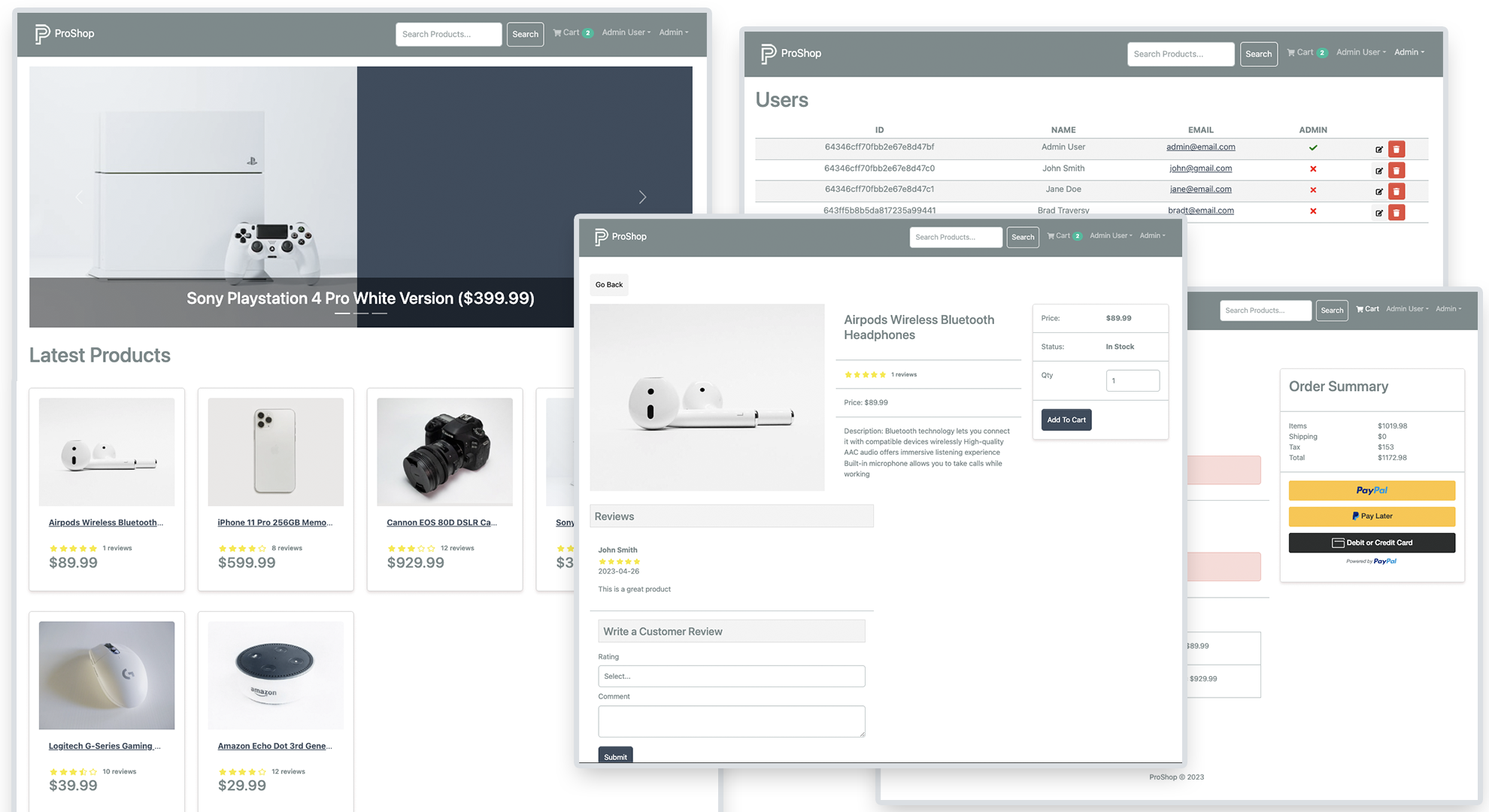
**ProShop eCommerce Platform (v2)**

eCommerce platform built with the MERN stack & Redux.

[](https://github.com/rothanakVISAL/proshop/blob/main/frontend/public/images/screens.png)

This project is part of my [MERN Stack From Scratch | eCommerce Platform](https://www.traversymedia.com/mern-stack-from-scratch) course. It is a full-featured shopping cart with PayPal & credit/debit payments.

This is version 2.0 of the app, which uses Redux Toolkit. The first version can be found [here](https://proshopdemo.dev/)

* [Features](https://github.com/rothanakVISAL/proshop#features)
* [Usage](https://github.com/rothanakVISAL/proshop#usage)
  + [Env Variables](https://github.com/rothanakVISAL/proshop#env-variables)
  + [Install Dependencies (frontend & backend)](https://github.com/rothanakVISAL/proshop#install-dependencies-frontend--backend)
  + [Run](https://github.com/rothanakVISAL/proshop#run)
* [Build & Deploy](https://github.com/rothanakVISAL/proshop#build--deploy)
  + [Seed Database](https://github.com/rothanakVISAL/proshop#seed-database)
* [Bug Fixes, corrections and code FAQ](https://github.com/rothanakVISAL/proshop#bug-fixes-corrections-and-code-faq)
  + [BUG: Warnings on ProfileScreen](https://github.com/rothanakVISAL/proshop#bug-warnings-on-profilescreen)
  + [BUG: Changing an uncontrolled input to be controlled](https://github.com/rothanakVISAL/proshop#bug-changing-an-uncontrolled-input-to-be-controlled)
  + [BUG: All file types are allowed when updating product images](https://github.com/rothanakVISAL/proshop#bug-all-file-types-are-allowed-when-updating-product-images)
  + [BUG: Throwing error from productControllers will not give a custom error response](https://github.com/rothanakVISAL/proshop#bug-throwing-error-from-productcontrollers-will-not-give-a-custom-error-response)
    - [Original code](https://github.com/rothanakVISAL/proshop#original-code)
  + [BUG: Bad responses not handled in the frontend](https://github.com/rothanakVISAL/proshop#bug-bad-responses-not-handled-in-the-frontend)
    - [Example from PlaceOrderScreen.jsx](https://github.com/rothanakVISAL/proshop#example-from-placeorderscreenjsx)
  + [BUG: After switching users, our new user gets the previous users cart](https://github.com/rothanakVISAL/proshop#bug-after-switching-users-our-new-user-gets-the-previous-users-cart)
  + [BUG: Passing a string value to our addDecimals function](https://github.com/rothanakVISAL/proshop#bug-passing-a-string-value-to-our-adddecimals-function)
  + [BUG: Token and Cookie expiration not handled in frontend](https://github.com/rothanakVISAL/proshop#bug-token-and-cookie-expiration-not-handled-in-frontend)
  + [BUG: Calculation of prices as decimals gives odd results](https://github.com/rothanakVISAL/proshop#bug-calculation-of-prices-as-decimals-gives-odd-results)
  + [FAQ: How do I use Vite instead of CRA?](https://github.com/rothanakVISAL/proshop#faq-how-do-i-use-vite-instead-of-cra)
    - [Setting up the proxy](https://github.com/rothanakVISAL/proshop#setting-up-the-proxy)
    - [Setting up linting](https://github.com/rothanakVISAL/proshop#setting-up-linting)
    - [Vite outputs the build to /dist](https://github.com/rothanakVISAL/proshop#vite-outputs-the-build-to-dist)
    - [Vite has a different script to run the dev server](https://github.com/rothanakVISAL/proshop#vite-has-a-different-script-to-run-the-dev-server)
    - [A final note:](https://github.com/rothanakVISAL/proshop#a-final-note)
  + [FIX: issues with LinkContainer](https://github.com/rothanakVISAL/proshop#fix-issues-with-linkcontainer)
  + [License](https://github.com/rothanakVISAL/proshop#license)

**Features**

* Full featured shopping cart
* Product reviews and ratings
* Top products carousel
* Product pagination
* Product search feature
* User profile with orders
* Admin product management
* Admin user management
* Admin Order details page
* Mark orders as delivered option
* Checkout process (shipping, payment method, etc)
* PayPal / credit card integration
* Database seeder (products & users)

**Usage**

* Create a MongoDB database and obtain your MongoDB URI - [MongoDB Atlas](https://www.mongodb.com/cloud/atlas/register)
* Create a PayPal account and obtain your Client ID - [PayPal Developer](https://developer.paypal.com/)

**Env Variables**

Rename the .env.example file to .env and add the following

NODE\_ENV = development

PORT = 5000

MONGO\_URI = your mongodb uri

JWT\_SECRET = 'abc123'

PAYPAL\_CLIENT\_ID = your paypal client id

PAGINATION\_LIMIT = 8

Change the JWT\_SECRET and PAGINATION\_LIMIT to what you want

**Install Dependencies (frontend & backend)**

npm install

cd frontend

npm install

**Run**

# Run frontend (:3000) & backend (:5000)

npm run dev

# Run backend only

npm run server

**Build & Deploy**

# Create frontend prod build

cd frontend

npm run build

**Seed Database**

You can use the following commands to seed the database with some sample users and products as well as destroy all data

# Import data

npm run data:import

# Destroy data

npm run data:destroy

Sample User Logins

admin@email.com (Admin)

123456

john@email.com (Customer)

123456

jane@email.com (Customer)

123456

**Bug Fixes, corrections and code FAQ**

The code here in the main branch has been updated since the course was published to fix bugs found by students of the course and answer common questions, if you are looking to compare your code to that from the course lessons then please refer to the [originalcoursecode](https://github.com/bradtraversy/proshop-v2/tree/originalCourseCode) branch of this repository.

There are detailed notes in the comments that will hopefully help you understand and adopt the changes and corrections. An easy way of seeing all the changes and fixes is to use a note highlighter extension such as [This one for VSCode](https://marketplace.visualstudio.com/items?itemName=wayou.vscode-todo-highlight) or [this one for Vim](https://github.com/folke/todo-comments.nvim) Where by you can easily list all the **NOTE:** and **FIX:** tags in the comments.

**BUG: Warnings on ProfileScreen**

We see the following warning in the browser console..

<tD> cannot appear as a child of <tr>.

and

warning: Received 'true' for a non-boolean attribute table.

Code changes can be seen in [ProfileScreen.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/screens/ProfileScreen.jsx)

**BUG: Changing an uncontrolled input to be controlled**

In our SearchBox input, it's possible that our urlKeyword is **undefined**, in which case our initial state will be **undefined** and we will have an uncontrolled input initially i.e. not bound to state. In the case of urlKeyword being **undefined** we can set state to an empty string.

Code changes can be seen in [SearchBox.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/components/SearchBox.jsx)

**BUG: All file types are allowed when updating product images**

When updating and uploading product images as an Admin user, all file types are allowed. We only want to upload image files. This is fixed by using a fileFilter function and sending back an appropriate error when the wrong file type is uploaded.

You may see that our checkFileType function is declared but never actually used, this change fixes that. The function has been renamed to fileFilter and passed to the instance of [multer](https://github.com/expressjs/multer" \l "filefilter)

Code changes can be seen in [uploadRoutes.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/routes/uploadRoutes.js)

**BUG: Throwing error from productControllers will not give a custom error response**

In section **3 - Custom Error Middleware** we throw an error from our getProductById controller function, with a *custom* message. However if we have a invalid **ObjectId** as req.params.id and use that to query our products in the database, Mongoose will throw an error before we reach the line of code where we throw our own error.

**Original code**

const getProductById = asyncHandler(async (req, res) => {

const product = await Product.findById(req.params.id);

if (product) {

return res.json(product);

}

// NOTE: the following will never run if we have an invalid ObjectId

res.status(404);

throw new Error('Resource not found');

});

Instead what we can do is if we do want to check for an invalid ObjectId is use a built in method from Mongoose - [isValidObjectId](https://mongoosejs.com/docs/api/mongoose.html" \l "Mongoose.prototype.isValidObjectId()) There are a number of places in the project where we may want to check we are getting a valid ObjectId, so we can extract this logic to it's own middleware and drop it in to any route handler that needs it.  
This also removes the need to check for a cast error in our errorMiddleware and is a little more explicit in checking for such an error.

Changes can be seen in [errorMiddleware.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/middleware/errorMiddleware.js), [productRoutes.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/routes/productRoutes.js), [productController.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/controllers/productController.js) and [checkObjectId.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/middleware/checkObjectId.js)

**BUG: Bad responses not handled in the frontend**

There are a few cases in our frontend where if we get a bad response from our API then we try and render the error object. This you cannot do in React - if you are seeing an error along the lines of **Objects are not valid as a React child** and the app breaks for you, then this is likely the fix you need.

**Example from PlaceOrderScreen.jsx**

<ListGroup.Item>

{error && <Message variant='danger'>{error}</Message>}

</ListGroup.Item>

In the above code we check for a error that we get from our [useMutation](https://redux-toolkit.js.org/rtk-query/usage/mutations) hook. This will be an object though which we cannot render in React, so here we need the message we sent back from our API server...

<ListGroup.Item>

{error && <Message variant='danger'>{error.data.message}</Message>}

</ListGroup.Item>

The same is true for [handling errors from our RTK queries.](https://redux-toolkit.js.org/rtk-query/usage/error-handling)

Changes can be seen in:-

* [PlaceOrderScreen.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/screens/PlaceOrderScreen.jsx)
* [OrderScreen.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/screens/OrderScreen.jsx)
* [ProductEditScreen.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/screens/admin/ProductEditScreen.jsx)
* [ProductListScreen.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/screens/admin/ProductListScreen.jsx)

**BUG: After switching users, our new user gets the previous users cart**

When our user logs out we clear **userInfo** and **expirationTime** from local storage but not the **cart**.  
So when we log in with a different user, they *inherit* the previous users cart and shipping information.

The solution is to simply clear local storage entirely and so remove the **cart**, **userInfo** and **expirationTime**.

Changes can be seen in:-

* [authSlice.js](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/slices/authSlice.js)
* [cartSlice.js](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/slices/cartSlice.js)
* [Header.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/components/Header.jsx)

**BUG: Passing a string value to our addDecimals function**

Our addDecimals function expects a **Number** type as an argument so calling it by passing a **String** type as the argument could produce some issues. It kind of works because JavaScript type coerces the string to a number when we try to use mathematic operators on strings. But this is prone to error and can be improved.

Changes can be seen in:

* [cartUtils.js](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/utils/cartUtils.js)
* [calcPrices.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/utils/calcPrices.js)

**BUG: Token and Cookie expiration not handled in frontend**

The cookie and the JWT expire after 30 days. However for our private routing in the client our react app simply trusts that if we have a user in local storage, then that user is authenticated. So we have a situation where in the client they can access private routes, but the API calls to the server fail because there is no cookie with a valid JWT.

The solution is to wrap/customize the RTK [baseQuery](https://redux-toolkit.js.org/rtk-query/usage/customizing-queries" \l "customizing-queries-with-basequery) with our own custom functionality that will log out a user on any 401 response

Changes can be seein in:

* [apiSlice.js](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/slices/apiSlice.js)

Additionally we can remove the following code:

const expirationTime = new Date().getTime() + 30 \* 24 \* 60 \* 60 \* 1000; // 30 days

localStorage.setItem('expirationTime', expirationTime);

from our [authSlice.js](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/slices/authSlice.js) as it's never actually used in the project in any way.

**BUG: Calculation of prices as decimals gives odd results**

JavaSCript uses floating point numbers for decimals which can give some funky results for example:

0.1 + 0.2; // 0.30000000000000004 🤯

Or a more specific example in our application would be that our airpods have a price: 89.99 and if we do:

3 \* 89.99; // 269.96999999999997

The solution would be to calculate prices in whole numbers:

(3 \* (89.99 \* 100)) / 100; // 269.97

Changes can be see in in:

* [PlaceOrderScreen.jsx](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/screens/PlaceOrderScreen.jsx)
* [cartUtils.js](https://github.com/bradtraversy/proshop-v2/tree/main/frontend/src/utils/cartUtils.js)
* [calcPrices.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/utils/calcPrices.js)

**FAQ: How do I use Vite instead of CRA?**

Ok so you're at **Section 1 - Starting The Frontend** in the course and you've heard cool things about [Vite](https://vitejs.dev/) and why you should use that instead of [Create React App](https://create-react-app.dev/) in 2023.

There are a few differences you need to be aware of using Vite in place of CRA here in the course after [scaffolding out your Vite React app](https://github.com/vitejs/vite/tree/main/packages/create-vite#create-vite)

**Setting up the proxy**

Using CRA we have a "proxy" setting in our frontend/package.json to avoid breaking the browser [Same Origin Policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy) in development. In Vite we have to set up our proxy in our [vite.config.js](https://vitejs.dev/config/server-options.html#server-proxy).

import { defineConfig } from 'vite';

import react from '@vitejs/plugin-react';

export default defineConfig({

plugins: [react()],

server: {

// proxy requests prefixed '/api' and '/uploads'

proxy: {

'/api': 'http://localhost:5000',

'/uploads': 'http://localhost:5000',

},

},

});

**Setting up linting**

By default CRA outputs linting from eslint to your terminal and browser console. To get Vite to ouput linting to the terminal you need to add a [plugin](https://www.npmjs.com/package/vite-plugin-eslint) as a development dependency...

npm i -D vite-plugin-eslint

Then add the plugin to your **vite.config.js**

import { defineConfig } from 'vite';

import react from '@vitejs/plugin-react';

// import the plugin

import eslintPlugin from 'vite-plugin-eslint';

export default defineConfig({

plugins: [

react(),

eslintPlugin({

// setup the plugin

cache: false,

include: ['./src/\*\*/\*.js', './src/\*\*/\*.jsx'],

exclude: [],

}),

],

server: {

proxy: {

'/api': 'http://localhost:5000',

'/uploads': 'http://localhost:5000',

},

},

});

By default the eslint config that comes with a Vite React project treats some rules from React as errors which will break your app if you are following Brad exactly. You can change those rules to give a warning instead of an error by modifying the **eslintrc.cjs** that came with your Vite project.

module.exports = {

env: { browser: true, es2020: true },

extends: [

'eslint:recommended',

'plugin:react/recommended',

'plugin:react/jsx-runtime',

'plugin:react-hooks/recommended',

],

parserOptions: { ecmaVersion: 'latest', sourceType: 'module' },

settings: { react: { version: '18.2' } },

plugins: ['react-refresh'],

rules: {

// turn this one off

'react/prop-types': 'off',

// change these errors to warnings

'react-refresh/only-export-components': 'warn',

'no-unused-vars': 'warn',

},

};

**Vite outputs the build to /dist**

Create React App by default outputs the build to a **/build** directory and this is what we serve from our backend in production.  
Vite by default outputs the build to a **/dist** directory so we need to make some adjustments to our [backend/server.js](https://github.com/bradtraversy/proshop-v2/tree/main/backend/server.js) Change...

app.use(express.static(path.join(\_\_dirname, '/frontend/build')));

to...

app.use(express.static(path.join(\_\_dirname, '/frontend/dist')));

and...

app.get('\*', (req, res) =>

res.sendFile(path.resolve(\_\_dirname, 'frontend', 'build', 'index.html'))

);

to...

app.get('\*', (req, res) =>

res.sendFile(path.resolve(\_\_dirname, 'frontend', 'dist', 'index.html'))

);

**Vite has a different script to run the dev server**

In a CRA project you run npm start to run the development server, in Vite you start the development server with npm run dev  
If you are using the **dev** script in your root pacakge.json to run the project using concurrently, then you will also need to change your root package.json scripts from...

"client": "npm start --prefix frontend",

to...

"client": "npm run dev --prefix frontend",

Or you can if you wish change the frontend/package.json scripts to use npm start...

"start": "vite",

**A final note:**

Vite requires you to name React component files using the .jsx file type, so you won't be able to use .js for your components. The entry point to your app will be in main.jsx instead of index.js

And that's it! You should be good to go with the course using Vite.

**FIX: issues with LinkContainer**

The LinkContainer component from [react-router-bootstrap](https://github.com/react-bootstrap/react-router-bootstrap) was used to wrap React Routers Link component for convenient integration between React Router and styling with Bootstrap.  
However **react-router-bootstrap** hasn't kept up with React and you may see warnings in your console along the lines of:

LinkContainer: Support for defaultProps will be removed from function components in a future major release. Use JavaScript default parameters instead.

Which is because React is removing default component props in favour of using default function parameters and LinkContainer still uses Component.defaultProps.  
However you don't really need LinkContainer as we can simply use the as prop on any React Bootstrap component to render any element of your choice, including React Routers Link component.

For example in our [Header.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/components/Header.jsx) we can first import Link:

import { useNavigate, Link } from 'react-router-dom';

Then instead of using LinkContainer:

<LinkContainer to='/'>

<Navbar.Brand>

<img src={logo} alt='ProShop' />

ProShop

</Navbar.Brand>

</LinkContainer>

We can remove LinkContainer and use the **as** prop on the Navbar.Brand

<Navbar.Brand as={Link} to='/'>

<img src={logo} alt='ProShop' />

ProShop

</Navbar.Brand>

**Changes can be seen in:**

* [Header.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/components/Header.jsx)
* [CheckoutSteps.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/components/CheckoutSteps.jsx)
* [Paginate.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/components/Paginate.jsx)
* [ProfileScreen.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/screens/ProfileScreen.jsx)
* [OrderListScreen.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/screens/admin/OrderListScreen.jsx)
* [ProductListScreen.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/screens/admin/ProductListScreen.jsx)
* [UserListScreen.jsx](https://github.com/rothanakVISAL/proshop/blob/main/frontend/src/screens/admin/UserListScreen.jsx)

After these changes you can then remove **react-router-bootstrap** from your dependencies in [frontend/package.json](https://github.com/rothanakVISAL/proshop/blob/main/frontend/package.json)

**License**

The MIT License

Copyright (c) 2023 Traversy Media [https://traversymedia.com](https://traversymedia.com/)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.