode>,
13. )

```

**Copied!**

- In the above code **store.js** file is imported within `<React.StrictMode>`. `<Provider>` from react-redux supplies the Redux store to all components within its hierarchy by passing store as props. This allows components, including `<App />`, to access and interact with the Redux store for state management.

## Step 7: Add product and store data globally

1. In the **ProductList** component, now initialize two variables, one named **disabledProducts** using **useState** hook and the other named **dispatch** which will utilize the functionality of **useDispatch()**.

```

1. 1
2. 2
3. 3
1. const dispatch = useDispatch();
2. const [disabledProducts, setDisabledProducts] = useState([]); // State to store disabled products

```

**Copied!**

- Ensure you have included the given statement at the top of the component.

```

1. 1
2. 2
3. 3
1. import { useDispatch } from 'react-redux';
2. import { useState } from 'react';
3. import { addItemToCart } from './CartSlice';

```

**Copied!**

In the above code, **addItemToCart** is used to get the reducer function detail to dispatch which product is added to the cart to **store.js**.

2. In the **ProductList** component, implement the functionality to add the data to the cart and send the data directly in **store.js** using **useDispatch** hook. In the `<button>` tag call one function **handleAddToCart** for **onClick** event.

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
1. <button
2.   className={`add-to-cart-btn ${disabledProducts.includes(product.id) ? 'disabled' : ''}`}
3.   onClick={() => handleAddToCart(product)}
4.   disabled={disabledProducts.includes(product.id)} // Disable button if product is in disabledProducts >
5.   > Add to Cart
6. </button>

```

**Copied!**

- This button, when clicked, invokes the **handleAddToCart** function with the product as an argument.
  - The button's appearance is dynamically determined by whether the product is included in the **disabledProducts** array, which disables the button if the product is in the array or if the product is added.
  - This functionality prevents adding duplicate items to the cart and provides visual feedback by styling the button as disabled when necessary.

3. Initialize function named as **handleAddToCart**.

```

1. 1
2. 2
3. 3
4. 4

```

```

1. const handleAddToCart = product => {
2.   dispatch(addItemToCart(product));
3.   setDisabledProducts([...disabledProducts, product.id]); // Mark the product as disabled
4. };

```

Copied!

▼ Click here for [ProductList.jsx](#) code

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42

1. import React, { useState } from 'react';
2. import { useDispatch } from 'react-redux';
3. import { addItemToCart } from './CartSlice';
4. import './ProductList.css'; // Import CSS file for component-specific styles
5.
6. const ProductList = () => {
7.   const dispatch = useDispatch();
8.   const [disabledProducts, setDisabledProducts] = useState([]); // State to store disabled products
9.
10.  const products = [
11.    { id: 1, name: 'Product A', price: 60 },
12.    { id: 2, name: 'Product B', price: 75 },
13.    { id: 3, name: 'Product C', price: 30 },
14.  ];
15.
16.  const handleAddToCart = product => {
17.    dispatch(addItemToCart(product));
18.    setDisabledProducts([...disabledProducts, product.id]); // Mark the product as disabled
19.  };
20.
21.  return (
22.    <div className="product-list">
23.      <h2 className="product-list-title">Products</h2>
24.      <ul className="product-list-items">
25.        {products.map(product => (
26.          <li key={product.id} className="product-list-item">
27.            <span>{product.name} - ${product.price}</span>
28.            <button
29.              className={`add-to-cart-btn ${disabledProducts.includes(product.id) ? 'disabled' : ''}`}
30.              onClick={() => handleAddToCart(product)}
31.              disabled={disabledProducts.includes(product.id)} // Disable button if product is in disabledProducts
32.            >
33.              Add to Cart
34.            </button>
35.          </li>
36.        ))}
37.      </ul>
38.    </div>
39.  );
40. };
41.
42. export default ProductList;

```

Copied!

## Step 8: Display products in shopping cart

- Now in **ShoppingCart.jsx** component, you will create logic that shows items that user have added to the shopping cart. It uses a special tool called Redux to manage the cart using redux and keep track of what the user is buying. The component lets the user see the items in the cart, their prices, and how many of

each item the user has added. Users can also remove items from the cart or change the quantity of each item. It's like having a virtual shopping basket that helps the user to keep track of what they will purchase.

2. For this to implement navigate to **ShoppingCart.jsx** component under src folder.

3. The basic structure of this component will be as shown in the screenshot.

4. Implement the given functionalities:-

- **Import:** The component imports necessary dependencies: React, useDispatch, useSelector from react-redux, and action creators (removeItemFromCart, clearCart, increaseItemQuantity, decreaseItemQuantity) from the CartSlice.

```
1. 1
2. 2
3. 3
1. import { useDispatch, useSelector } from 'react-redux';
2. import { removeItemFromCart, clearCart, increaseItemQuantity, decreaseItemQuantity } from './CartSlice'; // Assuming you have action
3. import './ShoppingCart.css';
Copied!
```

- **Function Component:** The ShoppingCart component is a functional component declared using the arrow function syntax.

- **Redux Hooks:** The component uses **useDispatch** and **useSelector** hooks from react-redux to interact with the Redux store. **useDispatch** is used to dispatch actions, and **useSelector** is used to extract data from the Redux store.

- **State Retrieval:** **cartItems** variable retrieves the array of items from the Redux store's state by selecting **state.cart.cartItems**. **totalAmount** calculates the total amount by iterating through **cartItems** and multiplying each item's price by its quantity, then summing them up.

```
1. 1
2. 2
3. 3
1. const dispatch = useDispatch();
2. const cartItems = useSelector(state => state.cart.cartItems);
3. const totalAmount = cartItems.reduce((total, item) => total + item.price * item.quantity, 0);
Copied!
```

Include above code before return of the function component.

- **Event Handlers:** **handleRemoveItem** dispatches the **removeItemFromCart** action with the ID of the item to be removed. **handleClearCart** dispatches the **clearCart** action to clear all items from the cart. **handleIncreaseQuantity** dispatches the **increaseItemQuantity** action to increase the quantity of a specific item. **handleDecreaseQuantity** dispatches the **decreaseItemQuantity** action to decrease the quantity of a specific item.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
1. const handleRemoveItem = itemId => {
2.   dispatch(removeItemFromCart(itemId));
3. };
4.
5. const handleClearCart = () => {
6.   dispatch(clearCart());
7. };
8.
9. const handleIncreaseQuantity = itemId => {
10.   dispatch(increaseItemQuantity(itemId));
11. };
12.
13. const handleDecreaseQuantity = itemId => {
14.   dispatch(decreaseItemQuantity(itemId));
15. };
Copied!
```

- **Rendering:** The component renders a shopping cart UI, listing each item in the cart along with a **<ul>** tag with class name **cart-items** with its name, price, quantity controls (buttons to increase or decrease quantity), and a remove button for each item. The total amount is displayed below the cart if it is greater than 0.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
1. {cartItems.map(item => (
2.   <li key={item.id} className="cart-item">
3.     <span>{item.name} - ${item.price}</span>
4.     <div className="quantity-controls">
5.       <button className="quantity-control-btn" onClick={() => handleDecreaseQuantity(item.id)}>-</button>
6.       <span> {item.quantity}</span>
7.       <button className="quantity-control-btn" onClick={() => handleIncreaseQuantity(item.id)}>+</button>
8.     </div>
9.   </li>
10. )}</ul>
11. <div>Total Amount: ${totalAmount}</div>
Copied!
```

```

9.           <button className="remove-item-btn" onClick={() => handleRemoveItem(item.id)}>Remove</button>
10.          </li>
11.        )}

```

**Copied!**

- o **Button Controls:** Quantity controls (- and + buttons) are provided to decrease or increase the quantity of each item. Clicking the - button invokes **handleDecreaseQuantity** with the item's ID. Clicking the + button invokes **handleIncreaseQuantity** with the item's ID.

- o A button labeled **Clear Cart** is provided to remove all items from the cart when clicked. It triggers the **handleClearCart** function.

```

1. 1
1. <button className="clear-cart-btn" onClick={handleClearCart}>Clear Cart</button>

```

**Copied!**

- if products have been added then only display totalAmount else renders nothing.

```

1. 1
1. <div>{totalAmount ? <div>'The total amount is {totalAmount}</div> : ''}</div>

```

**Copied!**

- Let's break it down:

- The outermost div element contains an expression inside curly braces {}.
- Inside the expression, there's a ternary operator (condition ? expression1 : expression2) used for conditional rendering in JSX.
- If totalAmount is truthy, a nested div element is rendered. This nested div contains a string 'The total amount is {totalAmount}', where {totalAmount} is intended to be the value of the totalAmount variable interpolated into the string.
- If totalAmount is falsy, an empty string is rendered.
- The result of the ternary operation is rendered inside the outer div element.

Click below to view code of **CartSlice.jsx**.

▼ Click here for code of **CartSlice.jsx**

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46

1. import { createSlice } from '@reduxjs/toolkit';
2. const initialState = {
3.   cartItems: [],
4. };
5. const CartSlice = createSlice({
6.   name: 'cart',
7.   initialState,
8.   reducers: {
9.     addItemAtCart(state, action) {
10.       const existingItem = state.cartItems.find(item => item.id === action.payload.id);
11.       if (existingItem) {
12.         existingItem.quantity += 1;
13.       } else {
14.         state.cartItems.push({ ...action.payload, quantity: 1 });
15.       }
16.     },

```

```

17.     removeItemFromCart(state, action) {
18.       state.cartItems = state.cartItems.filter(item => item.id !== action.payload);
19.     },
20.     clearCart(state) {
21.       state.cartItems = [];
22.     },
23.     increaseItemQuantity(state, action) {
24.       const itemToIncrease = state.cartItems.find(item => item.id === action.payload);
25.       if (itemToIncrease) {
26.         itemToIncrease.quantity += 1;
27.       }
28.     },
29.     decreaseItemQuantity(state, action) {
30.       const itemToDecrease = state.cartItems.find(item => item.id === action.payload);
31.       if (itemToDecrease && itemToDecrease.quantity > 1) {
32.         itemToDecrease.quantity -= 1;
33.       }
34.     },
35.
36.
37.   }
38. });
39. export const {
40.   addItemToCart,
41.   removeItemFromCart,
42.   clearCart,
43.   increaseItemQuantity,
44.   decreaseItemQuantity,
45. } = CartSlice.actions;
46. export default CartSlice.reducer;

```

**Copied!**Click below to view code of **ShoppingCart.jsx** component.

► Click here for code of ShoppingCart.jsx

## Step 9: Check the output

1. Stop the execution of the React application in the terminal by performing `ctrl+c` to quit.
2. Then, write the given command in the terminal and hit Enter.

```

1. 1
1. npm run preview

```

**Copied!**

3. To view your React application, refresh the already opened webpage for the React application on your browser. If it is not open, then click the Skills Network icon on the left panel. This action will open the "SKILLS NETWORK TOOLBOX." Next, select "Launch Application". Enter the port number **4173** in "Application Port" and click .

4. The output will display as per the given screenshot after adding products to the cart.

5. Add one more product and you will see the change in the total amount.

Please note that the `Add to Cart` button can be used only once to add a product. After this, it will be disabled and won't add the same product if you click on it again.

**Note-** To see the latest changes, you need to execute `npm run preview` again in the terminal.

**Congratulations! You have created an E-Commerce Data Rendering React application!**

## Step 10: Practice Task

1. Now in this practice exercise you will create one more component where you will implement the concept of super coin.
  - o Super coins are a form of loyalty or reward points offered by some e-commerce platforms or retailers as part of their customer loyalty programs. To see how much user have earned based on total cart amount you need to create this functionality.
2. For this create one Super Coin Component named `SuperCoin.jsx` by right clicking on the **Components** folder after selecting it.
3. Now initialize **superCoins** variable using `useState` hook along with its corresponding function before return of the component.

Hint: use `useState` hook to initialize variable with 0.

▼ Click here for the sample solution

```

1. 1
1. const [superCoins, setSuperCoins] = useState(0);

```

**Copied!**

4. Now you need to retrieve the `cartItems` from the cart slice of the Redux store's state to get the total quantity of number of products using the `useSelector` hook.

Hint: use `useSelector` hook to get the state of cart items.

▼ Click here for the sample solution

```

1. 1
1. const cartItems = useSelector(state => state.cart.cartItems);

```

**Copied!**

5. Next calculate the total amount by summing the product of the price and quantity for each item in the `cartItems` array using the `reduce` method.

Hint: use `useSelector` hook to get the state of cart items.

▼ Click here for the sample solution

1. 1

```
1. const totalAmount = cartItems.reduce((total, item) => total + item.price * item.quantity, 0);
```

**Copied!**

6. Now you need to update the `superCoins` state based on the `totalAmount`: setting it to 10, 20, or 30 coins for different ranges of the total amount, and to 0 if the amount is below 100. This effect runs whenever the `totalAmount` changes.

Hint: use `useEffect` hook to update the state of `superCoins`.

▼ Click here for the sample solution

1. 1

2. 2

3. 3

4. 4

5. 5

6. 6

7. 7

8. 8

9. 9

10. 10

11. 11

```
1. useEffect(() => {
2.   if (totalAmount >= 100 && totalAmount < 200) {
3.     setSuperCoins(10);
4.   } else if (totalAmount >= 200 && totalAmount < 300) {
5.     setSuperCoins(20);
6.   } else if (totalAmount >= 300) {
7.     setSuperCoins(30);
8.   } else {
9.     setSuperCoins(0);
10.  }
11. }, [totalAmount]);
```

**Copied!**

7. Now create `<div>` within return of function component using `jsx` syntax and integrate the `superCoins` variable state within `<div>` tag.

Hint: Use `{}` to include `superCoins` variable

▼ Click here for the sample solution

1. 1

2. 2

3. 3

4. 4

```
1. <div className="super-coins" style={{textAlign:'center'}}>
2.   <h2 className="super-coins-title">Super Coins</h2>
3.   <p className="super-coins-info">You will earn {superCoins} super coins with this purchase.</p>
4. </div>
```

**Copied!**

8. Check the output

- Save the changes and re-run the application.
- Add the product in the cart and when it reaches to **\$100** amount it will display that you have earned **10 Super Coins**.

9. You can increase the amount and depending upon the logic it will also increase supercoins value.

## Conclusion

**Congratulations! You have created an E-Commerce React application!**

In this lab, you have:

- Implemented Redux Toolkit for universal state management across the application.
- Developed a basic e-commerce platform using React and Redux.
- Featured a product list with an “Add to Cart” button for each item.
- Enabled users to view and manage items added to the cart, including removing items.
- Utilized the `useDispatch` and `useSelector` hooks to interact with Redux, providing global data accessibility.
- Ensured seamless state management throughout the application, enhancing user experience and scalability.

## Author(s)

Richa Arora

© IBM Corporation 2023. All rights reserved.