**Industrial Training Report/ Synopsis/ Minor Project**

**on**

Burger Ordering Web-app using React & Firebase

**A Project Report/Synopsis submitted in partial fulfilment of**

**the requirements for the award of**

**Bachelor of Technology**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**Navjot Singh**

**2019262**

**Under the supervision of**

**Ms. Ruchika Sharma**



**CGC – College Of Engineering**

Landran, Mohali. PIN-140307

**CANDIDATE’S DECLARATION**

I hereby declare that the work presented in this report entitled “Burger Ordering Web-app using React & Firebase”, in fulfillment of the requirement for the award of the degree Bachelor of Technology in Computer Science & Engineering, submitted in CSE Department, CGC - College of Engineering & affiliated to IK Gujral Punjab Technical University, Kapurthala, Punjab is an authentic record of my/our own work carried out during my degree under the guidance of Ms. Ruchika Sharma The work reported in this has not been submitted by me for award of any other degree or diploma.

Date : Navjot Singh

Place : 2019262

**CERTIFICATE**

This is to certify that the Project work entitled “Burger Ordering Web-app using React & Firebase”, submitted by Navjot Singh roll no. 2019262 , in fulfilment for the requirements of the award of Bachelor of Technology Degree in Computer Science & Engineering at CGC - College of Engineering, Landran, Mohali is an authentic work carried out by him/her under my supervision and guidance.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree.

Date : Ms. Ruchika Sharma

Place : Assistant Professor

**ACKNOWLEDGEMENT**

This project have been a result of sincere efforts. I am thankful to my mother for her unconditional support during the whole venture. I am also thanful to my mentor Ms. Ruchika Sharma for her valuable guidance, and taking out time from her busy schedule for the purpose of this project. Lastly, thanks are also due to my colleagues for providing various improvements and suggestions. This project would have not been possible without help of above mentioned persons.

Navjot Singh

**ABSTRACT**

The project have been made using React version 17.0.2 and firebase. React is not a stand-alone tech but is built upon JavaScript, and uses HTML, CSS and JavaScript in order to make components. Firebase has been used in backend. The app focuses on making a single page-web app which is capable of taking orders from the user, store it in the backend, and generate all the data from the back-end to the same app. All this will happen without the user reloading the web-app which makes it very convenient for the user in terms of speed. The orders placed by the user will be shown by the app dynamically, using graphical elements, both before and after the placement of the order. Before

the order is placed, the price and the order summary is shown and then the user proceeds to payment. Also, the details of the user are also recorded with validation, and this is also stored in the database.

**TABLE OF CONTENTS**

**..............................................................................................................................................................**

Student’s declaration............................................................................................................... i

Certificate by the guide........................................................................................................... ii

Acknowledgement................................................................................................................... iii

Abstract.................................................................................................................................... iv

Chapter 1 – Introduction and Installation of various tech-stack required..................................

1.1 Introduction to HTML..................................................................................

1.2 Introduction to CSS......................................................................................

1.3 Introduction to JavaScript.............................................................................

1.4 Introduction to Node.js..................................................................................

1.5 Introduction to React.....................................................................................

1.6 Installation of React and related project packages using node package manager...........................................................................................................

1.7 Introduction to Firebase..................................................................................

Chapter 2 – Setting up the project.................................................................................................

2.1 Planning our app..............................................................................................

2.2 Setting up the project.......................................................................................

2.3 Understanding about the components..............................................................

Chapter 3 – Implementing the project...........................................................................................

3.1 Creating layout component..............................................................................

3.2 Adding dynamic ingredient component...........................................................

3.3 Adding propType validation.............................................................................

3.4 Calculating the ingredient Sum dynamically....................................................

3.5 Adding build control component......................................................................

3.6 Displaying and updating the burger price.........................................................

Chapter 4 – Adding more features..................................................................................................

4.1 Adding the order button.....................................................................................

4.2 Creating the order summary modal...................................................................

4.3 Implementing the backdrop...............................................................................

4.4 Fixing the bug : Adding the price to the order summary...............................

4.5 Adding a toolbar.............................................................................................

Chapter 5 – Working on Navigation, layout and responsive design...........................................

5.1 Adding a burger logo.....................................................................................

5.2 SideDrawer : Creating responsive design.....................................................

5.3 SideDrawer : Improving responsive design..................................................

5.4 Reusing the Backdrop component................................................................

Chapter 6 – A look at backend.....................................................................................................

6.1 Using the firebase...........................................................................................

6.2 Creating the Axios instance............................................................................

6.3 Handling errors...............................................................................................

6.4 Fetching data from the backend......................................................................

Chapter 7 – Adding routing..........................................................................................................

7.1 Setting up routes.............................................................................................

7.2 Navigating to the checkout Page....................................................................

7.3 Navigating to the contact data component.....................................................

7.4 Fetching Orders..............................................................................................

7.5 Outputting the orders......................................................................................

Chapter 8 – Form Validation.........................................................................................................

8.1 Dynamically create Inputs...............................................................................

8.2 Handling form submission...............................................................................

8.3 Adding Validation Feedback............................................................................

Chaper 9 – Testing and deployment...............................................................................................

9.1 Deployment.......................................................................................................

**CHAPTER 1 : INTRODUCTION AND INSTALLATION OF VARIOUS TECH-STACK REQUIRED**

**1.1 Introduction to HTML**

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

**1.2 Introduction to CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

**1.3 Introduction to JavaScript**

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification.JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web.Over 97% of websites use it client-side for web page behavior,often incorporating third-party libraries.All major web browsers have a dedicated JavaScript engine to execute the code on the user's device.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

**1.4 Introduction to Node.js**

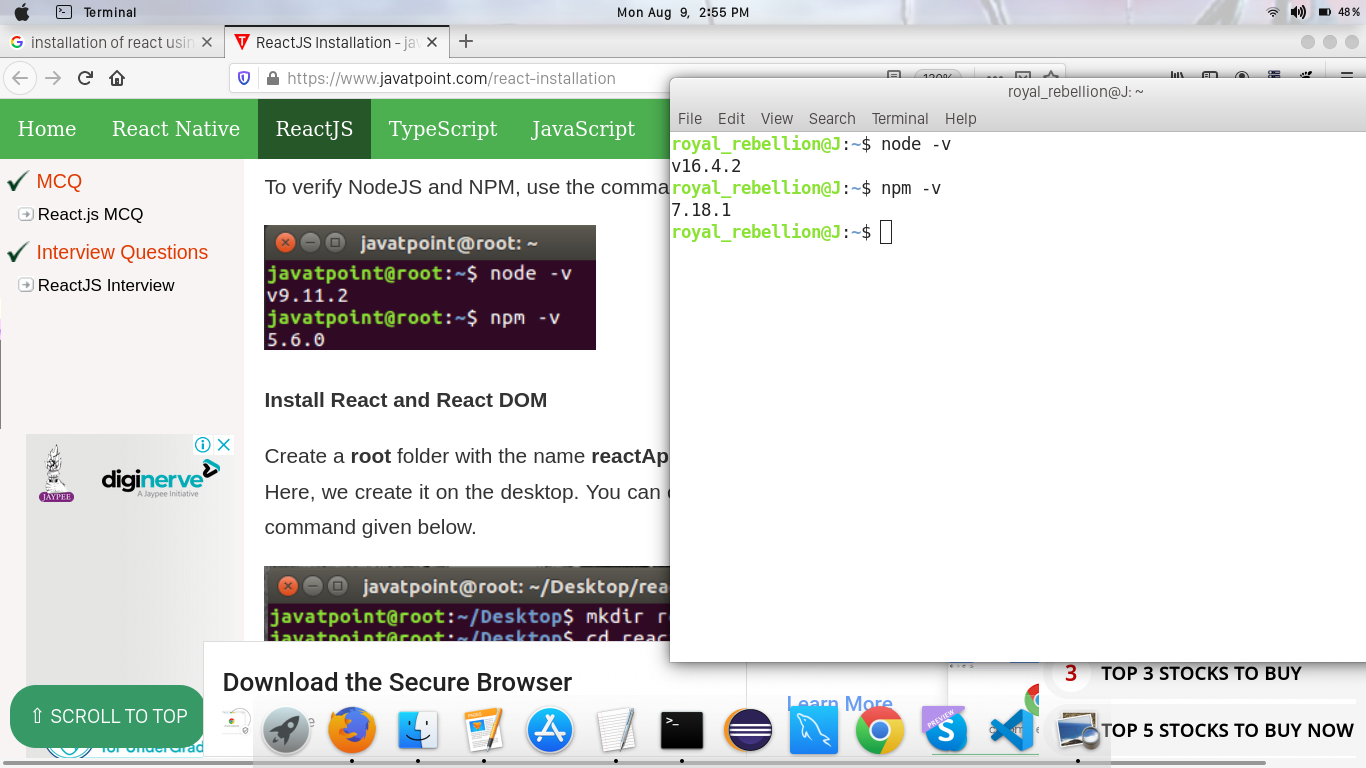
Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Though .js is the standard filename extension for JavaScript code, the name "Node.js" doesn't refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

**1.5 Introduction to React**

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

**1.6 Installation of React and related project packages using node package manager**

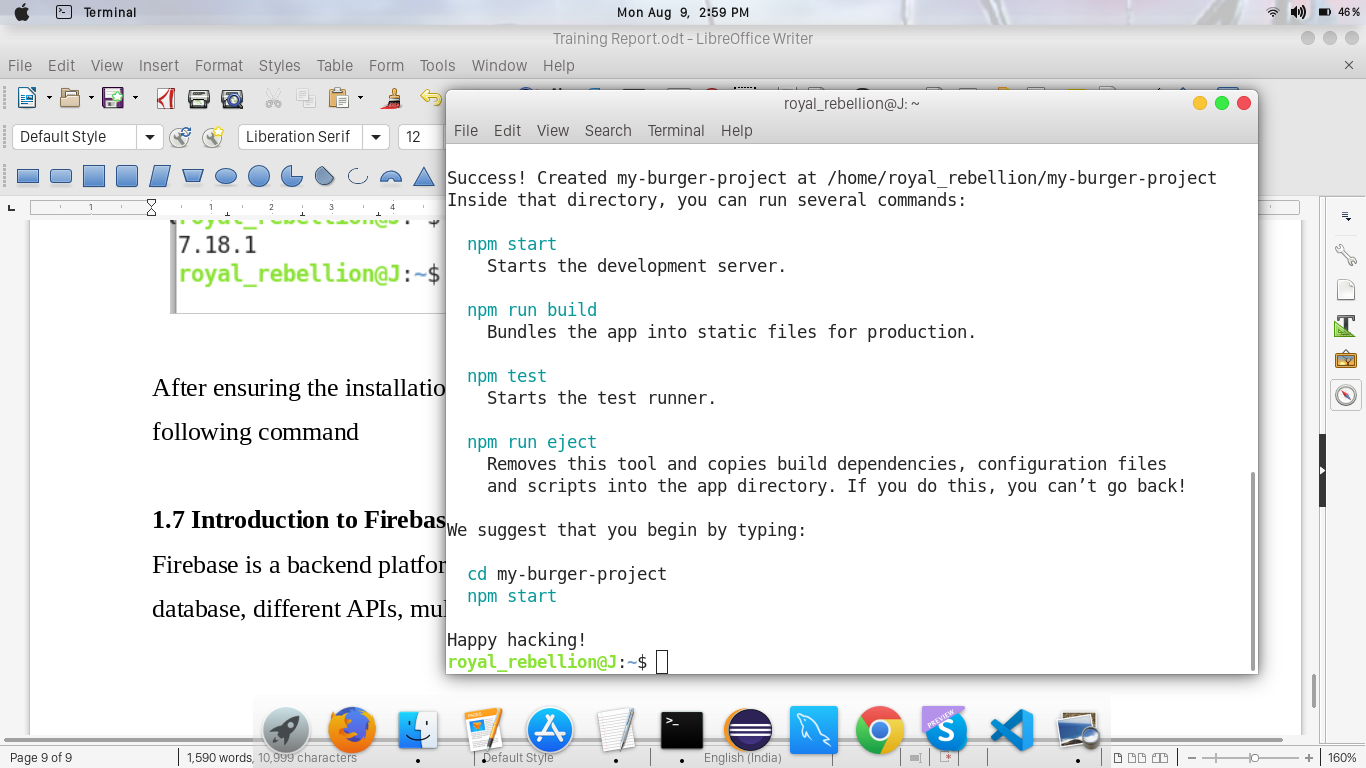
Prerequisites : Node.js and npm should be installed before-hand. They both are required to proceed to the installation of react. To confirm the installation of both type the following commands.

After ensuring the installation of both these projects, we need to install React by entering the following command

Now we can finally create our project with the help of freshly installed react using the following command



Upon complete installation, we will observe the following prompt from the React.



**1.7 Introduction to Firebase**

Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development.

Firebase is a backend platform for building Web, Android and IOS applications. It offers real time database, different APIs, multiple authentication types and hosting platform.

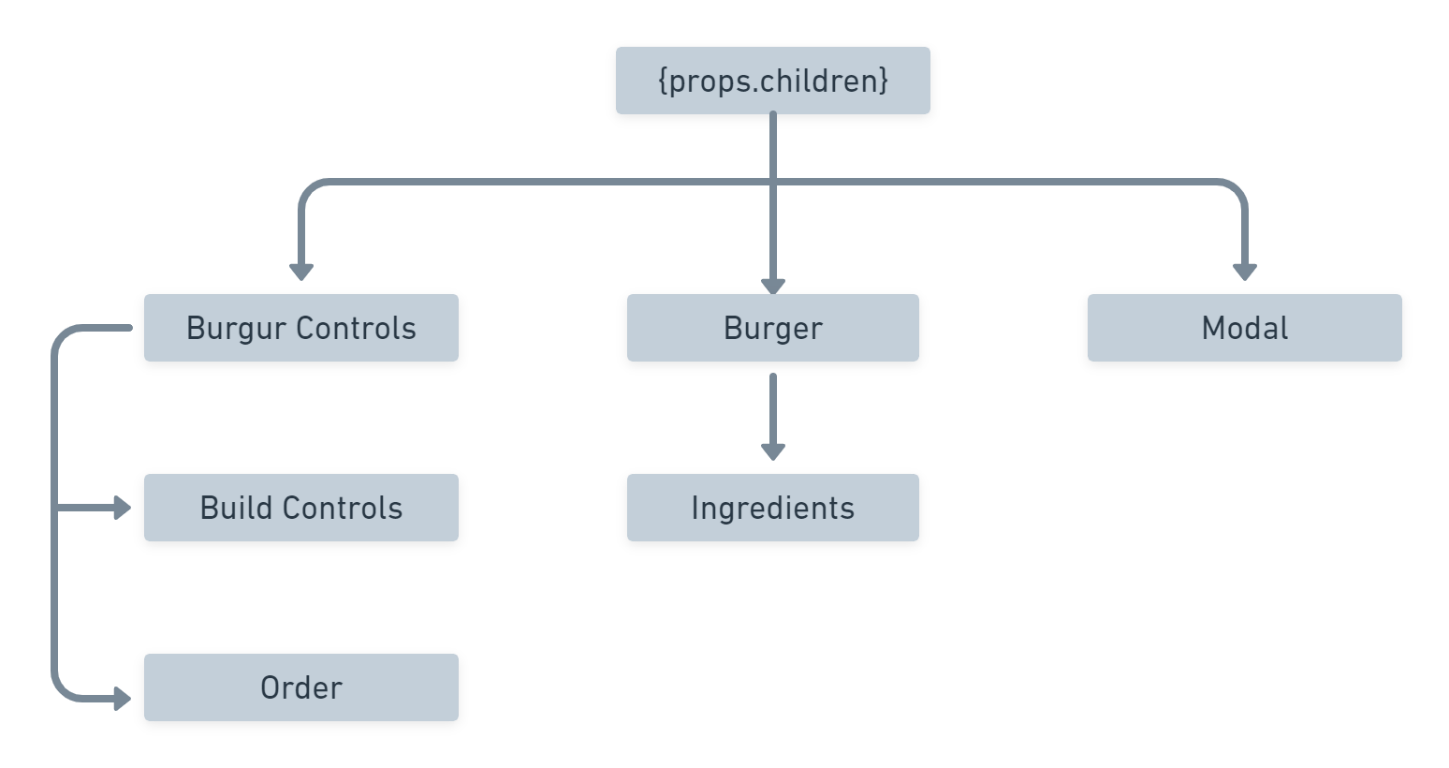
**CHAPTER 2 : SETTING UP THE PROJECT**

**2.1 Planning our app**

The overall layout of the app will be as follows :



Furthermore, the component {props.children} will be used to dynamically generate web-pages and give our app a multi-page application feel. It’s structure will be as follows



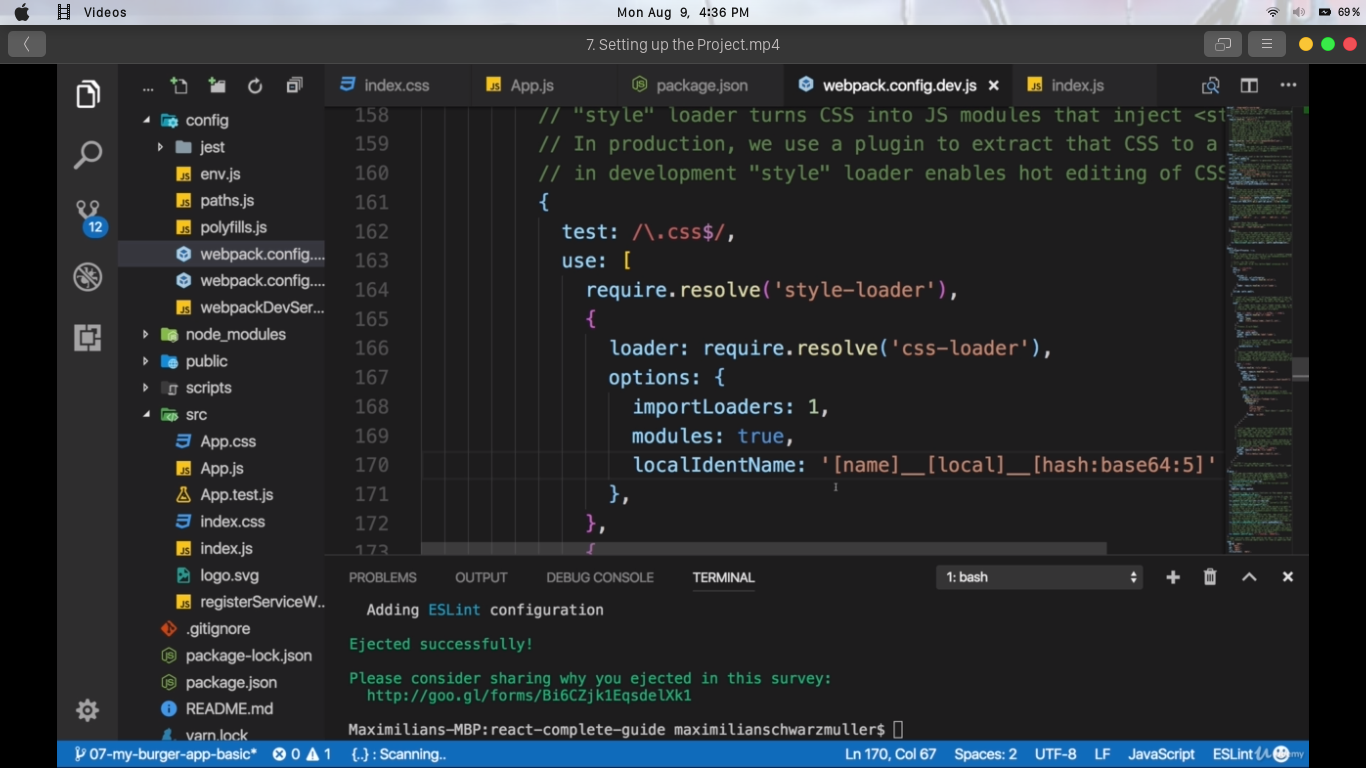
Here, the Toolbar will be static Sidedrawer is for responsive design and backdrop is to add functionality. Modal will be responsible for the order summary while Burger will represent the placed order in graphical way. Finally, BurgerControls will provide will the buttons and funtionalities for placing our order.

All the above mentioned parts will be react components.

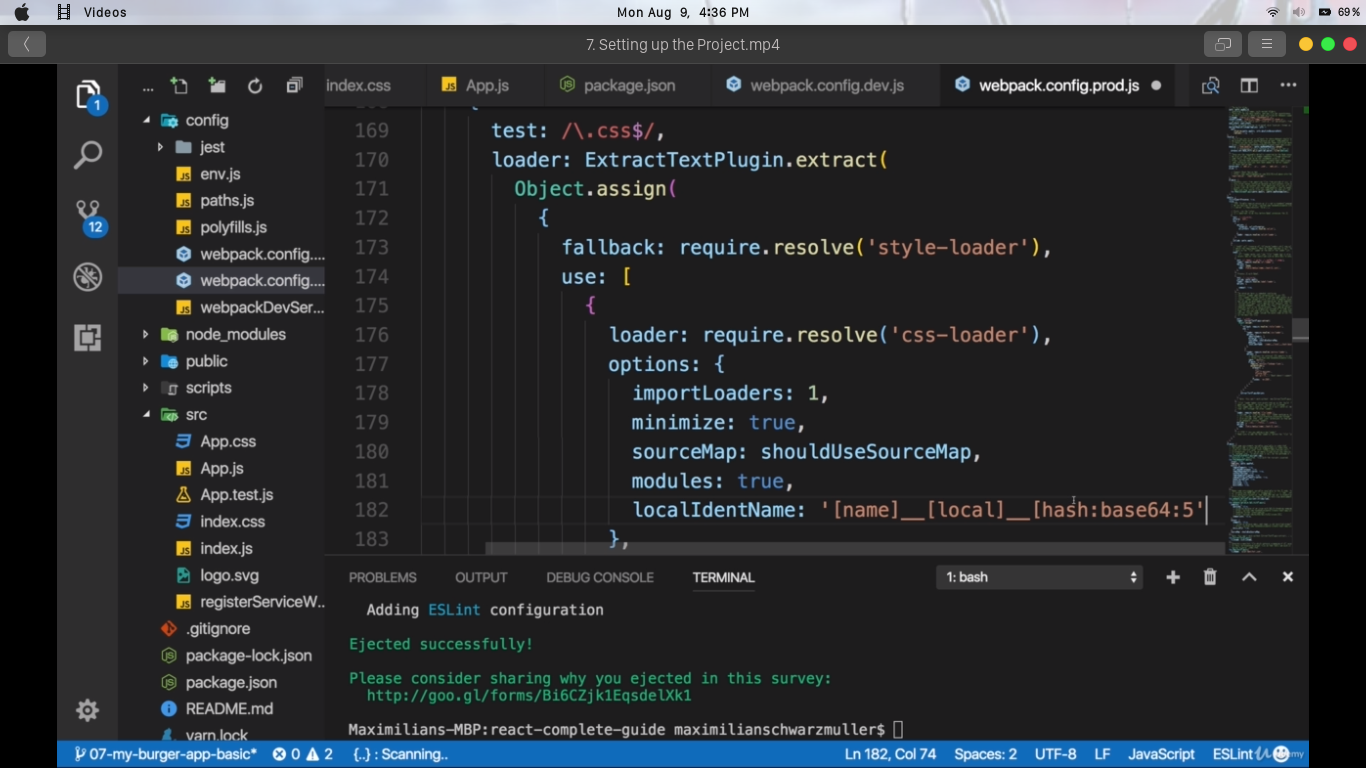
**2.2 Setting up the project**

Now since we have our layout ready and project created, we need to start building our project but before that we must ensure to set up the files properly. For that we need to ammend two in-built files namely

webpack.config.dev.js



webpack.config.prod.js



This is done to ensure we do not encounter any problems while importing our own css files.

**2.3 Understanding about the components**

A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier. You can see a UI broken down into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be your final UI.

In React, we mainly have two types of components:

1. Functional Components: Functional components are simply javascript functions. We can create a functional component in React by writing a javascript function. These functions may or may not receive data as parameters, we will discuss this later in the tutorial.

2.Class Components: The class components are a little more complex than the functional components. The functional components are not aware of the other components in your program whereas the class components can work with each other. We can pass data from one class component to other class components. We can use JavaScript ES6 classes to create class-based components in React.

**CHAPTER 3 : IMPLEMENTATION**

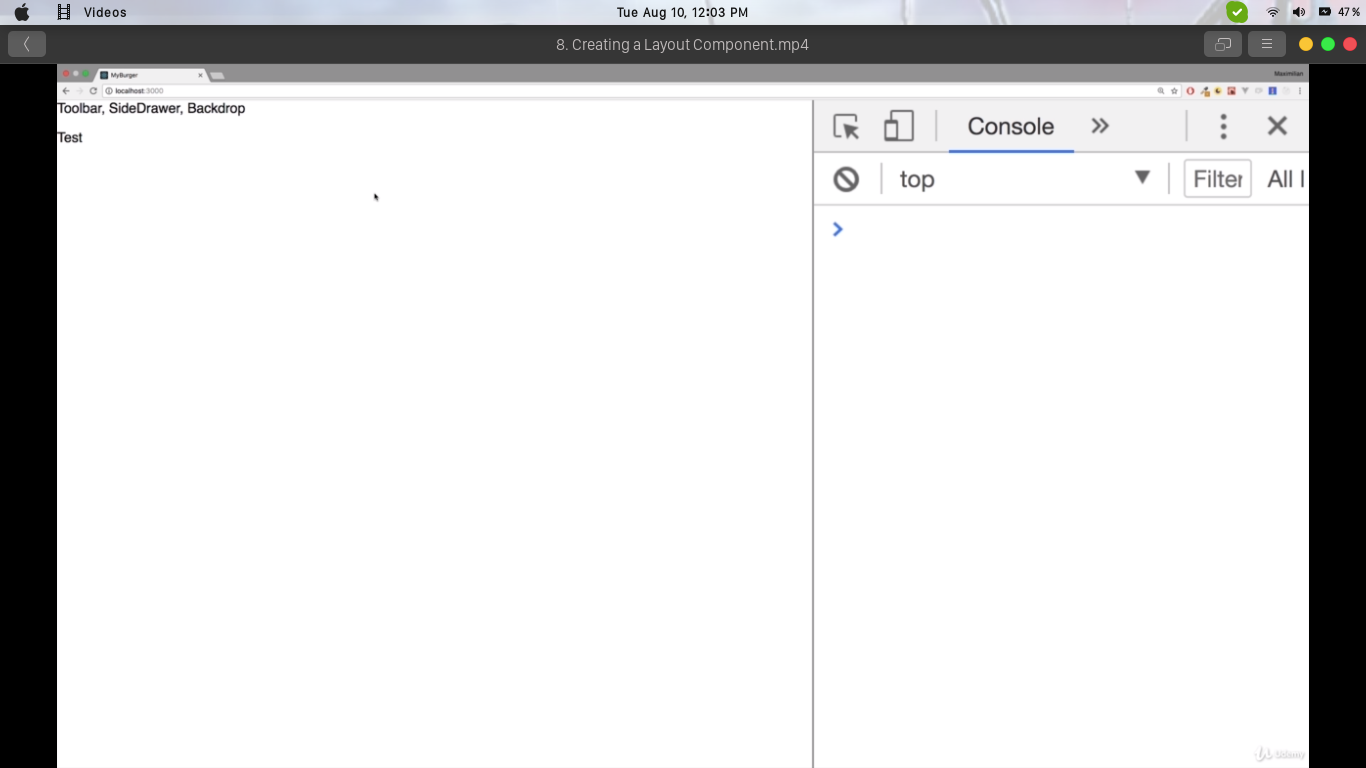
**3.1 Creating layout component**

First of all, will create that layout component though which will kind of be a wrapper implementing the toolbar and the burger builder then and later once we add routing and more pages to our application, we will take advantage of this layout where the toolbar will always be there but where we will then switch the page which is rendered in the content area of the layout so to say. Now we could simply turn the app.js file into a layout because it basically is the root component of our application and that would be a perfectly fine approach.Now this allows us to simply use this layout component as a wrapper around the core content component.

Now we want to have adjacent elements, that is why we will go with the higher order component approach and create such a utility auxiliary component. For that, create a new folder in the source folder, hoc for higher order component and there we will create an aux.js file.

Now the auxiliary component is really really simple, it's a functional component where we also won't manage any state with hooks and we will name it aux and here we just get props as an input and return props children, nothing else, no wrapping element.

We don't even have any JSX in this file which is also why we don't have to import React from React.



**3.2 Adding dynamic ingredient component**

It's now time to continue our work there and make sure that we do add the burger related components and especially the burger ingredients. For that, go to the components folder and create a new subfolder which we'll name burger, everything burger related shall go in there.

For example there, we will add the burger.js file which should be the burger we really are rendering to the screen. Now we can already say that the burger alone won't do the trick though, we need ingredients in that burger like bread, the salad, the meat, the cheese, the bacon, so for that, we'll create another subfolder here, the burger ingredient. And in there, we'll add a BurgerIngredient.js file and actually we want to start with that ingredient because well that is actually what we need to render something to the screen.

**3.3 Adding propType validation**

To add prop type validation and for this, run the command

*npm install --save prop-types*

This is the package, which is provided by the React team to validate the prop types of your React properties. Simply import prop types from prop types and you can start using that package.

Now we can add prop type validation by taking our class name, adding the prop types property with a lowercase.

Using the newly installed propTypes, we can make sure the type should be a string and we can chain another condition that it is required. So if we ever try to use the ingredient component without passing a type, we will get an error. So with that, the burger ingredient is finished.

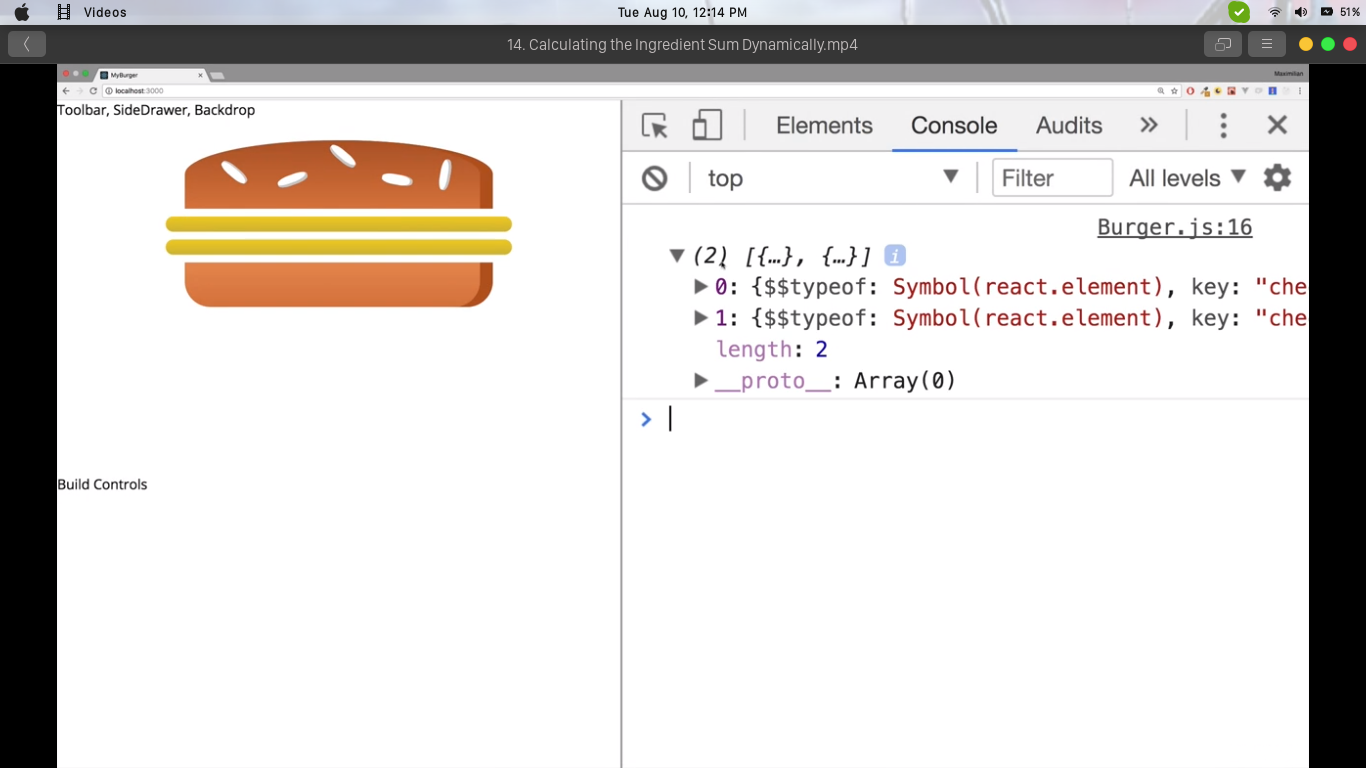
**3.4 Calculating the ingredient Sum dynamically**

Now with that, a first important step was taken, what happens now if we actually have a burger which starts with all ingredients set to 0 though. Now we see that but that's perfectly fine, we have no ingredients to render, we don't get an error because we just better if we could show some message like please start adding ingredients or something like that.

Now to show something in case we have no ingredients, we need to find out if we got no ingredients. Now transformedIngredients is always going to be an array though, just an array of empty arrays in the end.

So with that, we got an array which is either empty or contains the jsx elements and since it might be empty, we can simply check if transformedIngredients length is equal to zero, if that's the case then we want to output a paragraph where we say we start adding elements.

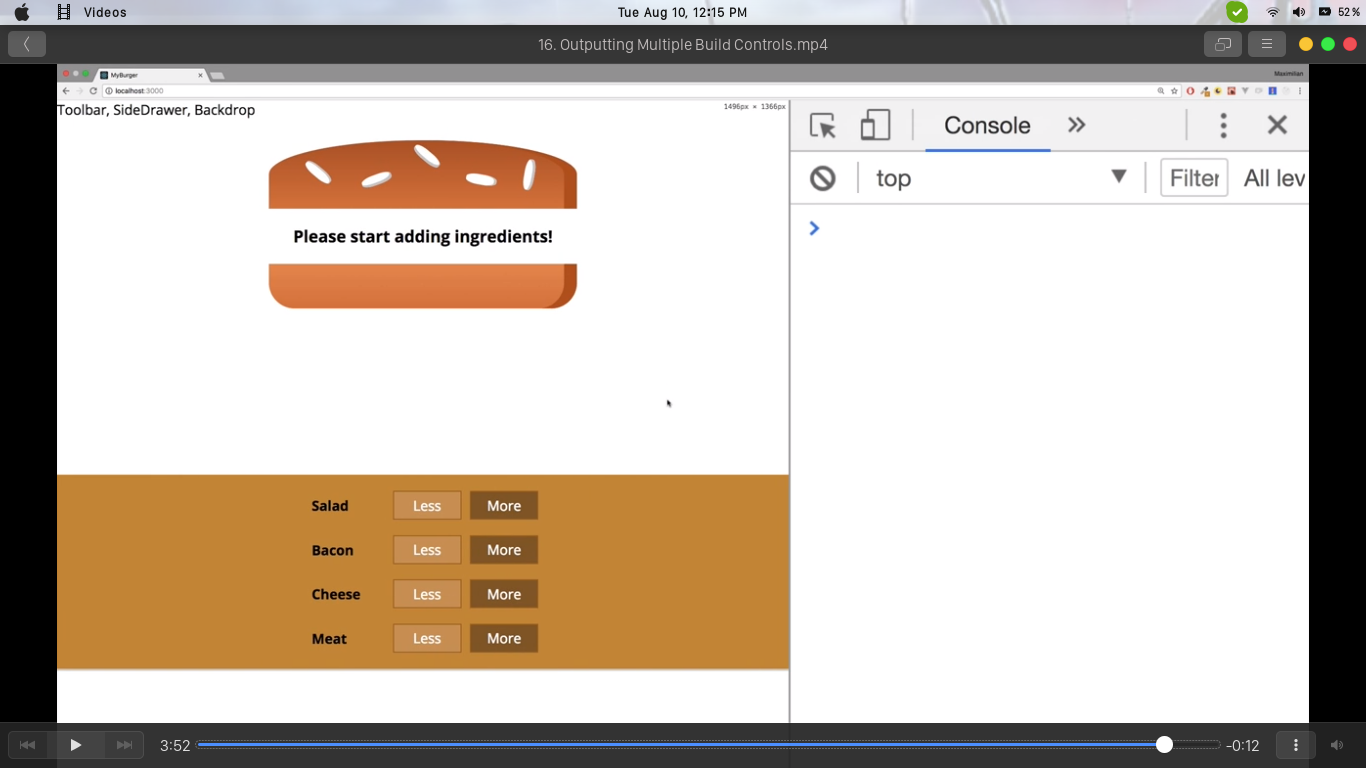
We will therefore convert this constant transformedIngredients to a variable which will be set to a paragraph when the ingredients array is of null type.



**3.5 Adding Build control Component**

Time to add some controls so that a user can actually control which ingredients a burger should have or should not have, and for that we need to add the build controls and we'll add this component in the burger folder, we'll add a new subfolder which all name build controls and there add a buildControls.js file.

Now the build controls will be a functional component so build controls is a function which receives props and which will return some jsx in the end.Therefore we of course also need to import react from react and we can then export the build control function as a default. In the end required CSS properties are set in the BuildControls.css file.



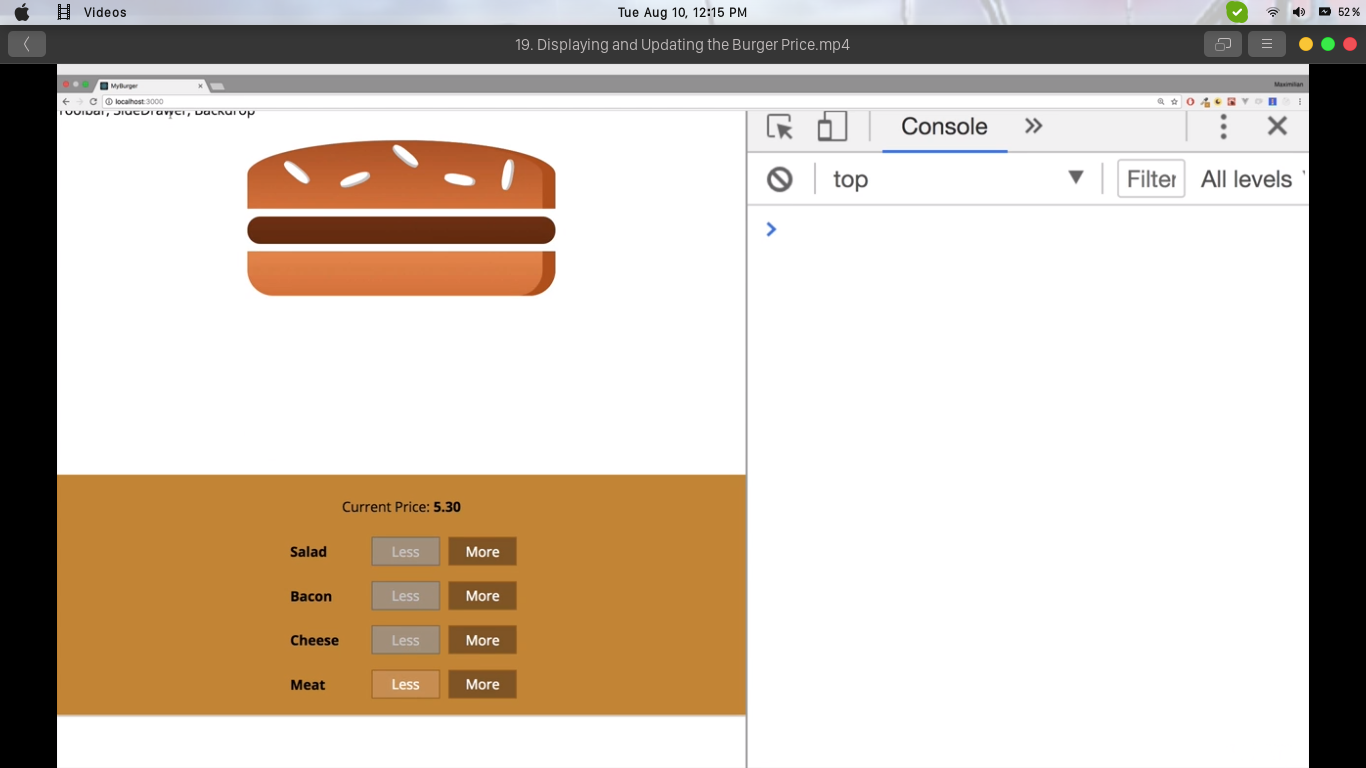
**3.6 Displaying and updating the burger price**

Uptill this point we made sure that we can also remove ingredients from the burger, now I want to update the burger price and display it as part of that burger too.

Now actually, if we add a lot of ingredients to a burger, it is scrollable because we configured it to be so the price should probably display it above the build controls, let's say.

So in the build controls here, besides outputting the controls,

we want to show the burger price and we can simply use a paragraph here where we say current price and then we expect to get the price passed in with props.

Now the price is calculated in the burger builder container,here we have the total price state. So we should pass this price to the build controls then because that is where we just added the usage of that property.

**CHAPTER 4 : ADDING MORE FEATURES**

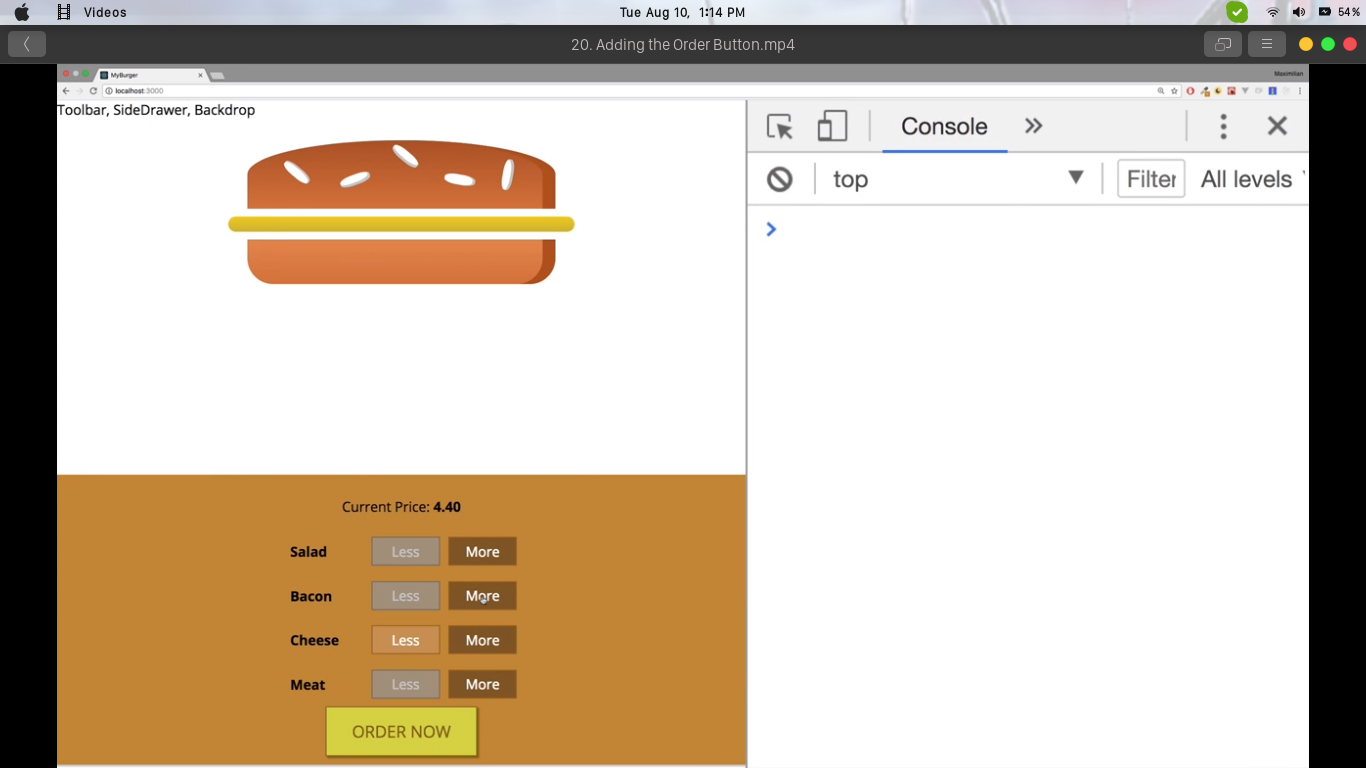
**4.1 Adding the order button**

We have our build control set up and can build a burger. Now we need to create a modal which will show us our order summary. This modal should be shown once we click checkout button which we also have to add below our build controls.

So we will add here a button which is a JSX object. The styling will be done in the corresponding css file.

Now to disable this button we need to check whether all ingredients have an amount of 0 or if at least one of the ingredients is at 1 which would make this button activated because then, we could of course click it. So this logic to check whether this button should be active so whether we can purchase the burger or not of course should managed in the burger builder container because that is where we have the state, that is where we know how much of each ingredient we want to add to this burger.

What we need to know here is the total amount of all ingredients so basically a sum of all the ingredients and we will simply add a new property where we will say purchasable and we will set this to false initially.Now purchasable should simply become true once we can buy this purchasable, once we can buy this burger so once at least one ingredient is one or more.

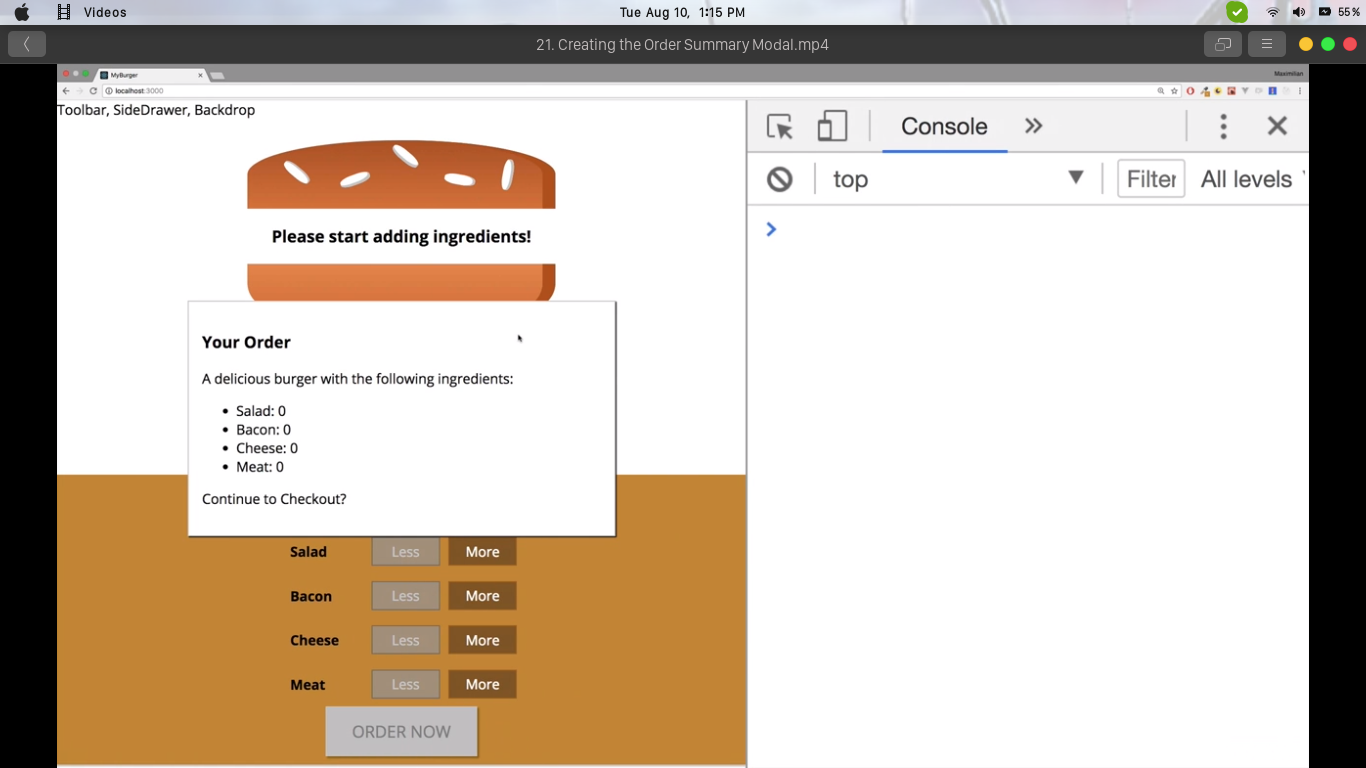


**4.2 Creating the order Summary modal**

We have added the order button. Now we want to make sure that once we click the button, we open this modal with the order summary. Now for that we obviously need a modal, we need a backdrop and we need to show some order summary, the idea behind the modal is to have a wrapping element which provides the styling which then simply wraps itself about any content we want to show in that modal, that makes sense to me at least.

Modal is going to be a functional component it doesn't have any state attached to it. The modal should simply be a div in the end which wraps itself about around any content,so a div which is wrapped around props.children and props.children really can be anything, can be our own components, can be some text, a paragraph, that is totally up to us how we use the modal and we can pass anything in there.Of course the wrapping div should receive some styling to well, to look good in the end, to have a modal look.

Once the modal is done, we need to import it in the burger builder component by navigating to that modal file which we can find in components.

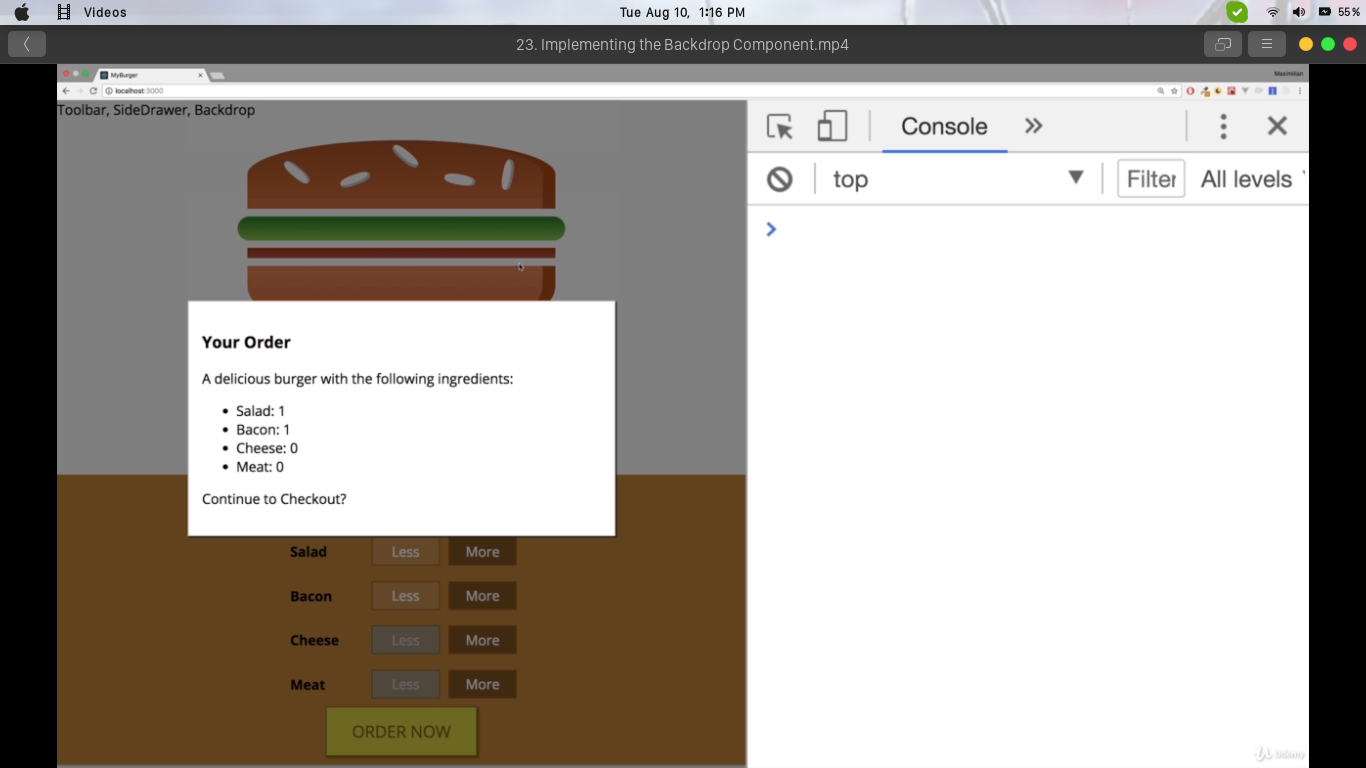


**4.3 Implementing the backdrop**

First of all create the backdrop component and add the styling for the backdrop. It's going to be a functional component which receives some props and return some JSX, it therefore also needs to import React from React and it needs to export itself as the default.

In the JSX code we just need to make sure if props show is true. If that is the case, again using a ternary expression here, we return a div, this div will simply hold the styling for the backdrop, otherwise I will return null.

Now we need to add the backdrop.css class so that we can use this through CSS modules and then backdrop should span the full height and width of the page to cover everything.



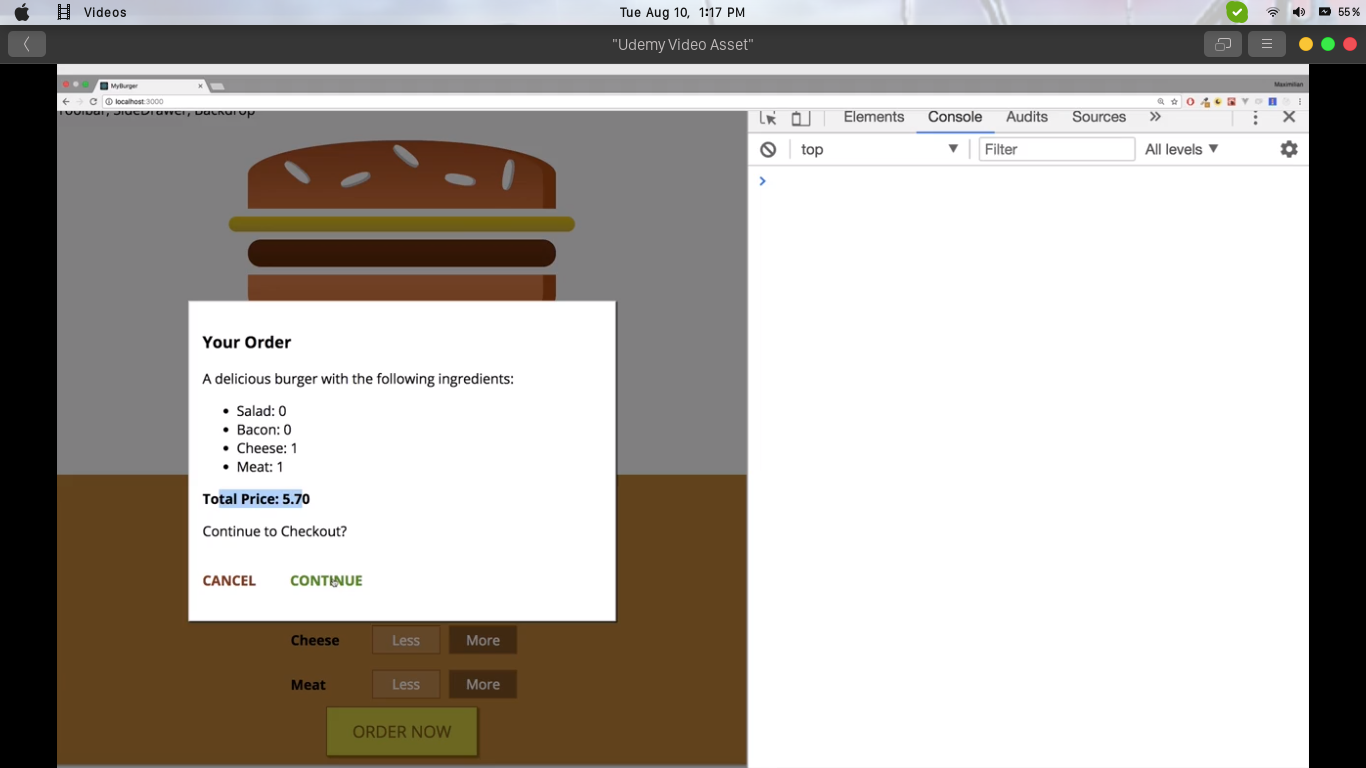
**4.4 Fixing the bug : Adding the price to the order summary**

We noticed though is if we build a burger and we order it, the total sum is missing in the order summary.

we have our orders summary component here and all we need to do is basically add some information below this ingredient summary we have here in the order summary component and there you could say total price.

We will simply accept the price property in order summary,that of course means that in the burger builder where we used the order summary, we now also have to pass the price as a property to that component, so simply access this state and there we have the total price.

With these changes, if we save both components we have a burger which we can now build an order and get the price too.



**4.5 Adding a toolbar**

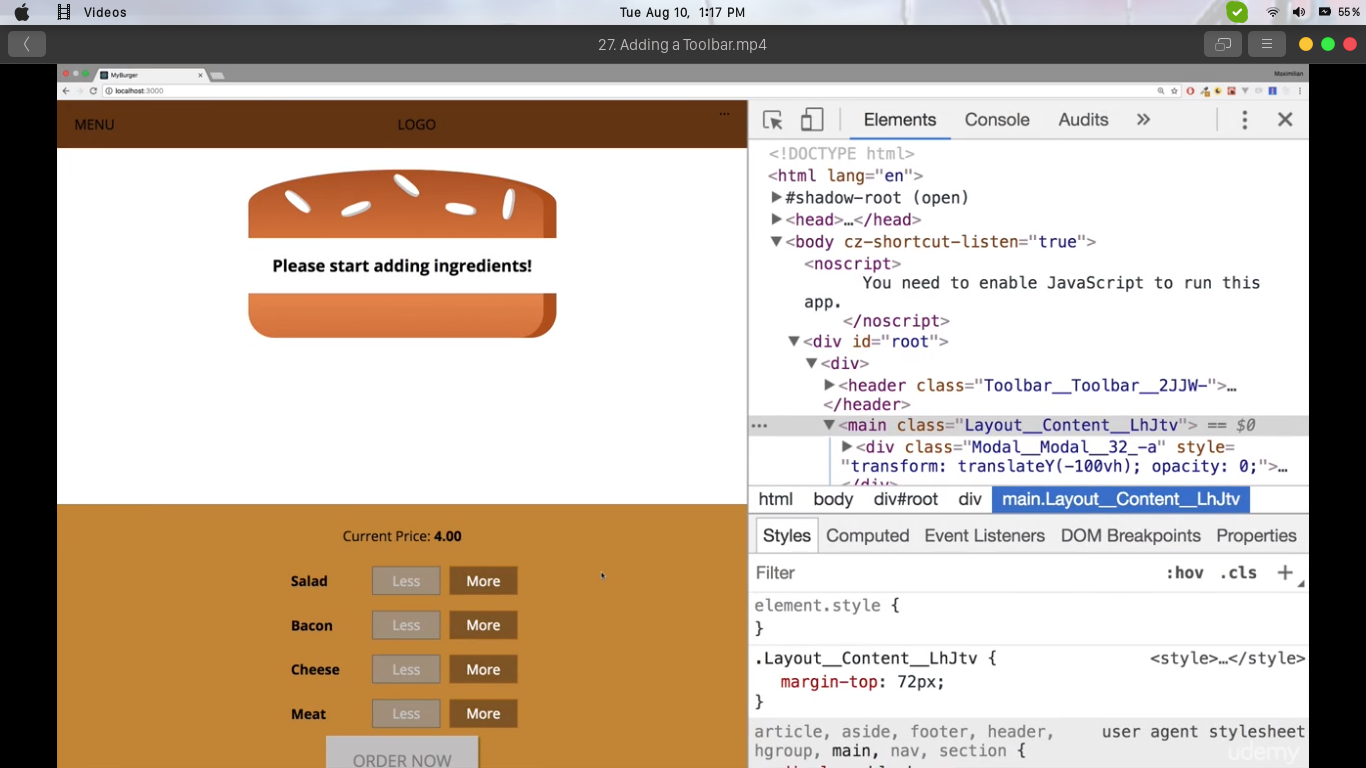
This of course is crucial because later once we add multiple pages, we need to have some way of navigating there and every application should probably have a toolbar.

Add a new folder in the components folder over there.

Call it navigation because we want to store all navigation related components like the toolbar, like the side drawer in there.

Add a new folder toolbar and in there toolbar.js, again a funcitonal component, it don't need to manage any state in there.

The styling properties will be placed in the corresponding css file.



**CHAPTER 5 : WORKING ON NAVIGATION, RESPONSIVE DESIGN AND LAYOUT**

**5.1 Adding a burger logo**

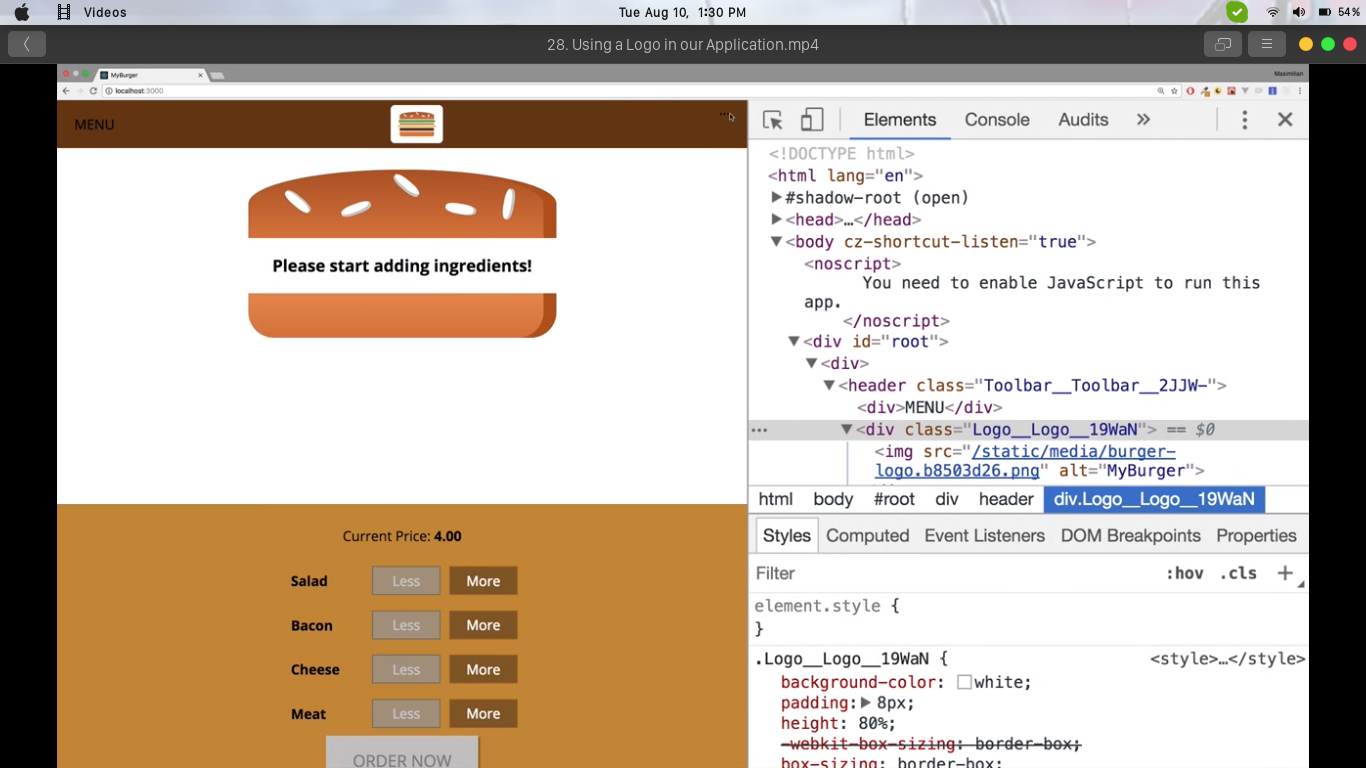
Uptil this point we have completed toolbar, let's now add a logo. Since we are going to use an image file as a logo so we are going to store that image file in the assets folder.

Now we will create a logo.js file which is again going ot be a funcitonal component.

We need to wrap it in a div which sets the background for the image though because the image itself has a transparent background.

Next we just need to refer to the image location (i.e. in the assets folder) in the src attribute of the image tag.

Once all is done, we just need to import the logo.js file in the toolbar.js



**5.2 SideDrawer : Creating responsive design**

It's time to add the navigation for mobile devices. For that we will add a new component in the navigation folder, the sideDrawer component in a sideDrawer folder.

It will consist of two components, the sideDrawer itself and the toggle button, that should be a button which opens the sideDrawer in the end. Also, we will need a backdrop for the sideDrawer but thankfully, we already got one so we'll reuse this component.

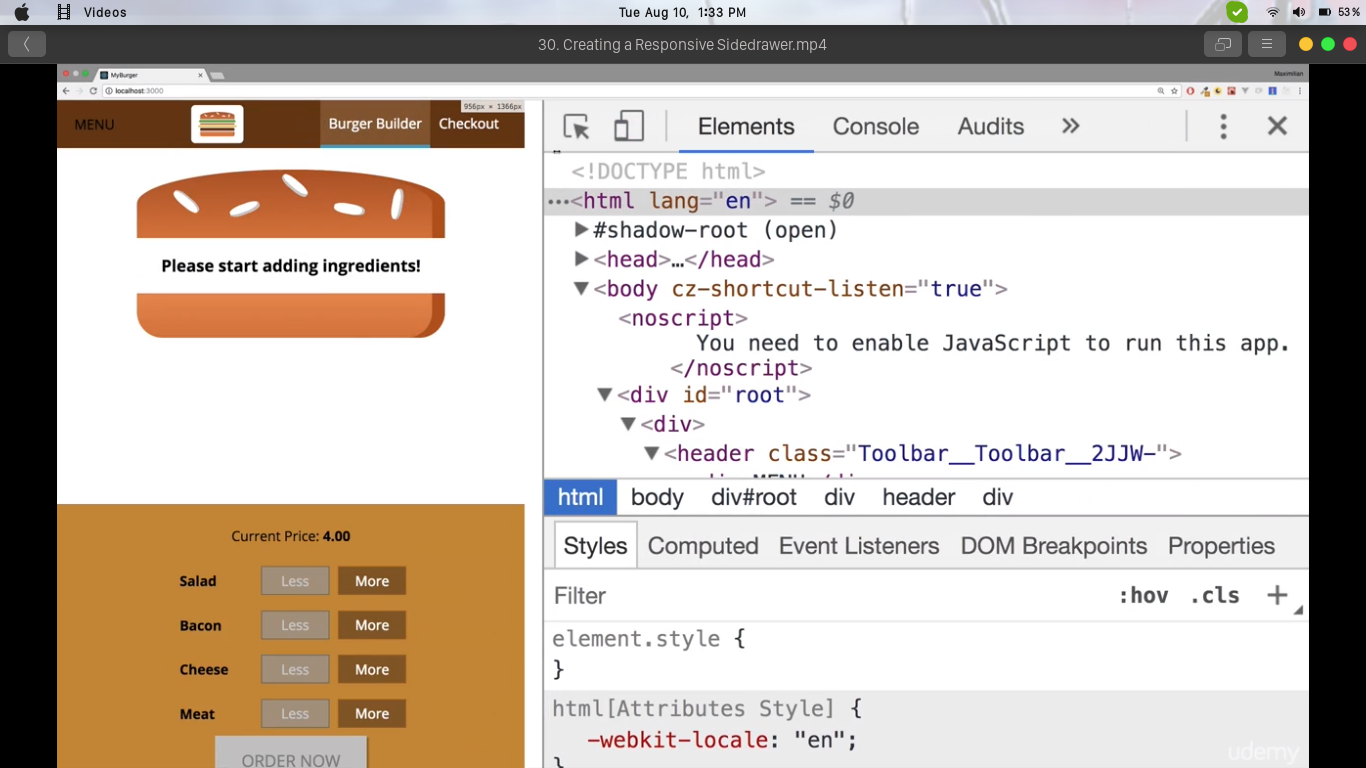
Let's work on sideDrawer now. The sideDrawer component is a normal functional component where we get some props and return something, that something will not be the jsx code immediately but a real function body we'll use here because before we return jsx in there, I want to do something here, we basically want to conditionally attach different css classes to make sure we play some animation when the drawer is shown.

So we'll add an open class which kind of slides it in and a close class which kind of slides it out

Now before we add any conditional classes, let's set up the sideDrawer and here simply use a div and in that div import both logo and navigation items.

Create a new css file the sideDrawer.css file. In there, add a sideDrawer class and the sideDrawer will only be visible on small device screens.

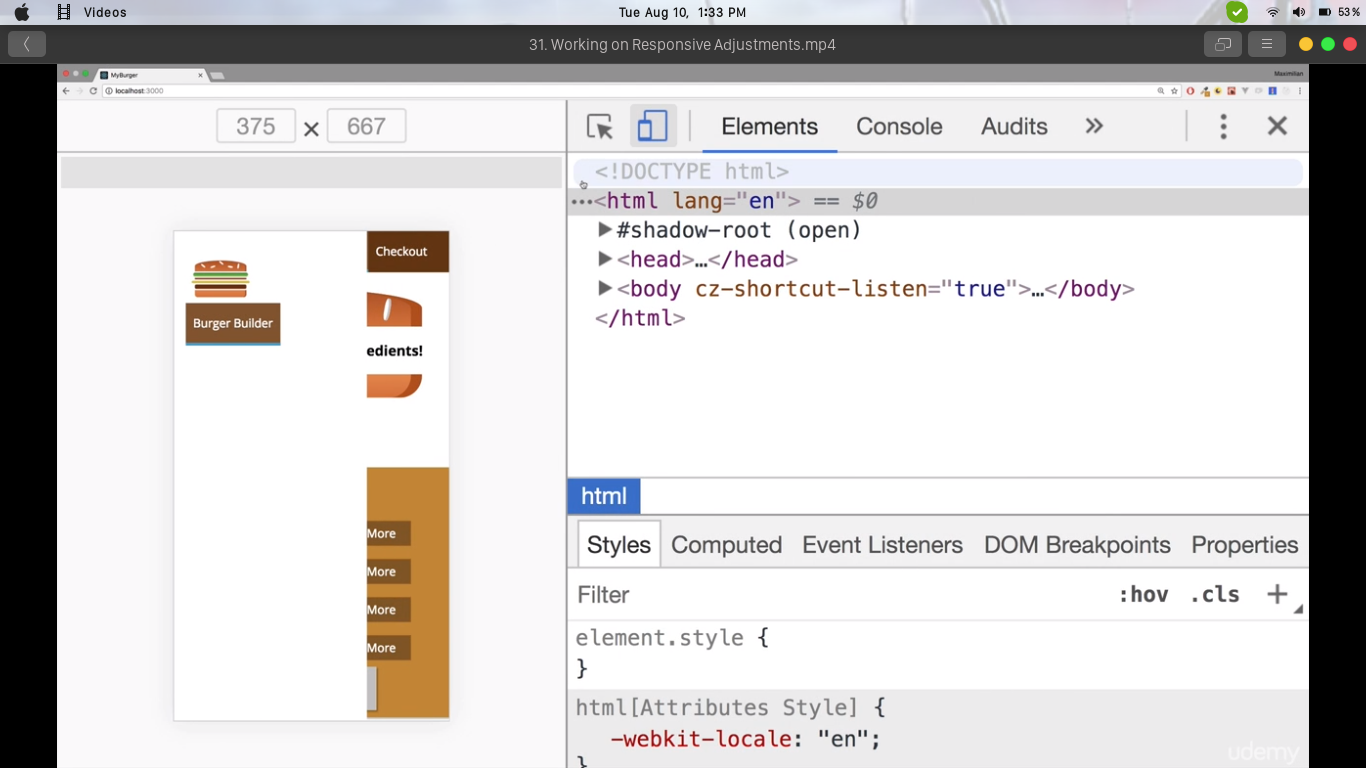
Using the media query we can make sure that we do set the screen size and the DOMs as per our requirement.



**5.3 SideDrawer : Improving responsive design**

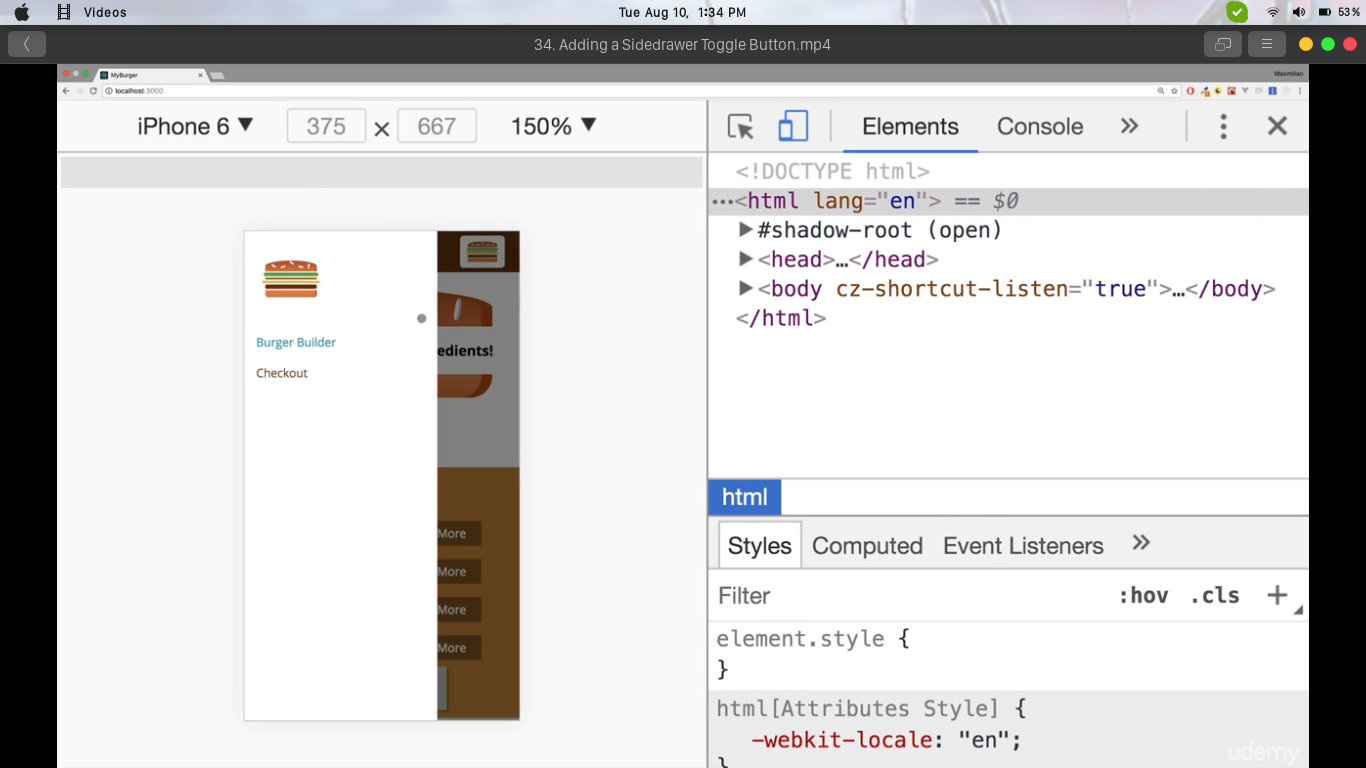
Till now, our side drawer is working fine but we are unable to get the elements as we wish to get them, simply due to the indifferences among various elements' size in mobile and web-app mode. Thus we run hit and try method to get the right mobile view application for our project.

After this module we are ensured of the safe portability of the web-app on the mobile screens.



**5.4 Reusing the Backdrop component**

Since our mobile view lacks the functionality of backdrop, so we will use the backdrop component we created earlier, and for that we will just import that component here in the sideDrawer.

****

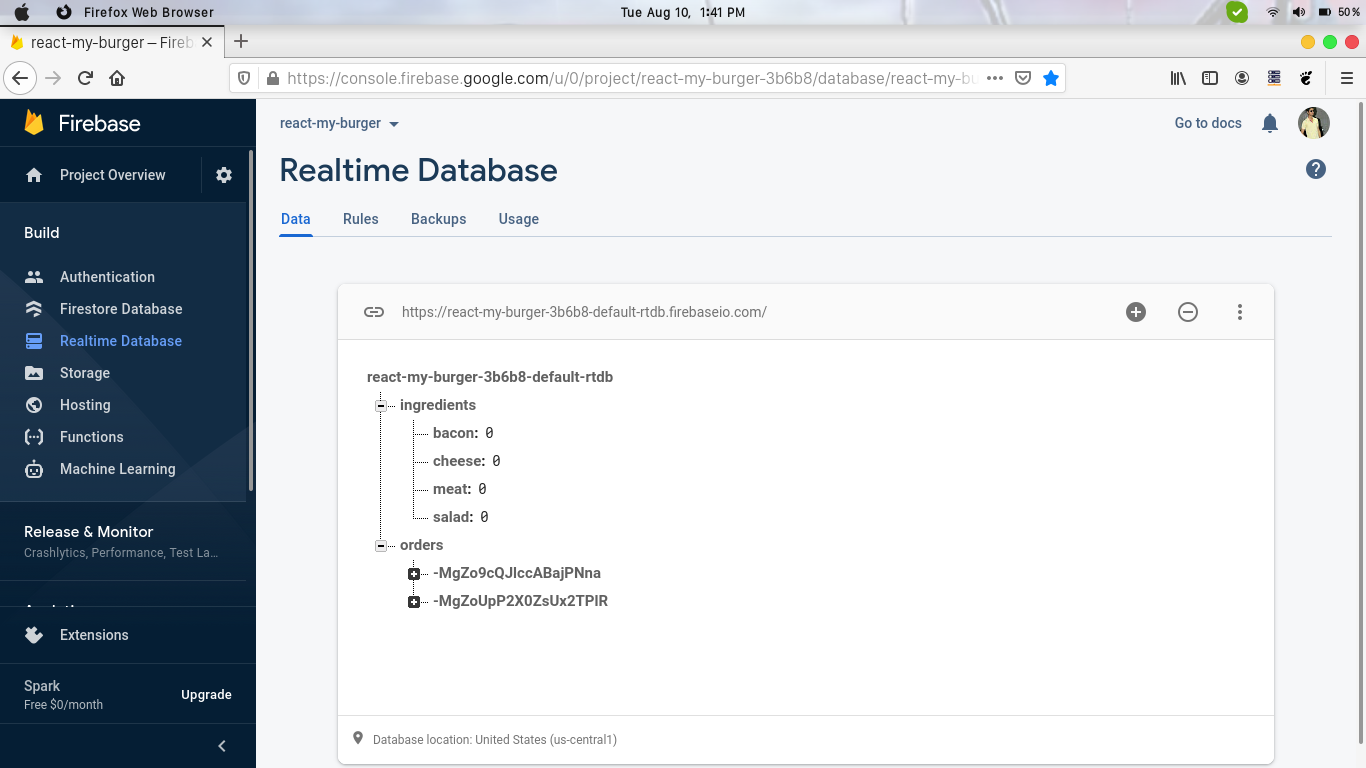
**CHAPTER 6 : A LOOK AT BACKEND**

**6.1 Using the firebase**

Firebase, backed by google, is just a dummy backend. We just need to have a google account.

Once we have successfully sign-up, we need to create a new project, by any name we want and then a corresponding link for that database will be generated. We will require to use this link to access the database and insert/fetch values into it.

And with that we're already ready to go We don't have to configure anything else we can get started and we can start sending HTTP requests to this endpoint or some nested endpoints as you going to see.



**6.2 Creating the Axios instance**

The axios instance is the one which will help us to populate the data into the data. And to use it we need to install a new package.

For that run the command

*npm install --save axios*

Once installed, we will use the axios to set our HTTP requests.

Import axios from the axios package and then call axios create and pass a javascript object to configure it.

Now in there set up the URL path generated from the firebase. This will act as our API.

After all set and done we have successfully created the axios instance.

**6.3 Handling errors**

For sake of simplicity, we set up some global error handler which shows a modal with the error message. To have a flexible way of showing an error, no matter which component, in which container it occurs, modal is going to get added to a higher level in the application so some higher component than this burger builder container where we catch errors and where we then make sure that our error modal is displayed.

In other words we simply want to create a higher order component with which I can wrap the burger builder and for that we are going to create my new higher order component in a folder which is going to be names withErrorHandler, which will contain the JS file of the same name.

withErrorHandler.js should be a function which takes the wrapped component as an input and which then returns a function which receives props and of course simply returns some JSX including the wrapped component.

Once the component is completed we just need to import it in the burger builder.

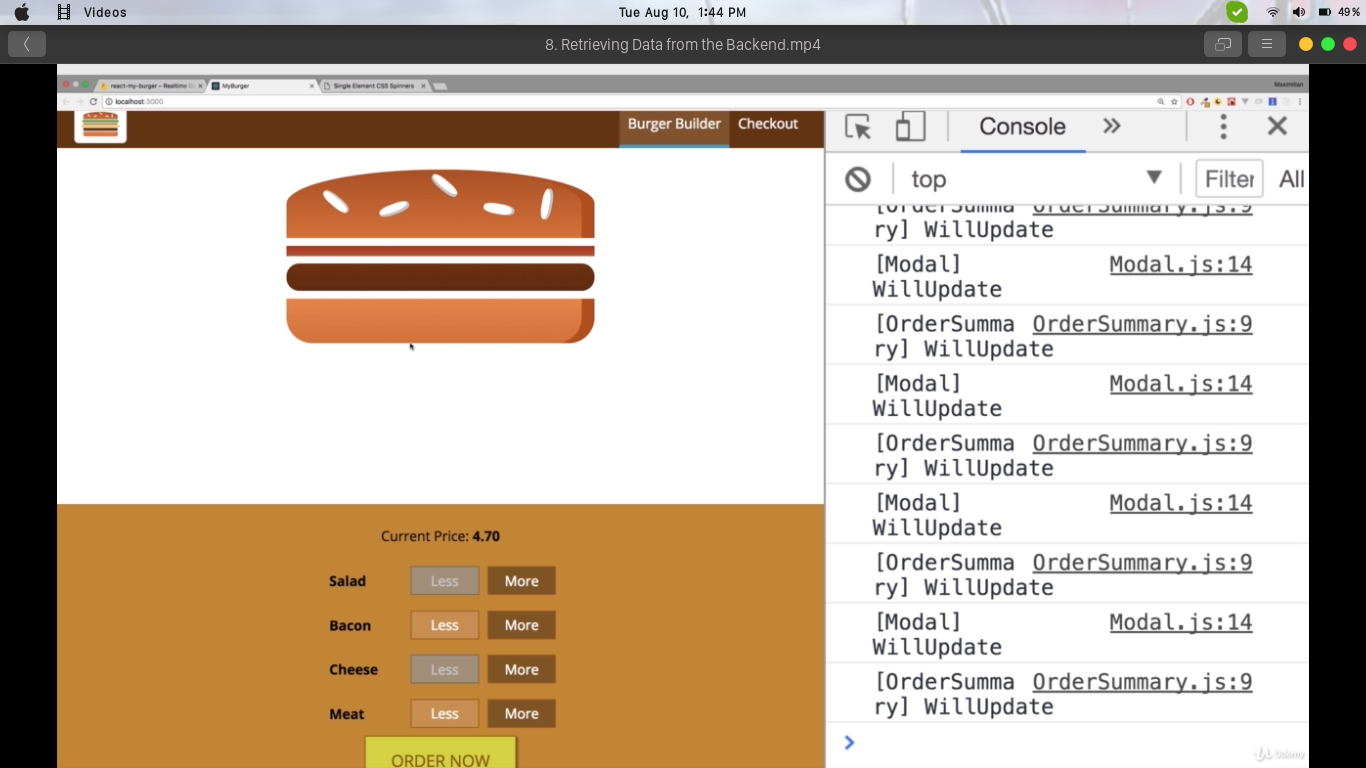
**6.4 Fetching data from the backend**

To begin fetching our orders from the backend, let us head to firebase and add a new node ingredient, with sub nodes of ingredient object used in React. Set their initial value to zero.

Now heading back to the burger builder, since we want to setup the state dynamically and for that a good place is componentDidMount.

We will add it here, componentDidMount. So we will be using axios and reach out to that URL we just copied with .json at the end.

This will now send a request to get our ingredients and as our answer to that request, will be the ingredients.



**CHAPTER 7 : ADDING ROUTING**

**7.1 Setting up routes**

To begin using routes, we need to install the react-router-dom, by typing the command

npm install --save react-router-dom

After installation, we can use react-router-dom by importing it from the package, as BrowserRouter.

We need to wrap everything where we want to use routing with the browser router and we will do that in the index.js file.

Now we can use routing in our project,

Now with the current set up, both would be loaded because path is treated as a prefix.

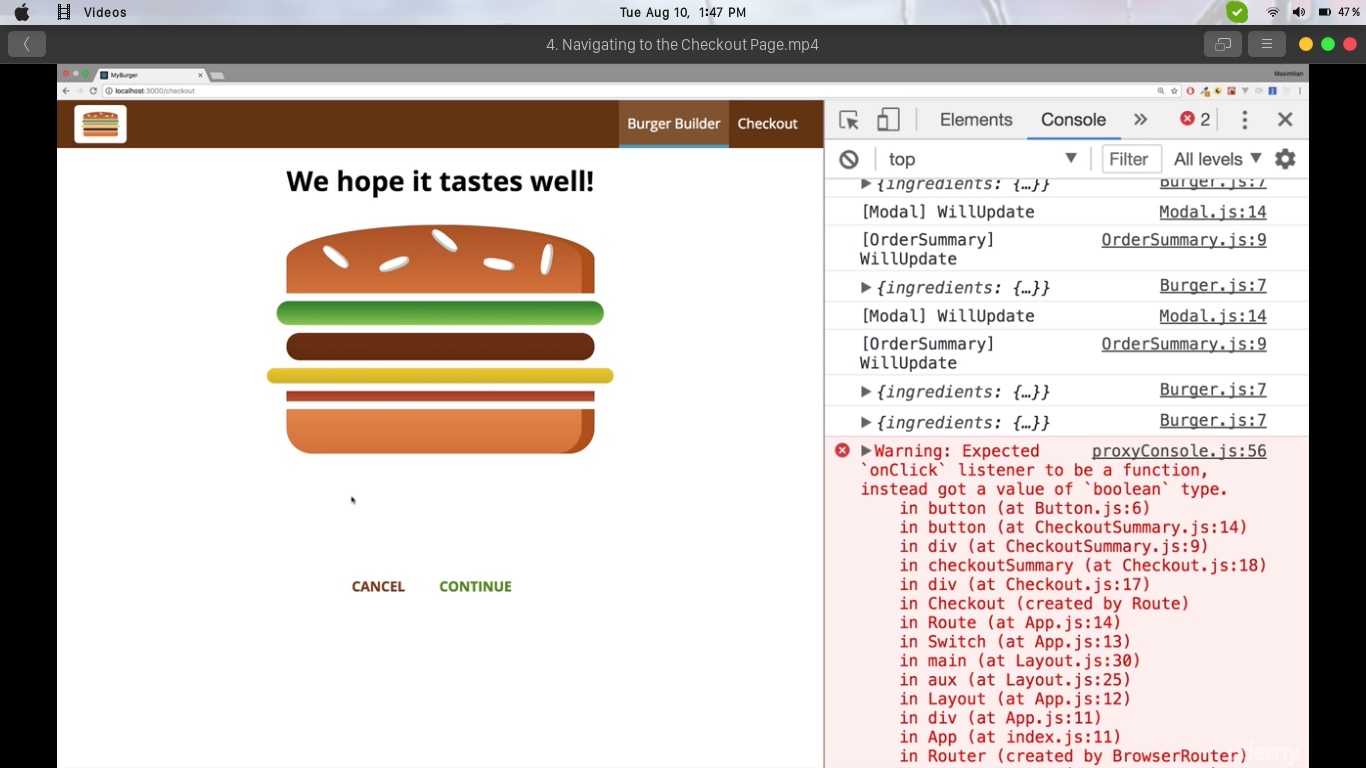
**7.2 Navigating to the checkout Page**

Now, since we have burger builder is part of the routable area of our project, we have access to the match location and history props.

We observe that Burger component is not loaded through a route object, only burger builder is. Burger builder is loaded through

this route and everything which is loaded through a route but only the direct components, only burger builder itself gets these special props. Components nested inside burger builder don't get them, we would have to pass them on manually.

We of course had access to these props, history is one of these special props provided by the router and there we have this push prop which allows us to basically switch the page and push a new page onto that stack of pages.



**7.3 Navigating to the contact data component**

Let's add the contactData component and actually that will also be a container, not only because it's loaded via our routing but because it will manage its own state especially once we covered forms in react. We will add it as a sub-folder and within it add contactdata.js file.

First we need to go inside of this contact data component, after which we can fetch whichever contact data we want and we will set up states to manage all these inputs.

This modules will just be taking the input from the user the normal Javascript way. Only diffrence in React will be that we will be fetching the data from the web-page, in the component.

The HTML DOM elements we will fetch data from will be coded the normal HTML way.

Once completed we just need to import it.

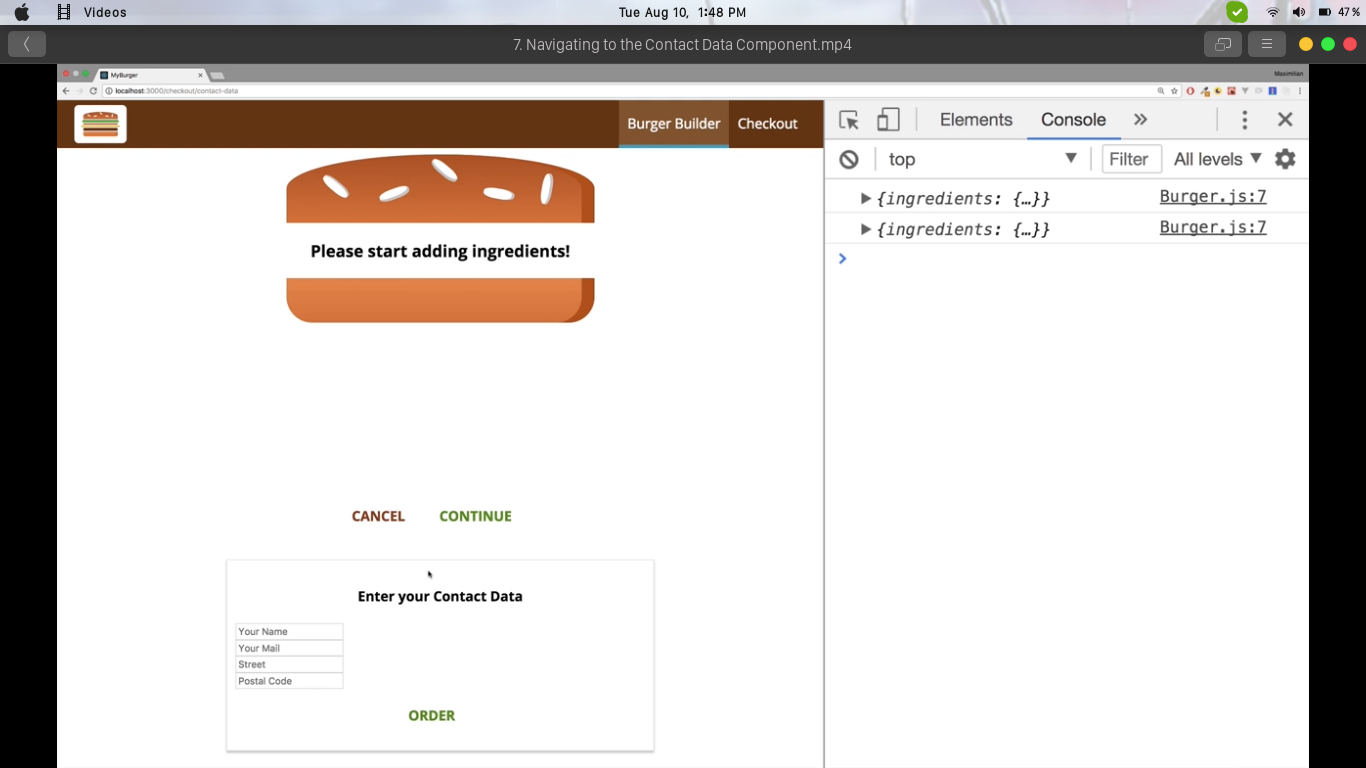
**7.4 Fetching Orders**

Now we have successfully submitted our data to the we just need to fetch it from the firebase. This time we will use it in our other react component and render it to the user.

We will edit orderSummary and provide CSS styling to it. We will also fetch the data of ingredients from the backend and thus every data we are going to show will be from the database.

Each order will have it's own card and we will dynamically assign the data to each one (after creating it) using the latest ESMA 6 script.

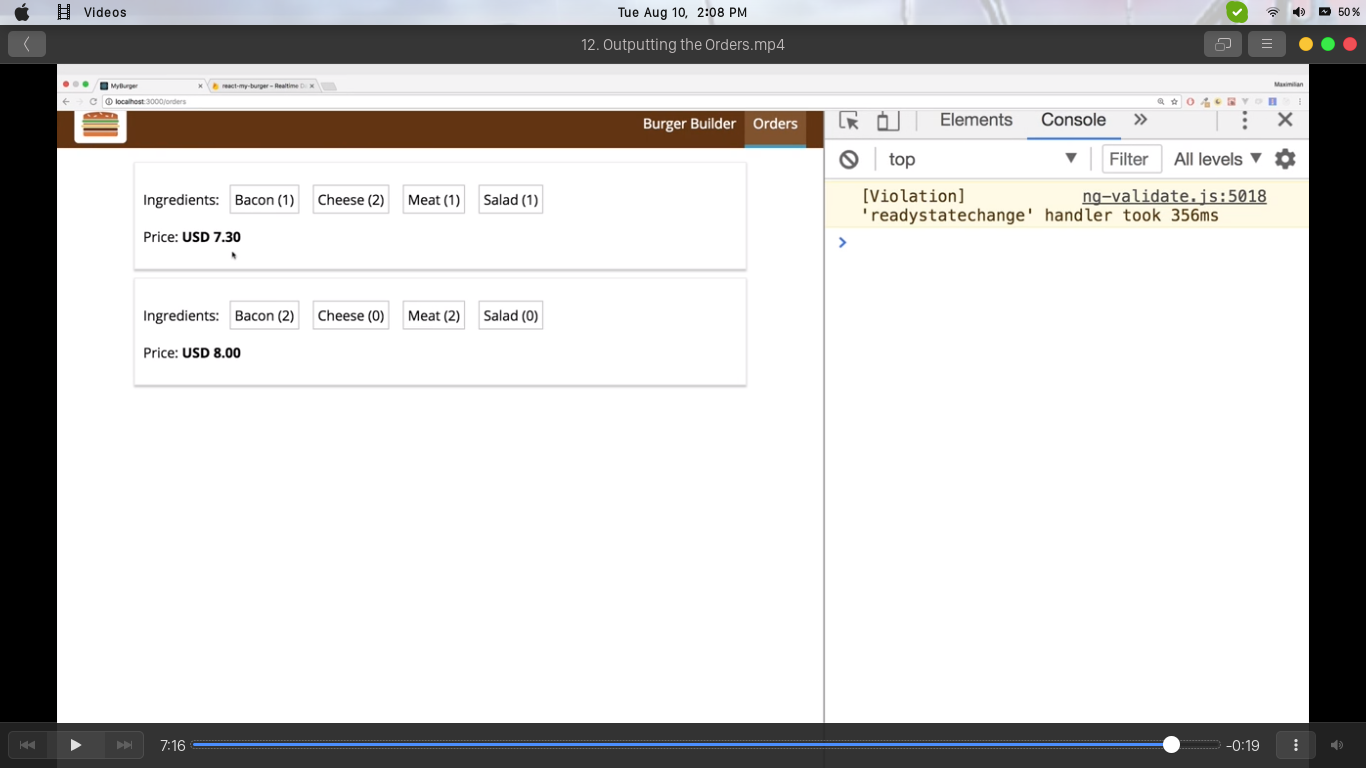
Since we are dealing with states we will need to use componentDidMount and to fetch data we will use axios.



**7.5 Outputting the orders**

Since we have implemented logic to fetch orders, we can now loop through them and output them. We will fetch each individual order and then render their components using the ESMA script.

The components rendered will have their own data, for example, corresponding to salad will show the quantity ordered and same for each ingredient.



**CHAPTER 8 : FORM VALIDATION**

**8.1 Dynamically create Inputs**

Uptil this point we defined our javascript object which should represent our form. Now we to output that in my render method here

we want to create all these inputs dynamically.

The corresponding changes we need to make will be made here, such that we would be able to use our self defined component for the purpose of getting input and modify it to our requirements.

For that we will need to first of all turn our order form object here into some kind of array we can oop through, an array where we basically have javascript objects where this key is just one property, an identifier property and then we also still have the other properties.

So with that information and let's create or push I a new object to the form elements array, this object should have an id which is the key, because again that is name and so on, these are identifiers and then it should have the config which is this.state.orderForm for a given key.

Now we can work with it by now creating our form here while looping through all these elements, and rendering them as required.

**8.2 Handling form submission**

We created a form which we simply set up in javascript then which we then handled dynamically with our own input component and our inputChangedHandler where we immutably update any effect that form elements, now let's make sure we can also submit this form.

For that, we have the button with the clicked handler.

onSubmit event handler we will use on the form itself, and there, we will now execute the method we used for that, the orderHandler onSubmit. Now in the orderHandler, we still get an event object here automatically passed by react.

Now we need to submit and the data is already managed in this state.

**8.3 Adding Validation Feedback**

To output feedback upon our validation result, we have to go to my input component and therefore each input element, we in the end want to add a special class to that element if it is valid.

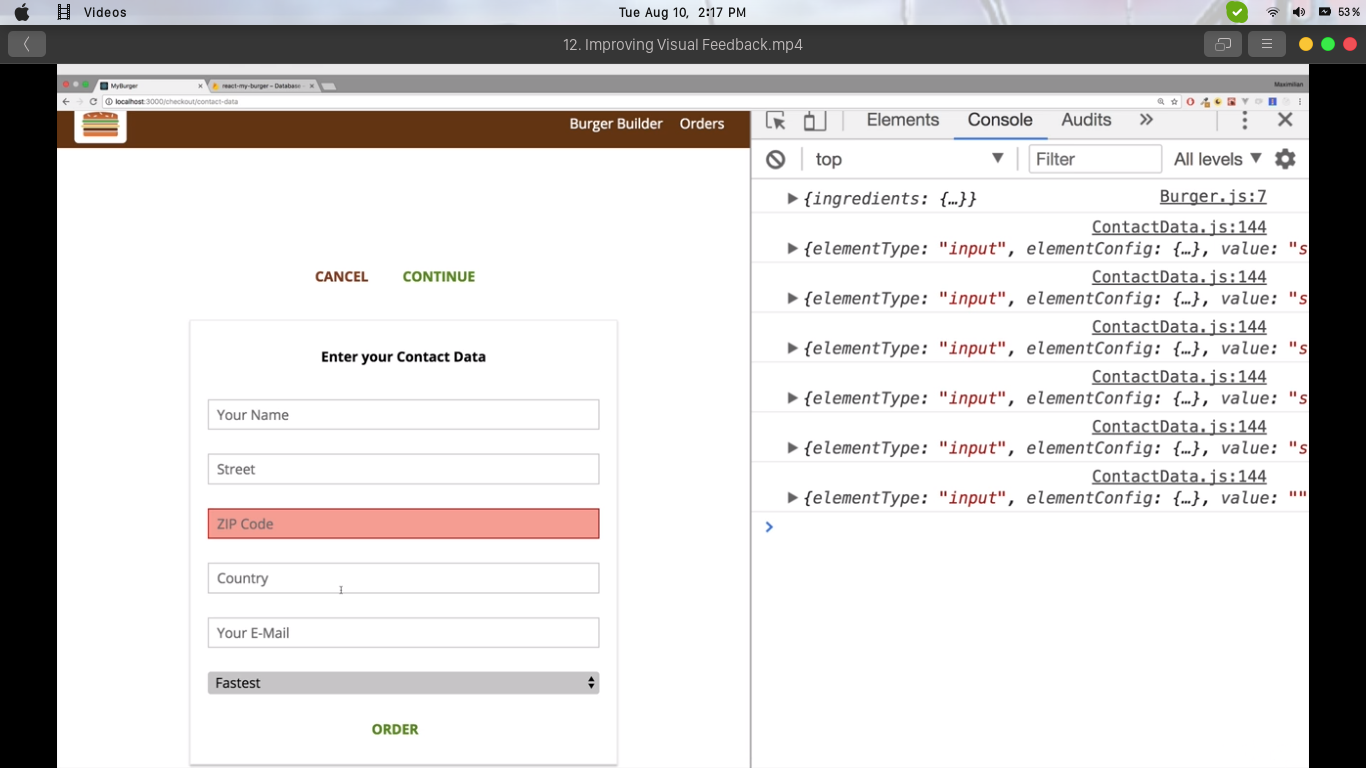
This originates our need to have our classes become more dynamic. For that we will add a new constant which we will name inputClasses which is an array which at the beginning is just classes.inputElement.

Now on the individual elements here, we'll replace classes.inputElement with our array, and join it using .join(' ') method of Javascript.

Now inputClasses also needs to receive the invalid property if the input is invalid and we'll do this with a if check, that check is independent of the type of the input so we only need to do it once at the beginning.

We'll check if props.invalid is set to true, if it is then we'll push a new class to our inputClasses.

For styling we will use the css file and import it here.



**CHAPTER 9 : DEPLOYMENT**

**9.1 Deployment**

We will run the command

*npm run build*

and this will give us a build folder having all the required files we need to deploy on our server. After running this command we will get our production ready file and our project will be ready to be put into use.

**REFERENCES**

Official react github : <https://github.com/facebook/react>

geeksforgeeks react tutorials : https://www.geeksforgeeks.org/reactjs-tutorials

tutorialspoint react tutorials : https://www.tutorialspoint.com/reactjs/index.htm

css spinner : https://projects.lukehaas.me/css-loaders