Burger Grills-India’s best Burger Restaurant

Created by: Soumyadipto Pal

Email: soumyadiptopal@gmail.com

Purpose of this project:

An online website that sells burgers. It suggests users to not only customise their burgers by adding or removing ingredients but also create their own burgers by adding ingredients in stock.

Technologies used:

1. Hyper Text Mark-up Language (HTML)
2. Cascading Style Sheet (CSS)
3. React JavaScript
4. Express
5. MongoDB

Getting Started:

Download Nodejs from the official website- <https://nodejs.org/en/> and install it.

Download MongoDB community edition from the official website- <https://www.mongodb.com/try/download/community> and install it.

Then follow this 3 steps to start the website:

1. Go to backend folder, open the terminal here, type npm init. Once the node-modules are downloaded, type node .\index.js. this will start the backend at port 5000.
2. Go to User folder, open the terminal here, type npm init. Once the node-modules have been downloaded, type npm start. this will start the customer interface at port 3000.
3. Go to admin folder, open the terminal here, type npm init. Once the node-modules have been downloaded, type npm start. this will start the customer interface at port 3006.

Structure of this Website:

Data Base

Backend

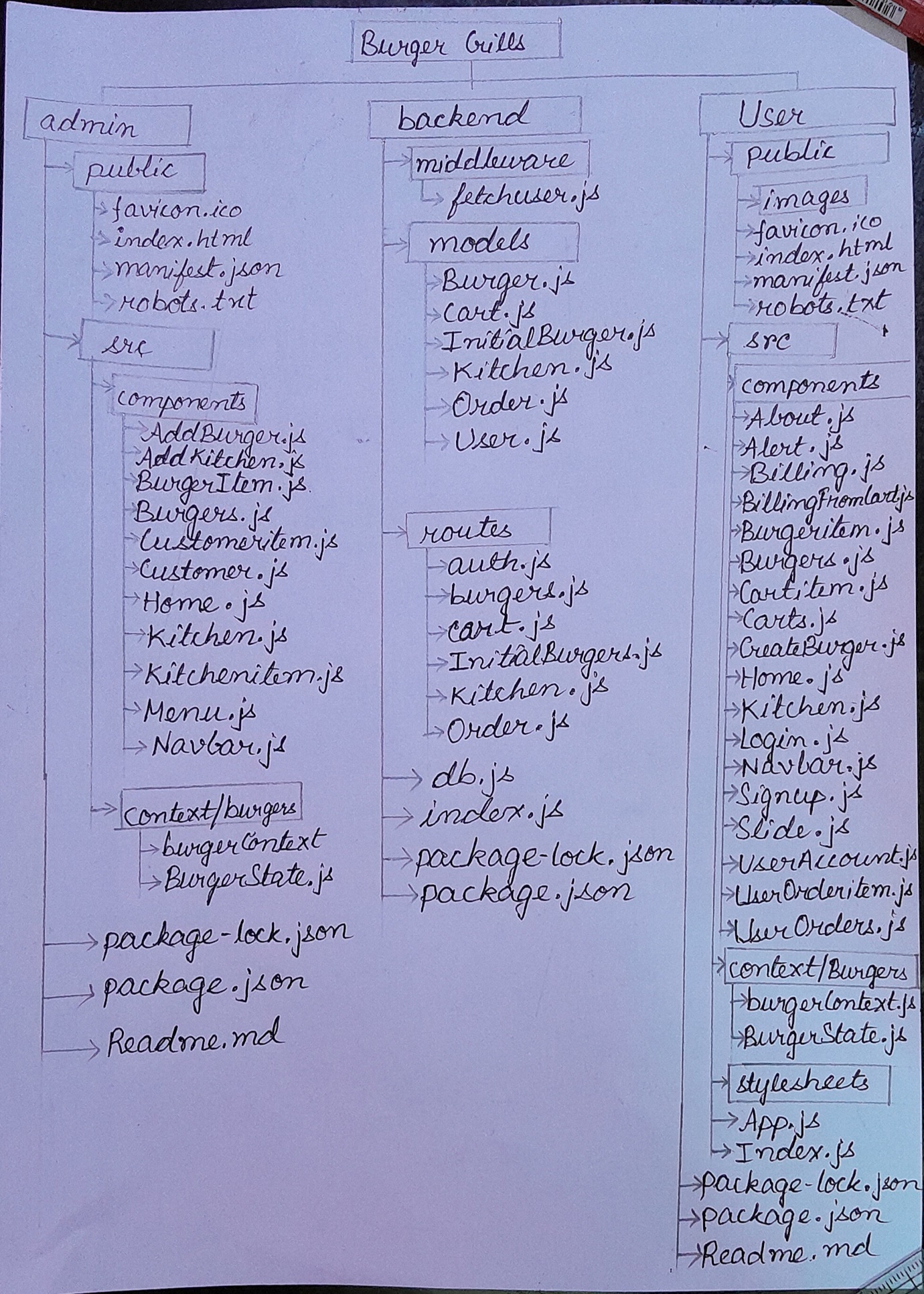
Burger Website

Admin Interface

Burger Website

User Interface

File-Structure:



User Interface:

The site where the customers come to place their orders.

Components:

About.js:

Helps the customer to know more about burger Grills.

Alert.js:

Display an alert passed to it.

Billing.js:

Helps the User to buy a single type of Burger. It displays the user’s name, email, address, payment-type and details of the burger the user is buying.

Functions:

getValue(p,q)- Finds out the product of p and q.

handleBuy(e)-Calls the function addOrder(), addPreviousOrders() from the burgerContext. Shows the necessary Alert. Then navigates to home page.

BillingFromCart.js:

Helps the user to buy multiple types of burgers from the Cart. It displays the user’s name, email, address and payment-type, and price of all the burgers the user is buying.

Functions:

getValue(p,q)- Finds out the product of p and q.

handleBuy(e)-Calls the function addOrder(), addPreviousOrders() from the burgerContext. Shows the necessary Alert. Then navigates to home page.

Burgeritem.js:

Displays the image, title, description and price of the burger passed to it from Burger.js. It also calls a function addCart()(a function passed to it from Burger.js) on clicking the “Add” button.

Burgers.js:

Calls the Burgeritem component and passes it different burgers one by one from burgers array. It also helps to perform action like “Add to Cart” and “Buy Now”. Also display a modal which helps the user to customise their Burger.

Functions:

handleAddToCart(e)-

Calls the functions getUser() and addToCart(cart) from the burgerContext if the user is logged in otherwise directs the user to the login page. It closes the modal and displays the necessary Alert.

handleClose()-

Closes the modal.

handleBuy()-

Calls getUser(), buy() from the burgerContext and navigates to the billing page if the user is logged in otherwise navigates to the login page. It displays the necessary alert and closes the modal.

add()-

Increases the quantity of burger by 1.

minus()-

Decrease the quantity of burger by 1.

addCart()-

Accepts an object as a parameter and assign its value to Cart. Displays the modal which allows the user to customise their burger.

CartItem.js:

Displays the image, title, description, price of the burger passed to it from Cart.js. It also calls a function deleteCart()(a function passed to it from Cart.js) on clicking “delete” icon. Also calls a function updateCart()(a function passed to it from Cart.js) on clicking “edit” icon

Cart.js

Displays the cart of the users. It also shows the user the total value of the cart. User can also change the quantity of an item in the cart.

Functions:

handleUpdateCart(e)-

Closes the modal. Calls the getUser() and updateToCart() functions from the burgerContext and shows the necessary alerts.

getValue(p,q)-

finds the product of p and q.

handleBuy(e)-

navigates to billing from cart if the user is logged in otherwise navigates to login page.

handleClose()-

Closes the modal and initialises the cart variable.

updateCart()-

Accepts an object as a parameter and initialises the value of cart. It then triggers a modal which will help the users to change the quantity of the items in the cart.

add()-

Increases the quantity of the items by 1.

minus()-

Decreases the quantity of the items by 1.

CreateBurger.js:

Allows the user to create their own burger recipes by using the items available in stock. It also changes the price of the burger accordingly with a given base price.

Functions:

handleAddToCart()-

Calls the getUser and addToCart() and shows the necessary alert if the user is logged in otherwise navigates to the home page. Here the burger created is input from the user and send as a parameter of addToCart().

handleBuy(e)-

Helps the user to buy a single type of burger. Calls getUser() and buy(cart) from burgerContext and navigates to billing if the uder is logged in otherwise navigates to login page.

add()-

Increases the quantity of the item by 1.

minus()-

Decreases the quantity of the item by 1.

onChange()-

Updates the item according to the input given by the user.

getValue(p,q)-

Finds out the product of p and q.

Home.js-

Creates the home page of the website by calling the Slide.js and Burger.js components.

KitchenItem.js-

Displays the ingredients available in the store and allows the user to add ingredients according to their choice.

Functions:

minus()-

Decreases the quantity of an item by 1 if it possible.

add()-

Increases the quantity of an item by 1 if there are sufficient items in stock.

Login.js:

Allows the user to login to their account by giving email and password.

Functions:

handleSubmit(e)-

Makes a POST request “http://localhost:5000/api/auth/login” by passing the users email and password as the body. It then receives the value send to it by the request and stores it in a variable json. If login is successful then display a suitable alert and store the authtoken passed by the request.

onChange()-

Updates Credentials object as the user types.

Navbar.js:

It creates a simple navbar that allows the user to navigate between home, about and create your own burger page. It also contains a search box that allows the user to search burgers according to their choice. It contains two buttons that allows the user to go to My Account and My Cart accordingly.

Functions:

handleClose(e)-

Closes the OffCanvas.

handleShows(e)-

Displays the OffCanvas.

handleChange(e)-

Changes the burgeritems according to the the search.

Signup.js:

Helps to create a new user by taking name, email, password, address, city, country, pin code, contact as input from the user.

handleSubmit(e)-

Makes a POST request “http://localhost:5000/api/auth/createuser” by passing the user’s name, email, password, address, city, country, pin code and contact as the body. It then receives the value send to it by the request and stores it in a variable json. If sign up is successful then display a suitable alert and stores the authtoken passed by the request in the localStorage.

onChange(e)-

Updates Credentials object as the user types.

Slide.js:

Used for displaying Carausel on the website which will help the user know more about Burger Grills.

UserAccount.js:

Displays the account details to the users and also helps them to update their details.

Functions:

onChange(e)-

Updates the credentials object as the user types.

handleUpdate(e)-

Calls the editUser() function from the burgerContext, shows the necessary alert and then navigates to the home page.

UserOrderItem.js:

Displays the details of the past order of the user passed to it by UserOrder.js

Functions:

getFormatedDate(date)-

Returns the date passed to it in the correct format.

UserOrder.js:

Displays the past Orders of the user.

burgerContext.js:

Creates a burger Context which allows the user to use functions in various components.

BurgerState.js:

Contains all the useful functions which will be passed as a context.

Functions:

getBurgers()-

Makes a get request `http://localhost:5000/api/Initialburgers/getBurger` to fetch all the burgers available for sale from the database.

getKitchen()-

Makes a get request `http://localhost:5000/api/kitchen/getKitchen` to fetch all the kitchen items (items which the user can add to modify their burgers) from the database.

getUser()-

Makes a post request `http://localhost:5000/api/auth/getuser to fetch the user details and stores the details in a variable ‘user’.

editUser(name, email, password, address, city, country, pincode, contact)-

Makes a put request `http://localhost:5000/api/auth/updateuser/${user.\_id}` to update the user’s credentials. It sends Content-type and auth-token as headers and user’s details in the body. It then updates the user variable.

buy(Cart)-

Helps to buy a single type of burger by updating item variable.

addToCart(Cart)-

Adds an item to user-specific cart. Makes a post request `http://localhost:5000/api/carts/addcart` to add an item to the cart. It sends Content-type and auth-token as headers and cart’s details in the body. Once an item is added to the cart, it then updates the Kitchen items by decreasing the quantity of the items in the extras.

addPreviousOrders(title, description, tag, price, image, extras, removals, quantity)-

Once a burger is bought by the user, this function is called. It makes a post request `http://localhost:5000/api/burgers/addburger` sending the details of the burger passed to it as body, and Content-type and auth-token as the headers. It will help the user to know about their orders.

addOrder(name, email, address, city, country, pincode, contact, Burger, flag)-

Makes a post request `http://localhost:5000/api/order/addorder` sending content-type as headers and the order details as body. ‘flag’ is use to detect whether the user is making order from their cart or otherwise. If the user is making order from the cart, it empties the cart. Else it updates the kitchen items by deceasing the quantity of the items in the extras.

deleteCart(id, extras, qty)-

Makes a delete request `http://localhost:5000/api/carts/deletecart/${id}` sending content-type and auth-token as headers. It the updates the kitchen items by increasing the quantity of the items in the extras.

getcart()-

Makes a get request `http://localhost:5000/api/carts/fetchallcart` by sending content-type and auth-token as headers. It fetches user specific items from the cart. It then stores the items in the cart in allCart.

showAlert(msg,type)-

Displays a alert passed to it in msg parameter for a particular time period.

updateToCart(updateItem)-

Helps to change the quantity of a particular type of burger in the cart. It makes a put request `http://localhost:5000/api/carts/updatecart/${id}` sending content-type and auth-token as headers and the updated quantity as body. It then calls the getCart() function to display updated cart.

getPreviousOrders()-

Makes a get request `http://localhost:5000/api/burgers/fetchallburgers` to fetch all the past orders made by the user. It sends content-type and auth token as headers.

Stylesheets-

These are used for designing the website.

App.js-

It creates a router which allows us to call different components depending on the paths.

index.js-

Creates a root element which renders App.js.

Public:

This folder contains all the media such as images, favicon and index.html. Any image used should be present in the image folder of this folder.

Index.html is used for giving a framework to the website.

BurgerGrills (Admin interface)

Components:

AddBurger.js:

Helps the admin to add burgers to the website which will be available for the customers for buying.

Functions:

handleClick()-

Calls the addBurger() function sending all the required details as parameters.

onChange(e)-

Updates the burger object as the user gives input.

AddKitcken.js-

Helps the admin to add Kitchen items to the website which can be used by the customers to customise their burgers.

Functions:

handleClick(e)-

Calls the addItem() function sending it all the required details as function parameter.

onChange(e)-

Updates the item variable as the user types.

Burgeritem.js-

Displays all the details of the burger passed to it as props. It makes a delete request whenever the delete button is pressed.

Burger.js-

Calls the component Burgeritem and passes it different burgers one by one from burgers array.

Customeritem.js-

Displays all the details of the Order passed to it as props.

Customer.js-

Calls the Customeritem component and passes it different orders one by one.

Home.js-

Calls the Customer components.

Kitcken.js-

Calls the Kitchenitem and passes it different items of the kitchenitem one by one. It also helps to update the number of kitchen items in stock.

Kitchenitem.js-

Displays the name, image and quantity of the item passed to it as props.

Menu.js-

Calls the Burger.js component and AddBurger.js component.

Navbar.js-

Creates a navbar that allows user to navigate to home, menu and Kitchen.

burgerContext.js:

Creates a burger Context which allows the user to use functions in various components.

BurgerState.js:

Contains all the useful functions which will be passed as a context.

Functions:

getBurgers():

Makes a get request `http://localhost:5000/api/auth/getallUsers` to fetch all the burgers which is available for sale.

addBurger(title, description,ingredients, tag, price, image)-

Makes a post request `http://localhost:5000/api/Initialburgers/addIburger` to add a new Burger by sending Content-type as headers and all the burger details as body. It then adds the new burger to burgers array.

getOrders()-

Makes a get request `http://localhost:5000/api/order/getorder` to get all details of the order made by a customer.

deleteOrder(id)-

Makes a delete request `http://localhost:5000/api/order/deleteorder/${id}` to delete the order with that particular id(passed as function parameter) from the database.

getKitchen()-

Makes a get request `http://localhost:5000/api/kitchen/getKitchen` to fetch all the details of the kitchen items in stock.

addItem(title, price, image, quantity)-

Makes a post request `http://localhost:5000/api/kitchen/addKitchen` to add a new item to the kitchen. It sends Content-type as header and title, price, image and quantity as body.

updateToKitchen(item)-

Makes a put request `http://localhost:5000/api/kitchen/updateKitchen/${item.eid}` to update the quantity of an item in the kitchen. It sends content-type as header and the updated quantity as body.

App.js-

It creates a router which allows us to call different components depending on the paths.

index.js-

Creates a root element which renders App.js.

Index.html is used for giving a framework to the website.

Backend:

Middleware

fetchuser.js-

Used for verifying the json web token.

Models: Helps us to create mongoose schema to store and read items easily from the database.

Burger.js-

Helps in storing the details of all the past orders made by a particular user.

Cart.js-

Helps in storing the details of items in cart of a particular user.

InitialBurger.js-

Helps us in storing the details of the burgers available for sale.

Kitchen.js-

Helps us in storing all the details of the items available for the users to modify their burgers.

Orders.js-

Helps in storing the Customer details and the orders details made by them.

User.js-

Helps in storing the user details such as name, email, password, address, city, country, pincode, contact.

Routes-

Helps us in creating all the necessary end-points.

db.js-

Connects the database to the backend.

Index.js-

Creates a backend express server at port 5000. It also helps in linking end-points.

Acknowledgements:

All the images have been taken from www.google.com.