# Amazona ECommerce Website

![amazona](/template/images/amazona.jpg)

## PLEASE DO NOT SEND PULL REQUEST

## ALL PRs WILL BE REJECTED UNTIL DECEMBER 2020

# React & Node Tutorial - Full ECommerce in 9 Hours [2021]

Welcome to my React and Node tutorial to build a fully-functional e-commerce website exactly like amazon. Open your code editor and follow me for the next hours to build an e-commerce website using MERN stack (MongoDB, ExpressJS, React and Node.JS).

## Demo Website

- 👉 Heroku : [https://newamazona-final.herokuapp.com](https://newamazona-final.herokuapp.com)

- 👉 AWS : [https://amazona.webacademy.pro](https://amazona.webacademy.pro)

## You Will Learn

- HTML5 and CSS3: Semantic Elements, CSS Grid, Flexbox

- React: Components, Props, Events, Hooks, Router, Axios

- Redux: Store, Reducers, Actions

- Node & Express: Web API, Body Parser, File Upload, JWT

- MongoDB: Mongoose, Aggregation

- Development: ESLint, Babel, Git, Github,

- Deployment: Heroku

- Watch React & Node Tutorial

## Run Locally

**### 1. Clone repo**

```

$ git clone git@github.com:basir/amazona.git

$ cd amazona

```

**### 2. Setup MongoDB**

- Local MongoDB

- Install it from [here](https://www.mongodb.com/try/download/community)

- Create .env file in root folder

- Set MONGODB\_URL=mongodb://localhost/amazona

- Atlas Cloud MongoDB

- Create database at [https://cloud.mongodb.com](https://cloud.mongodb.com)

- Create .env file in root folder

- Set MONGODB\_URL=mongodb+srv://your-db-connection

**### 3. Run Backend**

```

$ npm install

$ npm start

```

**### 4. Run Frontend**

```

# open new terminal

$ cd frontend

$ npm install

$ npm start

```

**### 5. Seed Users and Products**

- Run this on chrome: http://localhost:5000/api/users/seed

- It returns admin email and password

- Run this on chrome: http://localhost:5000/api/products/seed

- It creates 6 sample products

**### 6. Admin Login**

- Run http://localhost:3000/signin

- Enter admin email and password and click signin

**## Support**

- Q/A: https://webacademy.pro/amazona

- Contact Instructor: [Basir](mailto:basir.jafarzadeh@gmail.com)

## Lessons

**1. Introduction to this course**

1. what you will build

2. what you will learn

3. who are audiences

**2. Install Tools**

1. Code Editor

2. Web Browser

3. VS Code Extension

**3. Website Template**

1. Create amazona folder

2. create template folder

3. create index.html

4. add default HTML code

5. link to style.css

6. create header, main and footer

7. style elements

**4. Display Products**

1. create products div

2. add product attributes

3. add link, image, name and price

**5. Create React App**

1. npx create-react-app frontend

2. npm start

3. Remove unused files

4. copy index.html content to App.js

5. copy style.css content to index.css

6. replace class with className

**6. Share Code On Github**

1. Initialize git repository

2. Commit changes

3. Create github account

4. Create repo on github

5. connect local repo to github repo

6. push changes to github

**7. Create Rating and Product Component**

1. create components/Rating.js

2. create div.rating

3. style div.rating, span and last span

4. Create Product component

5. Use Rating component

**8. Build Product Screen**

1. Install react-router-dom

2. Use BrowserRouter and Route for Home Screen

3. Create HomeScreen.js

4. Add product list code there

5. Create ProductScreen.js

6. Add new Route from product details to App.js

7. Create 3 columns for product image, info and action

**9. Create Node.JS Server**

1. run npm init in root folder

2. Update package.json set type: module

3. Add .js to imports

4. npm install express

5. create server.js

6. add start command as node backend/server.js

7. require express

8. create route for / return backend is ready.

9. move products.js from frontend to backend

10. create route for /api/products

11. return products

12. run npm start

**10. Load Products From Backend**

1. edit HomeScreen.js

2. define products, loading and error.

3. create useEffect

4. define async fetchData and call it

5. install axios

6. get data from /api/products

7. show them in the list

8. create Loading Component

9. create Message Box Component

10. use them in HomeScreen

**11. Install ESlint For Code Linting**

1. install VSCode eslint extension

2. npm install -D eslint

3. run ./node\_modules/.bin/eslint --init

4. Create ./frontend/.env

5. Add SKIP\_PREFLIGHT\_CHECK=true

**12. Add Redux to Home Screen**

1. npm install redux react-redux

2. Create store.js

3. initState= {products:[]}

4. reducer = (state, action) => switch LOAD\_PRODUCTS: {products: action.payload}

5. export default createStore(reducer, initState)

6. Edit HomeScreen.js

7. shopName = useSelector(state=>state.products)

8. const dispatch = useDispatch()

9. useEffect(()=>dispatch({type: LOAD\_PRODUCTS, payload: data})

10. Add store to index.js

**13. Add Redux to Product Screen**

1. create product details constants, actions and reducers

2. add reducer to store.js

3. use action in ProductScreen.js

4. add /api/product/:id to backend api

**14. Handle Add To Cart Button**

1. Handle Add To Cart in ProductScreen.js

2. create CartScreen.js

**15. Implement Add to Cart Action**

1. create addToCart constants, actions and reducers

2. add reducer to store.js

3. use action in CartScreen.js

4. render cartItems.length

**16. Build Cart Screen**

1. create 2 columns for cart items and cart action

2. cartItems.length === 0 ? cart is empty

3. show item image, name, qty and price

4. Proceed to Checkout button

5. Implement remove from cart action

**17. Implement Remove From Cart Action**

1. create removeFromCart constants, actions and reducers

2. add reducer to store.js

3. use action in CartScreen.js

**18. Create Sample Users In MongoDB**

1. npm install mongoose

2. connect to mongodb

3. create config.js

4. npm install dotenv

5. export MONGODB\_URL

6. create models/userModel.js

7. create userSchema and userModel

8. create userRoute

9. Seed sample data

**19. Create Sample Products In MongoDB**

1. create models/productModel.js

2. create productSchema and productModel

3. create productRoute

4. Seed sample data

**20. Create Sign-in Backend**

1. create /signin api

2. check email and password

3. generate token

4. install json web token

5. install dotenv

6. return token and data

7. test it using postman

**21. Design SignIn Screen**

1. create SigninScreen

2. render email and password fields

3. create signin constants, actions and reducers

4. Update Header based on user login

**22. Implement SignIn Action**

1. create signin constants, actions and reducers

2. add reducer to store.js

3. use action in SigninScreen.js

**23. Create Register Screen**

1. create API for /api/users/register

2. insert new user to database

3. return user info and token

4. create RegisterScreen

5. Add fields

6. Style fields

7. Add screen to App.js

8. create register action and reducer

9. check validation and create user

**24. Create Shipping Screen**

1. create CheckoutSteps.js component

2. create shipping fields

3. implement shipping constant, actions and reducers

**25. Create Payment Screen**

1. create payment fields

2. implement shipping constant, actions and reducers

**26. Design Place Order Screen**

1. design order summary fields

2. design order action

**27. Create Place Order API**

1. createOrder api

2. create orderModel

3. create orderRouter

4. create post order route

**28. Implement PlaceOrder Action**

1. handle place order button click

2. create place order constants, action and reducer

**29. Create Order Screen**

1. build order api for /api/orders/:id

2. create OrderScreen.js

3. dispatch order details action in useEffect

4. load data with useSelector

5. show data like place order screen

6. create order details constant, action and reducer

**30. Add PayPal Button**

1. get client id from paypal

2. set it in .env file

3. create route form /api/paypal/clientId

4. create getPaypalClientID in api.js

5. add paypal checkout script in OrderScreen.js

6. show paypal button

**31. Implement Order Payment**

1. update order after payment

2. create payOrder in api.js

3. create route for /:id/pay in orderRouter.js

4. rerender after pay order

**32. Display Orders History**

1. create customer orders api

2. create api for getMyOrders

3. show orders in profile screen

4. style orders

**33. Display User Profile**

1. create user details api

2. show user information

**34. Update User Profile**

1. create user update api

2. update user info

**35. Create Admin View**

1. Create Admin Menu

2. Create Admin Middleware in Backend

3. Create Admin Route in Frontend

**36. List Products**

1. Create Product List Screen

2. Add reducer to store

3. show products on the screen

**37. Create Product**

1. build create product api

2. build Create Product button

3. define product create constant, action and reducer

4. use action in Product List Screen

**38. Build Product Edit Screen**

1. create edit screen

2. define state

3. create fields

4. load product details

5. add to routes

**39. Update Product**

1. define update api

2. define product update constant, action and reducer

3. use action in Product Edit Screen

**40. Upload Product Image**

1. npm install multer

7. define upload router

8. create uploads folder

9. Handle frontend

**41. Delete Product**

1. create delete api in backend

2. create delete constants, action and reducer

3. use it in product list screen

**42. List Orders**

1. create order list api

2. create Order List Screen

3. Add reducer to store

4. show products on the screen

**43. Delete Order**

2. create delete order action and reducer

3. add order delete action to order list

**44. Deliver Order**

1. create constant, actions and reducers for deliver order

2. add order deliver action to order details screen

**45. Publish To Heroku**

1. Create git repository

2. Create heroku account

3. install Heroku CLI

4. heroku login

5. heroku apps:create <yourname>amazona

6. Edit package.json for build script

10. Create Procfile

12. Create mongodb atlas database

19. Set database connection in heroku env variables

20. Commit and push

**46. List Users**

1. build api for list users

2. Create UserList Screen

3. create order details constant, action and reducer

**47. Delete Users**

1. build api for delete users

2. create order details constant, action and reducer

3. Use action in UserListScreen

**48. Edit User**

1. build api for update users

2. create edit screen UI

**49. Implement Seller View**

1. add seller menu

2. create seller route

3. list products for seller

4. list orders for seller

5. add Seller to Product List and Details Screen

**50. Create Seller Page**

1. create seller page

2. update product component and product screen

3. update product routes

**51. Add Top Seller Carousel**

1. install react carousel

2. implement actions and reducers for top sellers

3. use react carousel with data in Home Screen

**52. Force Order Items From One Seller**

1. update addToCart action to buy from one seller at an order

**53. Create Search Box and Search Screen**

1. create search bar in Header.js

2. add style

3. handle submit form

4. edit parse url to get query string

5. update product list api for search by name

**54. Add Advanced Search Filter**

1. filter by category

2. filter by price range

3. filter by average rating

**55. Complete Advanced Search**

1. filter by price

2. filter by rating

3. sort by price, rating, ...

**56. Rate and Review Products**

1. rate products

2. create actions and reducers

**57. Choose Address On Google Map**

1. create google map credentials

2. update .env file with Google Api Key

3. create api to send google api to frontend

4. create map screen

5. fetch google api

6. getUserLocation

7. install @react-google-maps/api

8. use it in shipping screen

9. apply map to the checkout screen

**58. BugFix: Running Locally Without Issue**

1. add seller info to data.js

2. seed product data with admin info as seller

3. fix isSeller and isAdmin on update user

4. remove auth from user details

**59. Implement Pagination**

1. add pagination to product router in backend

2. apply page number to actions and reducers in frontend

3. show page numbers in search screen

**60. Email order receipt by mailgun**

1. create mailgun account

2. add and verify your domain to mailgun

3. install mailgun-js

4. set api key in env file

5. change pay order in orderRouter

6. send email the

**61. Create Dashboard Screen**

1. Create chart data in backend

2. Build Chart screen

**62. Implement Live Chat With Customers**

1. use socket io to create backend

2. create chat box component

3. create support screen