

Amazon clone using React

Overview

Objective

You will be building an e-commerce application using React, which is inspired by Amazon.

Project Context

Businesses, in any field, have a lot of competition. They are always on the lookout for a proven way to increase their business revenue. Restaurants, retail shops, vape stores - whatever a business may want to sell, if the business doesn't have an e-commerce website, they are leaving money on the table!

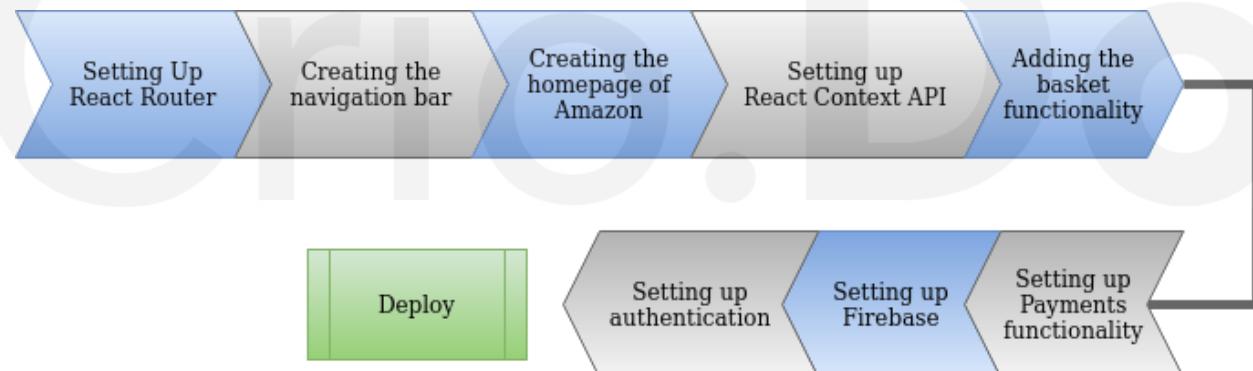
The world has moved online – a fact that businesses have to accept and put up a website to address. Amazon is a prime example of a website with all the key elements making up a good e-commerce site.

The e-commerce website of Amazon was initially put together with simple HTML, CSS and JavaScript. But as time progressed and different frameworks came into the limelight, the website got a makeover.

Through this project, we'll discover a way to build a functional clone of Amazon's e-commerce website, relying on React and Firebase.

Project Stages

The project consists of the following stages:



High-Level Approach

- The first task, once we get the development environment ready, will be to set up the React Router.

- Once we have everything in place, we can start off with creating the website header, which will basically serve as the navigation bar, as in most modern websites.
- Next up is the home page building. In this project, we'll be keeping it simple by showing all our sample products on the homepage.
- Then we'll be setting up the React Context API. The **Context API** is a component structure provided by the **React** framework, which enables us to share specific states across all levels of the application. In our project, we'll need to manage two states: basket (to manage the shopping cart) and user (for managing the details of the currently logged in user).
- For setting up the payments functionality, we'll be using APIs provided by **Stripe**.
- Handling our database and authentication needs to be supported and we'll be using **Firebase** for the same. Basically the database will be used to store the login information for the users, but the resource can be used for storing product information as well.
- Once we have **Firebase** setup, we can work on the Login page of our application.
- Successful implementation of the above requirements will lead to completion of the core implementation of our e-commerce solution. Next up, deploy!

At the end, we'll have a result similar to the one shown below.https://www.youtube.com/embed/ain1n_XR2ro

Primary goals

- Create header/navigation bar to navigate between pages.
- Create a home page to display products.
- Create a login page for user login.
- Add functionalities like basket, payment and authentication.

Credits

"Clever Programmer organization"

Task 1

Environment setup

Before the start of any development procedure, we need to set up the environment according to our application needs.

Requirements

- Install Node.js on your machine.
- Install and set up a React application.
- Start the development server to verify prior installation.

- Like any typical application, the source code should be in a `src` folder. Delete the 3 optional files from that folder: `App.test.js`, `logo.svg` and `setupTests.js`. You'll not be needing them.
- Remove all the content of the `App.css` file.
- Remove the unnecessary code from the `App()` function of `App.js` and add the below code to the `App()` function.

```
return <div className="app">Hello World!</div>;
```

- Add the following code to `index.css`.

```
*{  
margin: 0;  
padding: 0;  
}
```

It helps in tackling various formatting issues.

References

- [Download Node.js](#)
- [Create a New React App – React](#)
- [5 things you need to know for React](#)
- [Why use React? and not angular,vue](#)
- [Quick tutorial for React, want a lighter tutorial?](#)

Expected Outcome

The main objective of this milestone is to make sure that you have the required development environment in place.

On completion of the above requirements, run the application using `npm start` and the end result should be as shown in the screenshot below.



Task 2

Navigation with the React Router

React is a single page application. Which means that it doesn't support multiple routes by default. Routing is the ability to move between different parts of an application when a user enters a URL or clicks an element (link, button, icon, image etc) within the application. It enables us to transition from one view to another.

As mentioned earlier, moving between pages is one of the most salient features to be taken care of. So, in order to enable navigation in our application, we'll take the aid of a node package named **react-router-dom**.

Requirements

- Install React Router.
- Make a new component called "Home". **Components** are independent and reusable bits of code. They serve the same purpose as JavaScript functions but work in isolation and return HTML via a render function. Traditionally, every component has a .js file and a .css file. Follow the **BEM** (Block Element Modifier) convention when naming your components. We'll be working on functional components throughout this project. In simple words, functional components are javascript functions. By writing a javascript function, we can create a functional component in React Apps.
- Add the following code to your new component:

```
return <div className="home">Hello World Again!!</div>;
```

- To use the above component in our `App.js`, we need to set up the React Router. For doing so, first import the required dependencies.

```
import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
```

- Then, import the `Home` component and use it at `Route path = "/"`.

References

- [react-router-dom](#)
- [Block Element Modifier](#)
- [How To Create Custom Components in React](#)
- [How to handle Routing in React apps with React Router](#)
- [For the curious cats out there: Types of React Components you should know](#)

Expected Outcome

Your app should be able to render the data from the `Home` component by default at <http://localhost:3000>. The output should be as shown below.



Task 3

Creating the navigation bar

Now that we have our development environment and the React router in place, it's time to begin the actual job! We're going to start off by building the navigation bar of our web app.

Requirements

- We need a couple of icons in the navigation bar, like the search icon and the basket icon. React has a UI framework called **Material-UI**, which helps in building our frontend. Install the `@material-ui/core` and the `@material-ui/icons` packages.
- Create a new component called `Header`. As mentioned earlier, we'll be working on functional components throughout this project. Fulfill the following requirements:

- Add the amazon logo to the navigation bar.
- Add a search bar. The search bar should also have a search icon. It need not be functional at the moment.
- Create the navigation section of the header. Add the following sections:
 - Login
 - Returns & Orders
 - Amazon Prime
 - Shopping Basket
- Add the necessary code in the .css file of the component so that it looks similar to the one of Amazon.
- Include only the Header component in the App.js and use it at Route path = "/".

References

- Material-UI - Installation
- Material-UI - Usage
- The entire structure of a modern React app
- React Components
- Components in react
- Custom components

Expected Outcome

On completion of this milestone, the header of your webapp should look similar to the one shown below.



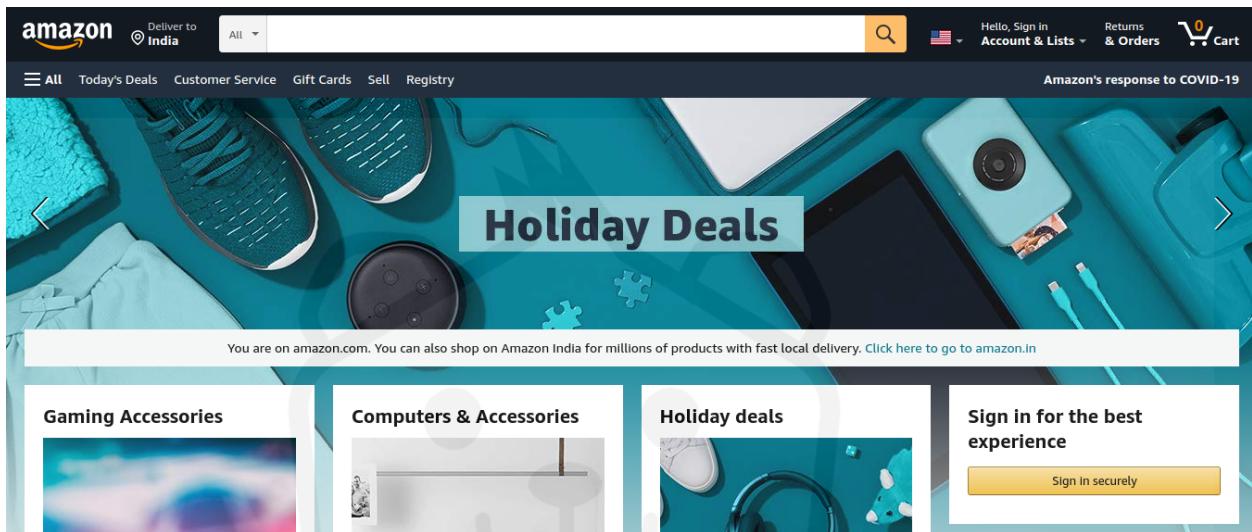
Task 4

Creating the homepage

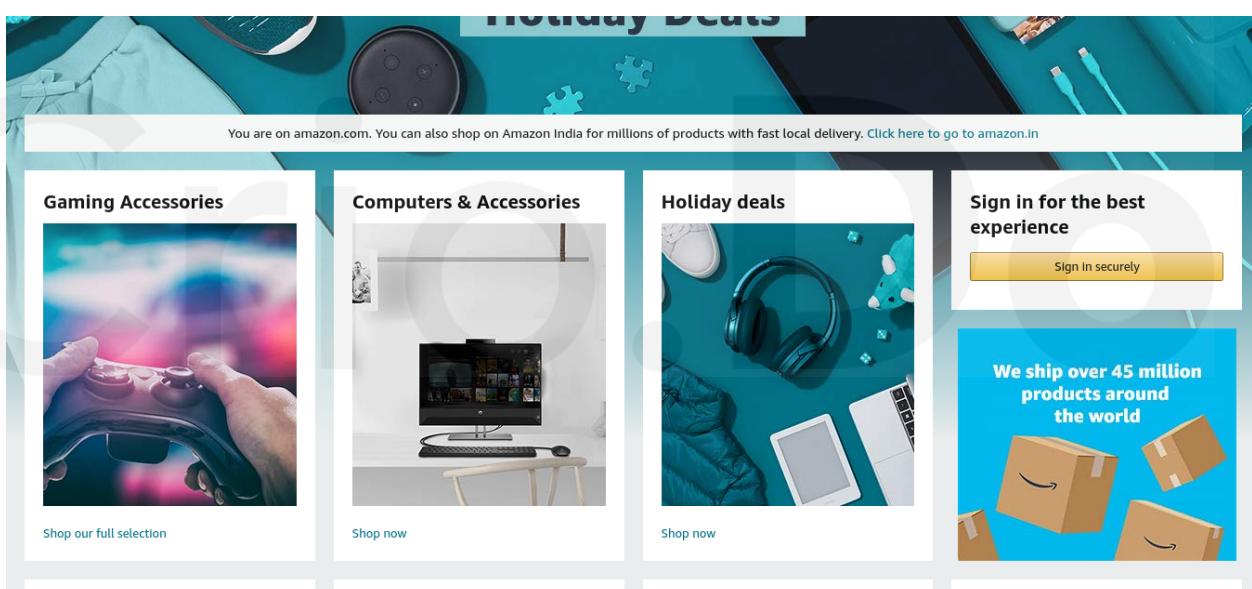
It's time for us to work on the homepage. The homepage of any website serves as the default page of that website. That is reason enough for the homepage to be really expressive as well as creative.

Requirements

- Go to your `Home.js`, i.e., your Home component, and do the following:
 - Similar to the image shown below, add an image to be displayed at the top of the landing page. You need to add this image in your home component.



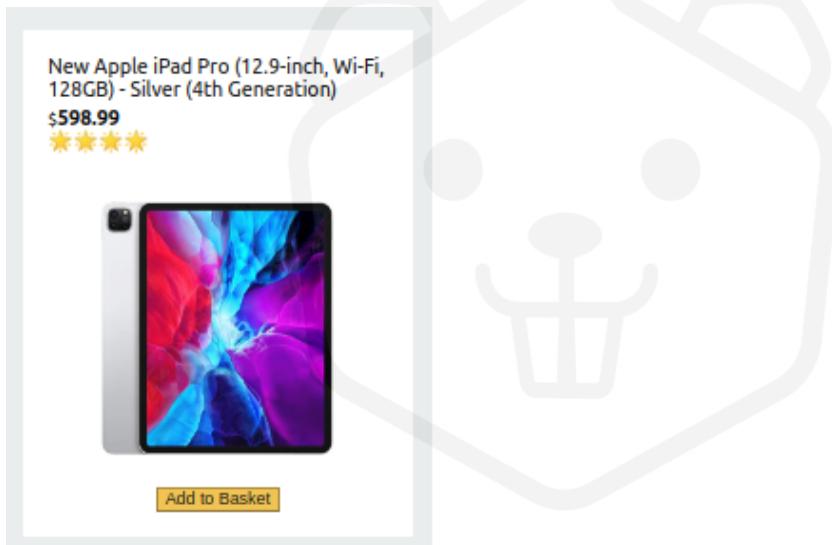
- If you observe the given image properly, you'll notice that there's a gradient style applied to the landing page image, which continues to turn into a greyish shade. You need to apply a similar gradient to the image you added. Go to [Amazon.com](#) or observe the image given below to get an idea about what we're expecting.



As you can see, since the products have a white background, it's easy to spot the contrast between the background of the rectangular components for 'Gaming'.

Accessories', 'Computers & Accessories', etc., and the surrounding greyish shade.

- Create a Product component. As mentioned before, you'll need two files: `Product.js` and `Product.css`. To attain reusability, pass parameters (referred to as `props` in `React.js`) like `id`, `title`, `image`, `price` and `rating` to your functional component (we'll be discussing functional components throughout this project, but you are free to explore other alternatives).
- Utilize the Product component in your Home component to render out a few sample products. Use proper `flex` CSS styles where required to get the desired (similar to [Amazon.com](#) products) result. You can refer to the image below to get an idea about how your component should look like on rendering.



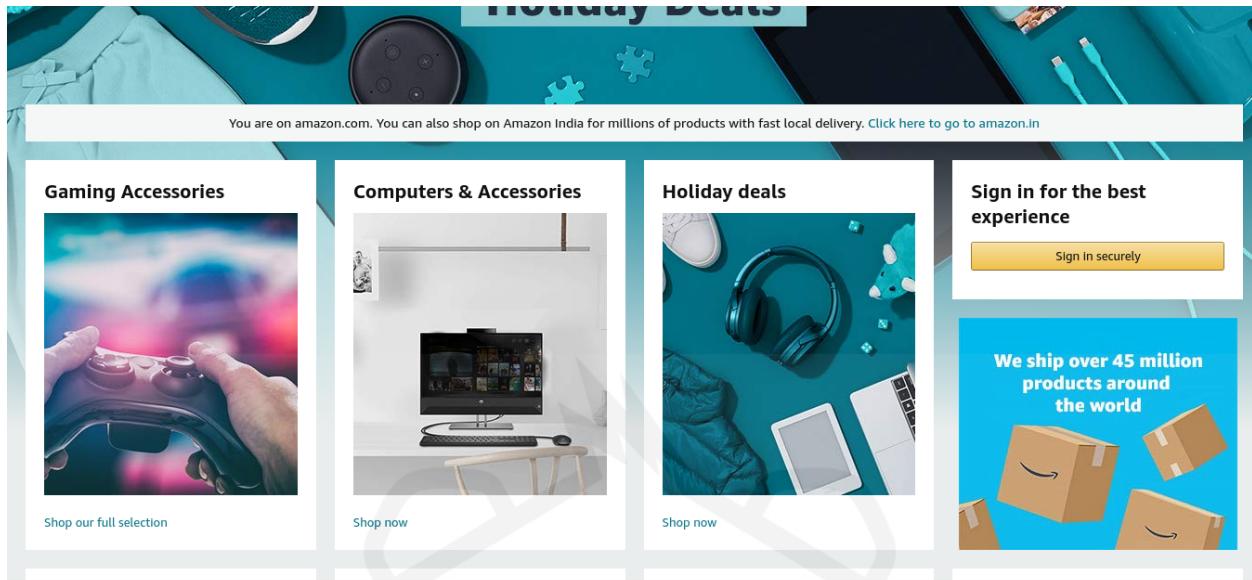
[Note: We are not expecting you to integrate database features to your application yet, such that it can access the product information from a database. We'll be discussing that in the optional milestones.]

References

- [Customize React components with props](#)
- [CSS Gradients](#)

Bring it On!

- Can you add a new component called Selection for example? In the given image below, you can see that there's a 'Gaming Accessories' selection, another for 'Computers & Accessories', etc. Your functional component in `Selection.js` will need 4 props: `id`, `title`, `image`, `route`. The given `route` in the component is to be responsible for redirecting the users to a page with the content related to the Selection component's link they clicked. For example, on [Amazon.com](#), if you click on [Show full collection](#) as given in the image below for "Gaming Accessories", it redirects you to a page with such products.



amazon Deliver to India Video Game Accessories

All Today's Deals Customer Service Gift Cards Sell Registry Amazon's response to COVID-19

1-16 of over 20,000 results Sort by: Featured

Department
Accessories

Avg. Customer Review
★★★★★ & Up
★★★★☆ & Up
★★★☆☆ & Up
★★☆☆☆ & Up

Featured Brands
 PlayStation
 Microsoft
 Nintendo
 amFilm
 OIVO
 Logitech G
 Orzly

Price
Under \$10
\$10 to \$15
\$15 to \$25
\$25 to \$35

Best Seller

DualSense Wireless Controller
ESRB Rating: Rating Pending | by PlayStation
★★★★★ ~ 3,750
PlayStation 5
\$69⁹⁶
Ships to India
More Buying Choices
\$56.45 (105 used & new offers)

Best Seller

Xbox Core Controller - Robot White
by Microsoft
★★★★★ ~ 829
Xbox One
\$59⁷⁸
Ships to India
More Buying Choices

Expected Outcome

On completion of this milestone, you should have a home page ready with sample products, which are rendered on the page at <http://localhost:3000> by default.

The result should be similar to the web page shown below.



The screenshot shows the Amazon homepage with a search bar at the top. Below the search bar, there's a banner for Prime Video with the text "prime video" and "Watch now". To the right of the banner is a grid of movie and TV show thumbnails, including "Knives Out", "The Office", "Hustlers", "Seamen", and "Alex Rider". The main content area displays four product cards:

- The Lean Startup: How Constant Innovation Creates Radically Successful Businesses Paperback**
\$11.96
★★★★★

[Add to Basket](#)
- Kenwood kMix Stand Mixer for Baking, Stylish Kitchen Mixer with K-beater, Dough Hook and Whisk, 5 Litre Glass Bowl**
\$239
★★★★★

[Add to Basket](#)
- Samsung LC49RG90SSUXEN 49" Curved LED Gaming Monitor**
\$199.99
★★★★

- Amazon Echo (3rd generation) | Smart speaker with Alexa, Charcoal Fabric**
\$98.99
★★★★★

- New Apple iPad Pro (12.9-inch, Wi-Fi, 128GB) - Silver (4th Generation)**
\$598.99
★★★★★


Task 5

Setting up React Context API

The Context API provides a way to share data values between components without having to pass a prop through every level of the app tree. The React Context API is a component structure provided by the React framework. It basically solves the problem of [prop drilling](#).

Requirements

- Create a file `StateProvider.js` with the following contents:

```
import React, { createContext, useContext, useReducer } from "react";
// Prepare the dataLayer
export const StateContext = createContext();

// Wrap our app and provide the Data layer
export const StateProvider = ({ reducer, initialState, children }) => (
  <StateContext.Provider value={useReducer(reducer, initialState)}>
    {children}
  </StateContext.Provider>
);

// Pull information from the data layer
export const useStateValue = () => useContext(StateContext);
```

- Create a file `reducer.js` which fulfills the following criteria:
 - Declare all application level states which are to be used later by the application. (You'll need at least 2 states, namely basket, to contain shopping basket details of logged in user, and user, to contain details of the logged in user).
 - Create a function which can calculate the total amount of the items in the shopping basket.
 - Export the reducer.
- Do the following in `index.js`:
 - Import the `reducer` and `initialState` from `reducer.js` and `StateProvider` component from `StateProvider.js`.
 - Enclose the `App` component with `StateProvider` so that the children components can get access to the states too.
 - Pass the `reducer` and `initialState` to the `StateProvider`.

References

- [State and Lifecycle in React](#)
- [React Context API: What's it all about?](#)

- Learn React Context in 5 Minutes - A Beginner's Tutorial
- An Introduction To React's Context API

Expected Outcome

You should be able to configure **React Context API**, along with the reducer, so that the components can access the basket and user info.

Task 6

Adding the Basket functionality

What's more exciting than adding items to your basket when you're shopping? In this milestone, we'll be adding this functionality to our application.

Requirements

- In your Product.js, import the useStateValue hook from the StateProvider.
- You need to use the useStateValue hook to bring in a state named basket (for example) and also get a dispatch function which allows you to dispatch actions to change the state in the reducer.

```
const [{ basket }, dispatch] = useStateValue();
```

- You need to dispatch the item information to the data layer. Your code should look similar to the code block shown below:

```
const addToBasket = () => {
  dispatch({
    type: "ADD_TO_BASKET",
    item: {
      id: id,
      title: title,
```

```
image: image,  
price: price,  
rating: rating,  
},  
});  
};
```

- Also, you need a `onClick` function for executing the above dispatch code.

```
<button onClick={addToBasket}>Add to Basket</button>
```

- In the `reducer.js` you need to add something similar to this to the `switch (action.type)` block:

```
case "ADD_TO_BASKET":  
return {  
...state,  
basket: [...state.basket, action.item],  
};
```

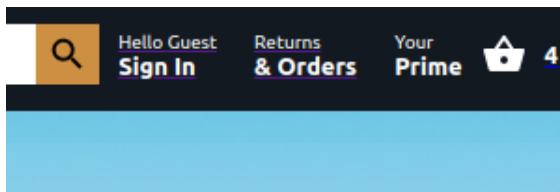
- Import the `useStateProvider` to your `Header` component and make use of the basket information to show the number of items in the basket.

References

- [Organizing state in Redux](#)
- [What Even Is A Dispatch Function?](#)

Expected Outcome

You should be able to see the updated count of the number of items in the basket beside the basket icon. Suppose we've got 4 items in the basket, then the desired result will be as shown below.



Task 7

Setting up Firebase and Authentication

Firebase is a great service provided by Google for configuring the backend of any web application with all the general necessities like database preparation, authentication using various methods, etc. In this milestone, we'll be preparing our database and setting up authentication using email and password.

[Note: Use the references provided to implement the following requirements.]

Requirements

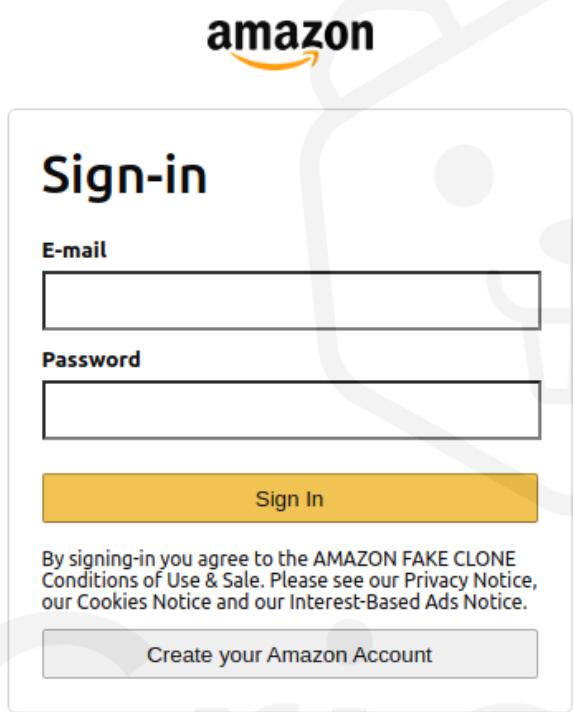
- Create a project on Firebase.
- Setup the database and setup sign-in method using Email/Password.
- Register your application and set up Firebase hosting.
- Install necessary Firebase dependencies in your local setup.

```
npm install -g firebase-tools && npm install firebase && firebase login
```

- Open the `Firebase SDK snippet` section in your Firebase project and copy paste the necessary configuration in a file named `firebase.js`.
- In this file, utilize the below code to make use of Firebase authentication and database.

```
const firebaseApp = firebase.initializeApp(firebaseConfig);
const db = firebaseApp.firestore();
const auth = firebase.auth();
export { db, auth };
```

- Create a new component named `Login`. Style the component so that it looks similar to the one shown below.



[Note: Feel free to check out [Amazon.com](https://www.amazon.com) for reference.]

- Configure the Firebase authentication with the `Login` component.

References

- Setup up database and connect it to your application, [additional reference](#)
- [Firebase authentication - 1, 2](#)
- [OnAuthStateChanged To Listen For Login/Register](#)
- [AuthStateListener in Firebase](#)
- [An example for authentication for React apps using firebase](#)
- [Sign-out feature using firebase authentication](#)

Bring it On!

- Configure other sign-in methods like FB sign-in, Google sign-in, etc.

Expected Outcome

You should be able to create a Firebase project and prepare database and authentication for your application. Also, you should have a functional `Login` component in place.

Task 8

The Checkout page

Now it's time for us to work on the checkout page. So, let's get right into it.

Requirements

- Create a `Checkout` component on the `Checkout` page. The `Checkout` component renders this page and will use two more components. Let's name them `CheckoutProduct` and `SubTotal` for example.



Exclusive offer for Amazon customers

Ocean credit card

39.9% APR Representative variable

Learn more

Hello, ayushshawz@gmail.com

Your shopping Basket

Kenwood kMix Stand Mixer for Baking, Stylish Kitchen Mixer with K-beater, Dough Hook and Whisk, 5 Litre Glass Bowl \$239

★★★★★

Remove from Basket

Intelligent Lending Ltd (credit broker). Capital one (exclusive lender).

Subtotal (1 items): \$239

This order contains a gift

Proceed to Checkout

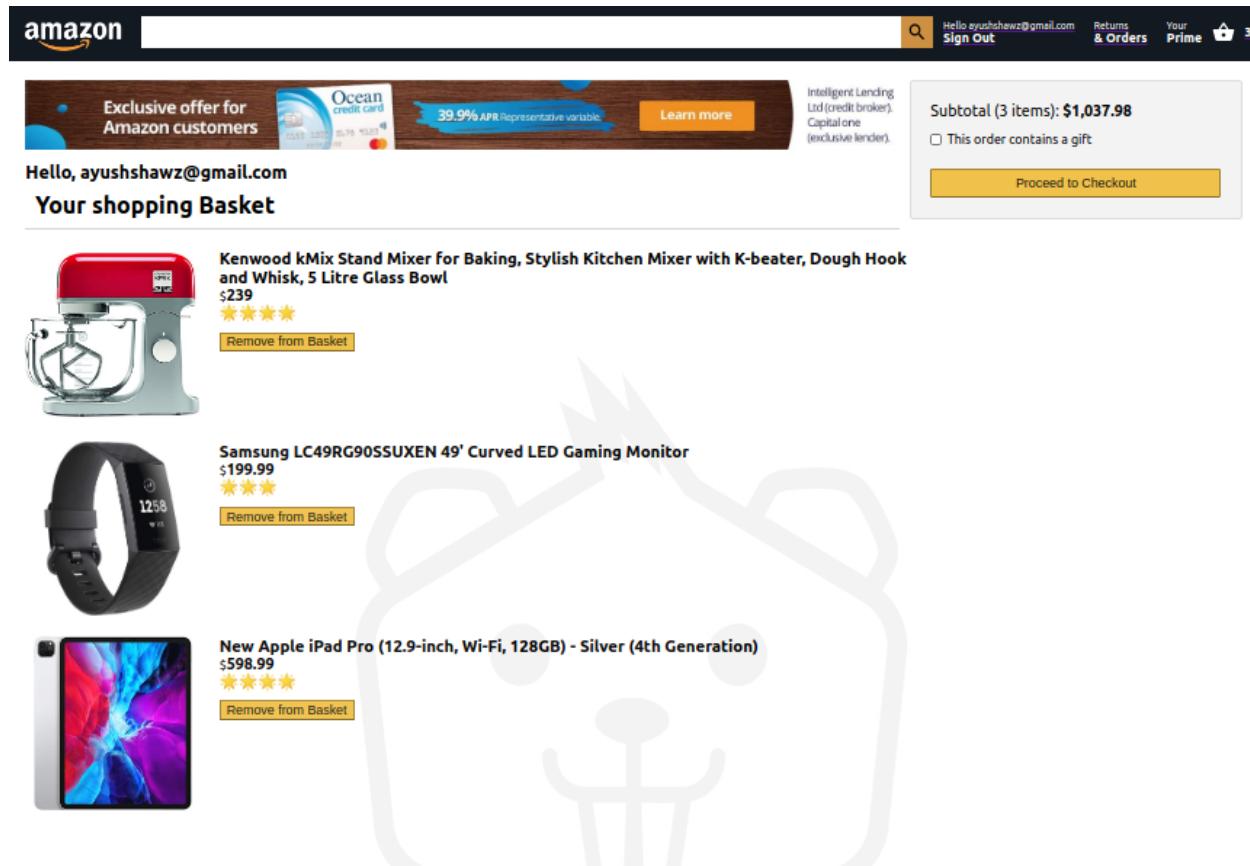
SubTotal Component

CheckoutProduct Component

- As you can see in the above image, we need the components styled as shown. [Note: You are free to refer to [Amazon.com](#) too.]
- The mentioned components should be functional as desired. The `SubTotal` component should be able to calculate the total price and display the same and the `Remove From Basket` button of the `CheckoutProduct` component should be functional such that on clicking the button, the product is removed from the basket and the price is deducted from the `SubTotal` component.

Expected Outcome

You should have a fully functional `Checkout` Page at your disposal. With a few products in the basket, the `Checkout` page might look something like this:



amazon

Hello ayushshawz@gmail.com

Returns & Orders Your Prime

Exclusive offer for Amazon customers  **39.9% APR Representative variable**

Intelligent Lending Ltd (credit broker). Capital one (exclusive lender).

Subtotal (3 items): \$1,037.98

This order contains a gift

Task 9

Task Title

Spice It Up!

Payments functionality and more

By now our application looks good and is a proper e-commerce solution, except it is missing one very important aspect of commerce and that is the payment facility. Moreover, this application is a clone inspired from [Amazon.com](#). We can add more functionality and tinker with the styling of the application to make it our own.

[Note: This milestone is not mandatory. Those who want to go that extra step are welcome to give it their best!]

Requirements

- Integrate [Stripe](#) in your React application.
- Give a new name to your e-commerce application and make a new logo for it. Use that logo in place of the Amazon logo.
- Change the styles of all the visual components.

- Integrate Firebase database functionality for storing product information in the database provided by Firebase and fetch the product information when required to display on the website.

References

- Integrating Stripe in your React app with React Stripe.js
- Retrieving data from database
- Add real-time database with your React app (refer only)

Bring it On!

- Reach out to local businesses and present them your product and offer them the chance to make their business go online! This might spark a potential freelance career.

Expected Outcome

You should be able to integrate payment functionality using Stripe in your React application. Also, you should be able to customize the website to your needs and make it look like your own.

Task 10

Deployment!

It's time to deploy our website and show it to the world!

Requirements

- Build your React application.

```
npm run build
```

- Setup Firebase hosting and deploy!

```
firebase deploy
```



References

- Deploy and Host a React app on Firebase
- Deploying on Firebase
- Difference between deployment and hosting

Bring it On!

- Get your own domain name and configure it with your deployed application.

Expected Outcome

You should be able to deploy your React application to Firebase so that it's accessible to everyone on the internet.



Crio.Do