

---

# Fast Shopping App

React App & Node.js API

## SUMMARY

The general idea of this test is to build a small app which allows any user to make a purchase in a few steps.

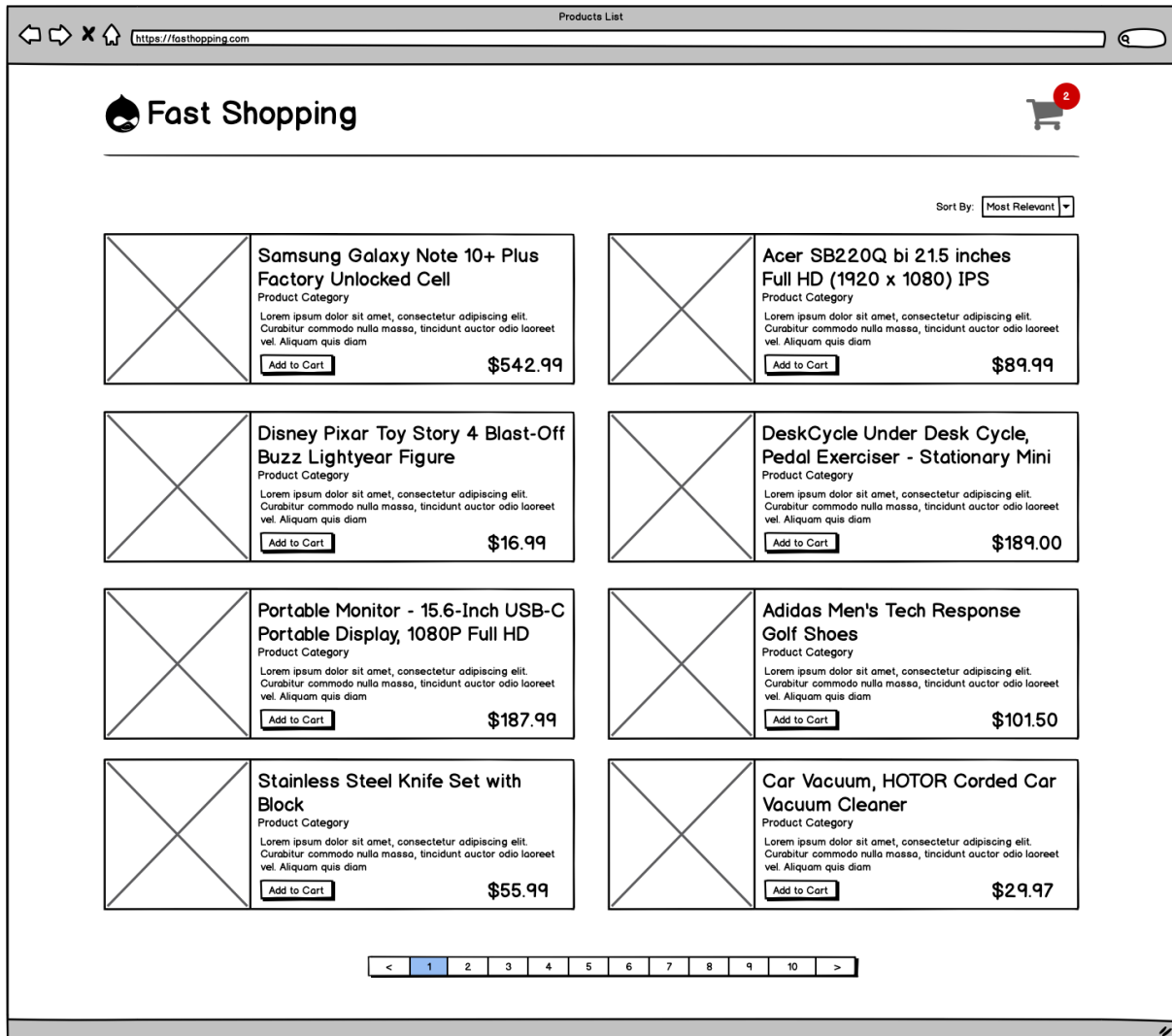
## USE CASES

### PRODUCTS LIST

As a customer, I need a page where I can browse the products that Fast Shopping offers. The products list is the main page of the site and it has:

- a. Header with logo and business name.
- b. At the right of the header, a shopping cart icon which displays how many items has been added to the cart. The icon is clickable to take user to the shopping cart page.
- c. Sort by control where users can sort by: **Alpha Order (default value), Lowest Price and Most Recent.**
- d. Next, users will find a list of cards. Each card represents a product, and displays:
  - i. Image.
  - ii. Name.
  - iii. Price.
  - iv. Category. A product could belong to multiple categories.
  - v. Brief description.
  - vi. CTA to add product to the cart. When a product is added the cart, the shopping cart bubble should be refreshed to display the new counter.
- e. Finally, if there are more than 20 items in the list, we'll see a paginator.

Here is a wireframe of the screen so you can visualize the desired layout.



## SHOPPING CART


As a customer, I need a shopping cart to create an order. The access point to this screen is the shopping cart icon at the top right of the product list page. Once the customer gets to the shopping cart page s/he will find:

- Page header, pretty similar to main page.
- List of added products. Here, the customer can delete a product from the cart, increase or decrease the quantity, see the unit price and total price per item.
- At the bottom, the customer will find the order total and a couple of buttons to go next or return to the product list page.

Here is the proposed design:









Shopping Cart

https://fastshopping.com/cart

 **Fast Shopping**

### Shopping Cart

Check Out

	<b>Adidas Men's Tech Response</b>  <small>Product Category</small>	Unit Price <b>\$101.50</b>	Qty <div>1 2 3 4</div>	<b>\$101.50</b>
	<b>DeskCycle Under Desk Cycle, Pedal Exerciser - Stationary...</b>  <small>Product Category</small>	Unit Price <b>\$189.00</b>	Qty <div>1</div>	<b>\$189.00</b>
	<b>Car Vacuum, HOTOR Corded Car Vacuum Cleaner</b>  <small>Product Category</small>	Unit Price <b>\$29.97</b>	Qty <div>2</div>	<b>\$59.94</b>
	<b>Acer SB220Q bi 21.5 inches Full HD (1920 x 1080) IPS</b>  <small>Product Category</small>	Unit Price <b>\$89.99</b>	Qty <div>1</div>	<b>\$89.99</b>

[Continue Shopping](#)

Total: **\$440.43**

Check Out

## CUSTOMER INFORMATION AND ORDER SUMMARY

As a customer, I need a way to enter my personal information, the address where I want to receive my order and the summary of the products being bought. The app resolves this requirement splitting this in two areas:

**Customer Information Area:** this section the place where the app gathers the customer information required to make the delivery. Given the app doesn't have a login or register process this area is a bit singular and its flow depends on new customer or existing customer:


1. When a customer has never bought anything on our site, it needs to fill the form that is shown when it chooses "New Customer".
2. If a customer has already bought in our site, we should have its contact information stored, so that, it can choose "Existing Customer" and fill in the input with the email used before and then clicking on "Lookup". The app brings the customer information.

In case they made a mistake typing the email, the app has the lookup again option to be able to enter a different email address.

**Order Summary:** This section is a table that shows the products that customer wants to buy along with the quantities and the totals. This is an informational area and customer cannot edit anything at this stage. The place order button creates the order and finishes the process.

Order Summary

https://fastshopping.com/checkout



Customer Information

Are you? ☒ New Customer ☐ Existing Customer

Full Name\*

ID\*

Address\*

Phone Number

Email\*

Order Summary

Product	Unit Price	Units	Total Price
Adidas Men's Tech Response Golf Shoes	10150	1	10150
DeskCycle Under Desk Cycle, Pedal Exerciser - Stationary	189.00	1	189.00
Car Vacuum, HOTOR Corded Car Vacuum Cleaner	29.97	2	59.94
Acer SB220Q bi 21.5 inches Full HD (1920 x 1080) IPS	89.99	2	179.98

Total: \$87.96

Place Order

This is the alternative flow when the customer chooses the “Existing Customer” option:

Existing Customer Flow

Step 1: search by email

Are you? ☐ New Customer ☒ Existing Customer

Email\*

Lookup

Step 2: user found

Are you? ☐ New Customer ☒ Existing Customer

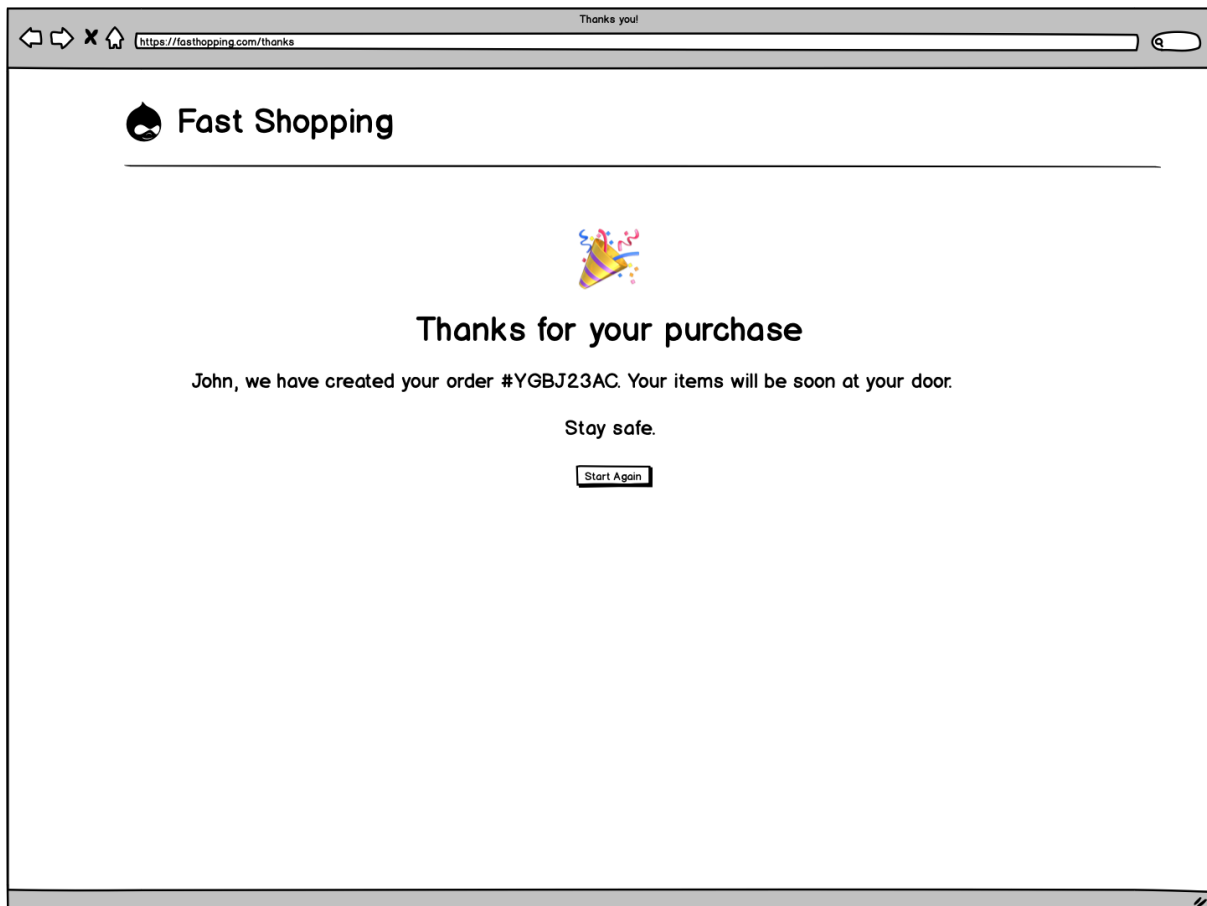
Welcome back, John Doe

ID: 123456  
Address: Main Street, #100-200, Wonderland  
Phone Number: 0000 000 000  
Email: johndoe@email.com

[Not John? Lookup again](#)

## THANKS SCREEN

As a customer I need a way to know if my order was processed successfully and my order number for tracking. That's why thank you page is created. The app only shows this screen if the order was created and it shows the order number. At the end of the page, there is a button to start buying again.



## TECHNICAL REQUIREMENTS

- The app must look good on desktops, tablets and mobiles - it **needs to be responsive**, make sure you **meet all of the requirements**. Please check the images delivered along this document to see wireframes with better resolution.
- Build the frontend with **React.js and CSS3**. Feel free using CSS preprocessors - use of styled components is a plus.
- Use **react router** to navigate between pages.
- Build the **server API with NodeJS**. Express or similars libraries/frameworks work.
- Data must be stored in a database. Please implement **MySQL**.

- Please use **redux and redux-sagas**, to handle the general state of the application and perform calls to the server. Also, you may use **mobx**. Choose whichever you feel more comfortable with.
- The **order needs to be persistent**. If browser is closed and reopened, **the order in progress should be still available**. We suggest redux-persist for this but feel free to implement your own approach.

## ASSETS AND LOOK AND FEEL

Feel free to use your own color scheme and icons. We encourage you to use [material ui](#) to leverage the development process. The only thing that it's required is to respect the layout proposed in the wireframes.

## HOW TO DELIVER

- Include a readme.md file in your app with instructions on how to run your project. Please make sure that following the readme file the application will run with no issues.
- Please deliver the source code on a public repository in Github.
- If you deploy the application on a public environment and share with us the url would be pretty nice! However this is not required but it is a plus.
- Include a SQL script to create the DB. We love to run and play with the app so please include some test data in the script.
- Please include an ER diagram of the DB.
- To deliver your test please "Reply to all" on the email thread where Sancrisoft sent these instructions. Include any information that you think is relevant to run and see your implementation.