## Part 01:

"start": "PORT=2000 react-scripts start",

A picture containing shape

Description automatically generated

Created a basic template with index.html & style.css.

## Part 02: Install tools

## Part 03: Design web template:

A picture containing shape

Description automatically generated

## Part 04: Display Products

Graphical user interface, application

Description automatically generated

Create Index.html., style.css file and created basic UI Design. i.e implemented a basic list of products using html tag i.e 5 div tags.

Graphical user interface, application, Teams

Description automatically generated

## Part 05: Create React App

Create basic react app and move main div tag of index.html to react application, at the end of this lesson we are able to see product list from static js array.

To create react application Npx create-react-app frontend

Cd frontend

Npm start

Copy paste content of body tag into app.js file. Remove certain files-app.css, logo-svg , copy style.css and paste it under body of index.css.

Copy images folder to public folder under frontend

All data in App.js is static data- hard coded 5 times to display 5 elements, and we need to load data from js array. To do that:

Go to src folder: create data.js file (which is source of products)

Data.js: recording all details of products

Text

Description automatically generated

Update App.JS file, and instead of having 6 card, we want to have only one cart div, we use map function in app.js file to convert data.js product items to html elements. And convert static name to product.name.

Hence just keep the first div and remove all other div’s.

We import data from data.js: import data from ./data

Text

Description automatically generated

Using map:

Data is a object, products is array of products stored in data.js file.

Text

Description automatically generated

So basically we load data of all the items from a javascript array into the main div, and use map function which iterates for n time(= no of products i.e for each product from products invoke everything written under () of map(()=>()))

## Part 07: Create Product components

What are components in react: [React Components (w3schools.com)](https://www.w3schools.com/react/react_components.asp)

The main concept of react is to use components- basically reuse code: It is a good practice to to write all the components into a component folder and use it whenever needed.

Graphical user interface, text, application

Description automatically generated

So basically we make the code more clean and reusable. So we copy the content of div in map function and paste in component file called - product.js. so we can now invoke the entire content of product.js file as <product></product>, also to display the content of each product object we have to pass an object as: product={product}, is an product object that stores 1 product

Graphical user interface, text

Description automatically generated

We write the code to display product inside react functional component:

Graphical user interface, text, application, chat or text message

Description automatically generated

How does the react component determine which value to view ? so for this we need to pass props

A screenshot of a computer screen

Description automatically generated with medium confidence

## Part 08: Create HomeScreen, ProductScreen, add routes from product details to app.js

Graphical user interface, text

Description automatically generated

Go to frontend fol: npm install react-router-dom

By installing router dom we can define multiple routes for our application and for each route specific screen response will be shown to user.

To implement route go to app.js and wrap the whole content inside return function to <Browser router> tag and we are ready to define routes

So 1st route is for homescreen and the components that respond to this is{Homescreen} when we set exact attribute it means, only if url=”/” then render this element.

Route 2: for product screen



When user enters address with this format we will render product screen. Id: contains the id of the products



Create a new screen folder and add all screens under screens folder.

Now how does routing work?

For every product image we have specified <a href =”product/${product\_id}”> hence this means when we click on this link - url will be localhost:3000/product/3. Here we get value=3 from product.\_id .

Now when we write this: 

It means when you are using url as mentioned above, it will redirect us to product screen component.

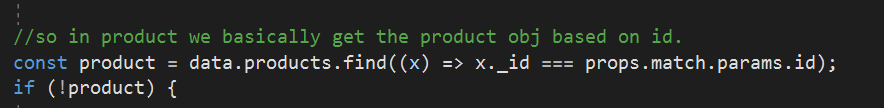
Eg:

Graphical user interface, text, application

Description automatically generated

Now for all the product id be it 1/2/3.. we get this same page. Now how do we ensure that particular product page opens up when we click that product : we write this code in product screen.

We fetch the product from array of products whose id is mentioned by user in the url. Now using this product object we can display all of its details on the screen.



find is an array function that finds an element based on the criterion that we define.

here we are searching for x.\_id which is the id for inside products array and comparing it with props.match.params.id

props.match.params.id = the value that user enter in id in route path="/product/:id" in URL

we compare this value with items in products array

Graphical user interface, text, application, email

Description automatically generated

then in return() we write the code to display details of particular product.

## Part 09:

## Here we start building Backend, we create a very basic node.js application that build an api for us that returns list of products to frontend – server.js

Text

Description automatically generated

So do npm init in amazone-clone folder, once we do that we will have package.json file in amozone-clone .

Next -Create a backend folder:

create server.js,it is the entry point of our application here we create an express server, express is a node package that helps creating a server very easily and powerful, install express: npm install express. This will add dependencies section and express will appear in package.json

In server.js-> import express, and create an app for express

Text

Description automatically generated

To open this:

Type in terminal-> node backend/server.js

Click on log link and u see :

A picture containing text

Description automatically generated

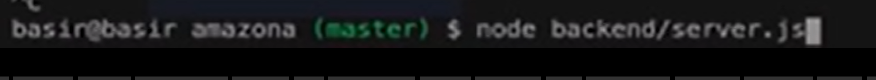
We successfully created our first server using node and express

Copy data.js folder from src to backend folder. So now we have all the products in backend.

Now in server.js we add a route: app.get(‘/api/products’) and its body we return data.products – list of products

So in frontend when we hit on api /products this route activates and sends list of products from data.js

To see this, Go to frontend:

Start the server 

In the URL, type: Graphical user interface, text, application

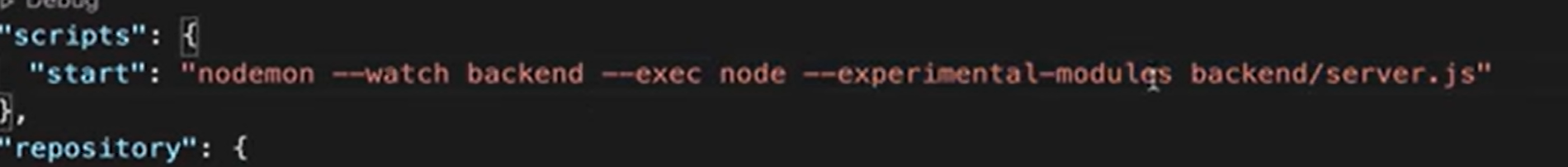
Description automatically generated

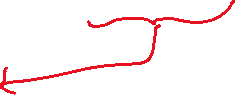
Result:

Text, letter

Description automatically generated

Install nodemon- to automatically re run your application when there is a change in your code (hende we wont have to re run server every time from terminal)





So when there is a change : this command will run.

## Part 10: Load products from Backend:

Text

Description automatically generated

Here we fetch products from backend, we have ajax request at \api\products and this return list of products to us. We have used this api using axios library. Have also created loading box component to show loading before fetching data from backend.

Step 1: Setting proxy to package.json file of frontend folder, so when you send a request to proxy it will be proxied/redirected to backend.

2. Install Axios. It is a library to send ajax request to backend.

3. And in home screen instead of fetching data from data.js from frontend we are fetching data from backend i.e /api/products .

4. Define react hook. To define hook state we use usestate, this hook is used to manage the state of this react component



Products is an empty array, and when we want to change value of products we use setproducts

We fil this products using useffect function .

It is another hook that happenes when your component didmount your webpage- that is after rendering the components this function will run.

It accepts 2 parameters. 1st : function , 2nd: array

In useEffect we send request to backend and fetch all the details.

Graphical user interface, application

Description automatically generated

This array accepts list of dependencies, by having this and after rendering your component this function will run only for 1 time and that is what we want – so here we will send ajax req to backend and fetch details from backend

This func is an assync fun becoz ajax req is an assyn req and in the body of this request we can fetch data from backend.

Graphical user interface, text

Description automatically generated

By passing this fun array in backend will be transferred to data in frontend.

Next step is to setting products that we are getting from backend.

Hence products array contains data from backend. And then we call getchData()

// You fixed your own error finallyyyyyyy. Fixed proxy settings 😊.

Here able to load data from backend

Application

Description automatically generated with medium confidence

Graphical user interface, application, Teams

Description automatically generated

Now we add loading component, initially we set loading as false as by default we are not loading anything.

We are going to use setLoading and set it true before sending ajax req (axios.get), and set it false after loading .

We use one more hook for error, we set the error to false by default as there are no error. To catch error we wrap the ajax req in try catch block. So in case if there is any error in ajax req. we catch it in catch block and set setError(err.message) and loading false

## Part 12: Add Redux to HomeScreen: Text Description automatically generated A picture containing text, clock Description automatically generated

Till now we have only implemented the View component. In the view section we are using view component. In prev lessons we have sent ajax request to backend to get list of products, but this time we are going to use redux to get things from backend and put them in redux state instead of component state.

So we dispatch an action in view part, and this action sends a request to redux store to make a change in the state of our application.

To make a state change we need to make an action and dispatch that action. In redux store we have 2 things- a state and a reducer

State: current state of our application

Reducer : is a function that takes current state make a change in state and return new state.

1st step to creating a redux is creating a store : store.js in src folder.

To create store: we need 2 things: 1 . Initial state 2. Reducer.

Text

Description automatically generated

To use redux store we need to go to index .js, which is the main entry point of our react application, here we wrap whole application inside provider and set store ad an property to this provider.

We need to add redux extensions to see redux store on inspect window.

This is what we get , when we add redux extension: No store found, to have reduc store we need to add some code in store.jsA screenshot of a computer

Description automatically generated

## Install redux thunk using npm install redux-thunk in frontend

Redux-thunk makes it possible to send ajax request to our redux actions, add it to your store