**Food Ordering App**

* First, we identify the components that we must make. Here we made UI, Layout, Meals, and Cart component.
* Now we make the Header component in the Layout folder. In this component we made header named Daily meals and button which Is used to add item to the cart. We have created a separate file for the cart button. In that component we have used 3 spans first one will contain the icon; second one will contain the name and the third one will contain the badge/number of items.
* Now we will make the Meals component that will render the meals. For that in the Meal component we make 3 files that contains Meals.js, MealsSummary.js and AvailableMeals.js which are self-explanatory by their names. In the AvailableMeals.js component we will map the Dummy data which is an array of object to the component using the .map() function. And then we will call these 2 components in the Meal.js file and after that in App.js
* Now as the next step we will wrap this meal into a card component which will look good. That card component is made in the UI folder which provides some HTML structure and CSS code. After that we created another folder which will contain meal items. And, we make one more component which holds the form on how much item you want to add. And also created another component named Input.js in UI folder. In there in the input tag we use the spread operator to spread props.input. This ensures that all the key value pairs in this input which receives on props input are added as props to input. {…props.input} it will that id there is an object that shows {type:text} then it will add this as type = text in the input tag. Now with that added we can import this to MealItem component. So in this component in the Input tag we use the doubly curly braces where one curly brace is to evaluate a JS expression which is passed in as a value. And then that expression just happens to be Javascript object
* Now we work on the Cart component which should render all the cart items and of course display the total amount. And then also give us the button to order or leave the cart.
* Now we make the modal for that we will create a file named Modal.js in the UI folder and also added some CSS. We will use React Portal for both the backdrop so that thing behind the modal overlay which blocks the interaction with the rest of the page. To render a portal, we first go to public->index.html and there we got the root div where the overall react application is being rendered and above that root div we will add another div with the id overlay and that is where we will portal our modal. To create a portal syntax is React.DOM.createPortal(<Backdrop />,//place where the item is to portal).
* So now we ensure that cart modal shows up when clicked on the cart and closes when clicked on the close button. For that, we need to manage some state one state where the cart is visible and another where the cart is not visible. Now we use the react usestate method to change the state of the cart. Now show cart handler should be called whenever the button is clicked. However this button is the part of the Header component so we get that function from inside the Header. We have done this by using props drilling. To close the cart when clicked on the backdrop we in the App.js we pass a onClose prop in Cart component and also in the modal.js file in the Backdrop component we made a prop named onClick which will be passed where this component is called and after that in the Cart.js file in the <Modal onClose={props.onClose}>.
* **Context Api: -**  we use context to make sure that data can be added to the cart and also manage the overall data of the cart. To start off we make a store folder, in that we first make the car-context. We initialize the context and in that we will have items array since we are going to mange couple of cart items. And we will have an total amount and then we have two functions which will allow us to update that context. Now we got this context generally created and now we have to manage the context with useState or useReducer for that we will create another file name cartProvider.js and the goal of this component is simple to manage the current context data and provide that context to all the components that want access to it.
* **Using the Context: -** For adding cart item we need to go to the cart provider because that’s where we manage the state. Now whenever the add to cart handler function is called that item should be added to the cart and if that item already exits in the cart I will update that. And the next step we will add a cart reducer function outside the component function. Now we have learned that in the reducer function we receive a state object and an action automatically by React. The action is dispatched by us later in the code and the state is simply the last state snapshot of the state managed by the reducer. Then we will create another constant named default Cart state that will contain default values. (lecture 177) Now we need to make sure that as the next step that add to cart handler is being called from some component in our application like the Meals component where we have meal item and other.