**Task 4 Report**

**1. Task Description**

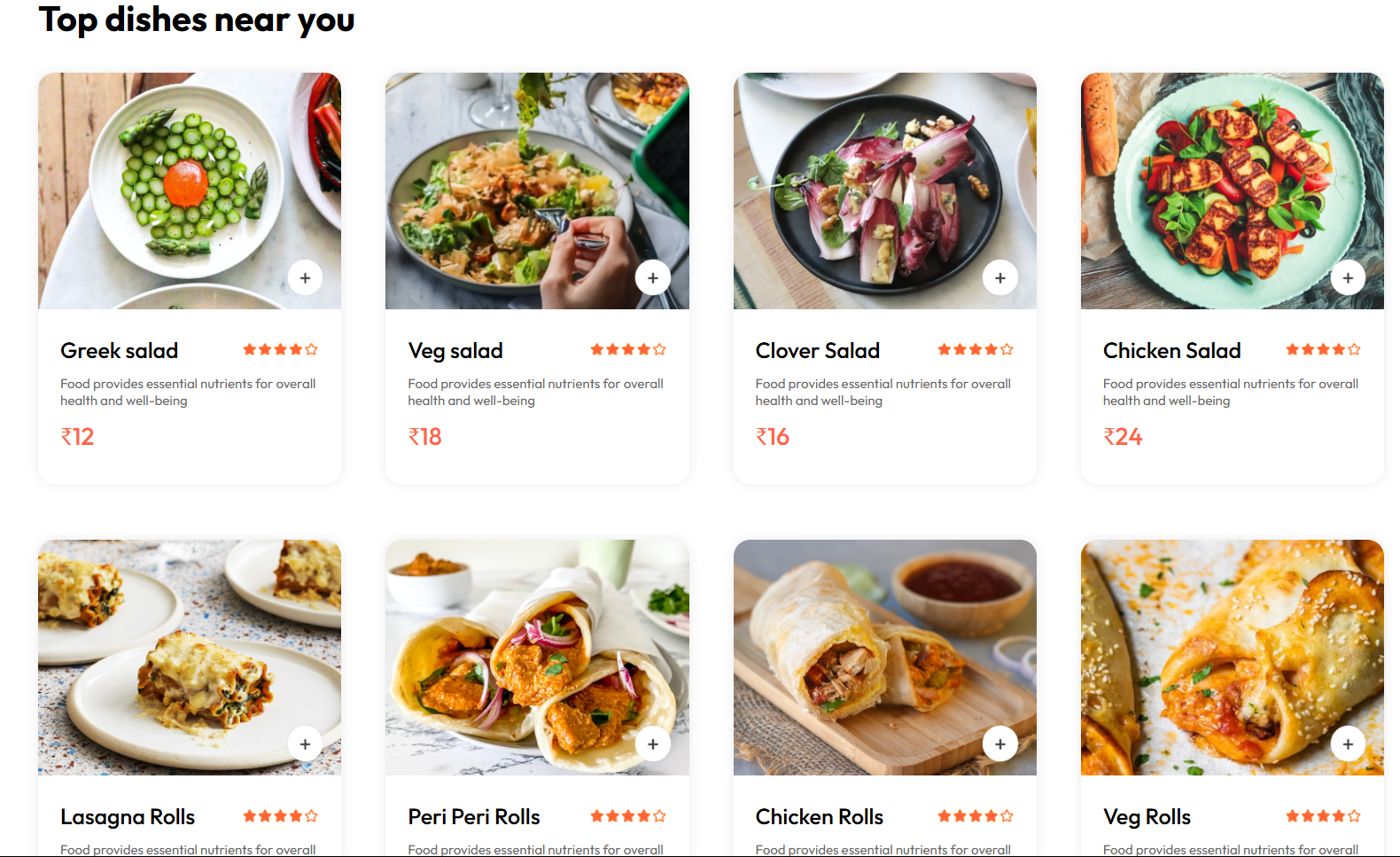
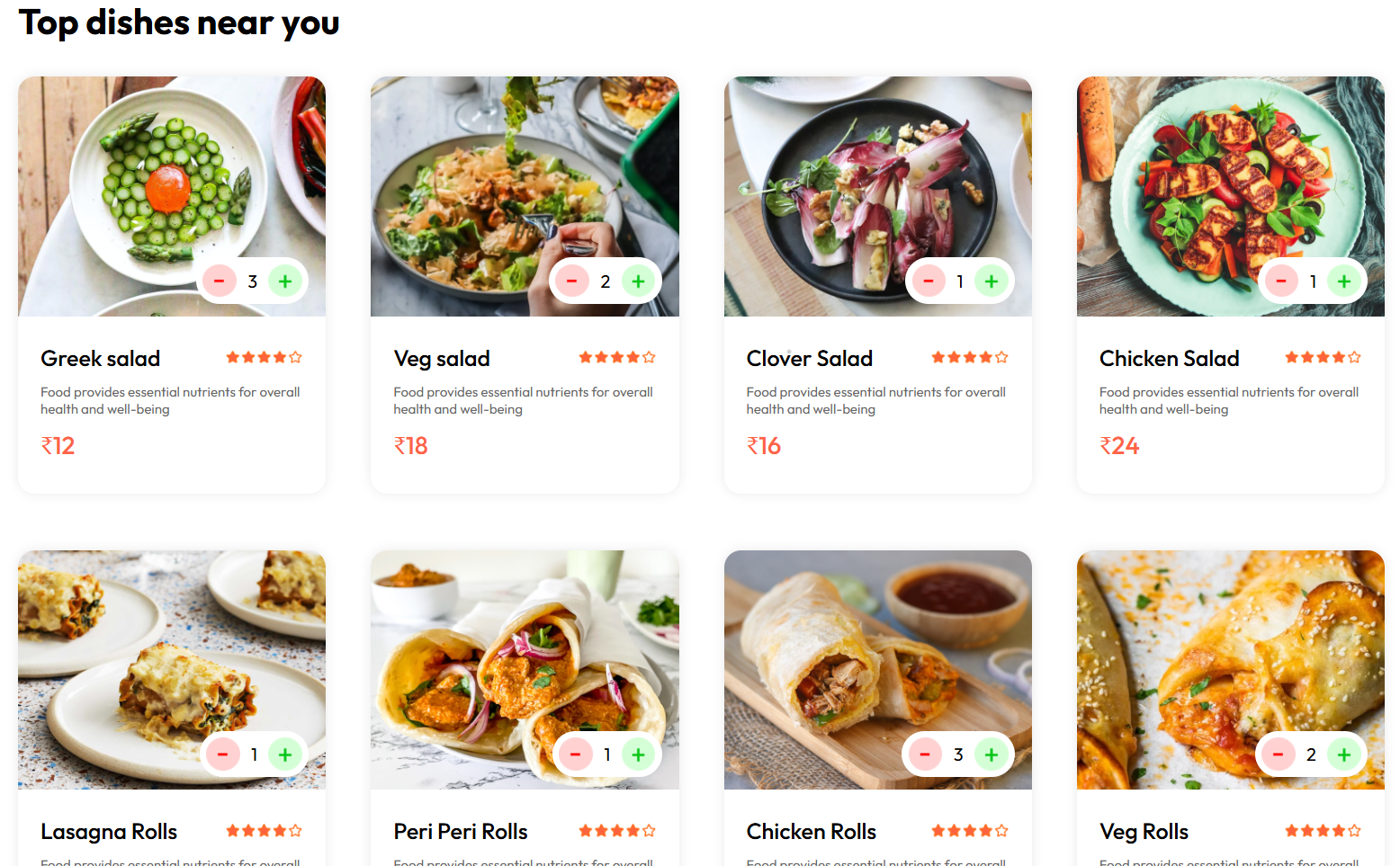
This task involves implementing an "Add to Cart" functionality in a React app for an e-commerce-like application. Users can add items to their cart, view the total amount, and manage cart quantities (add or remove items). The app also dynamically calculates the total price of items in the cart.

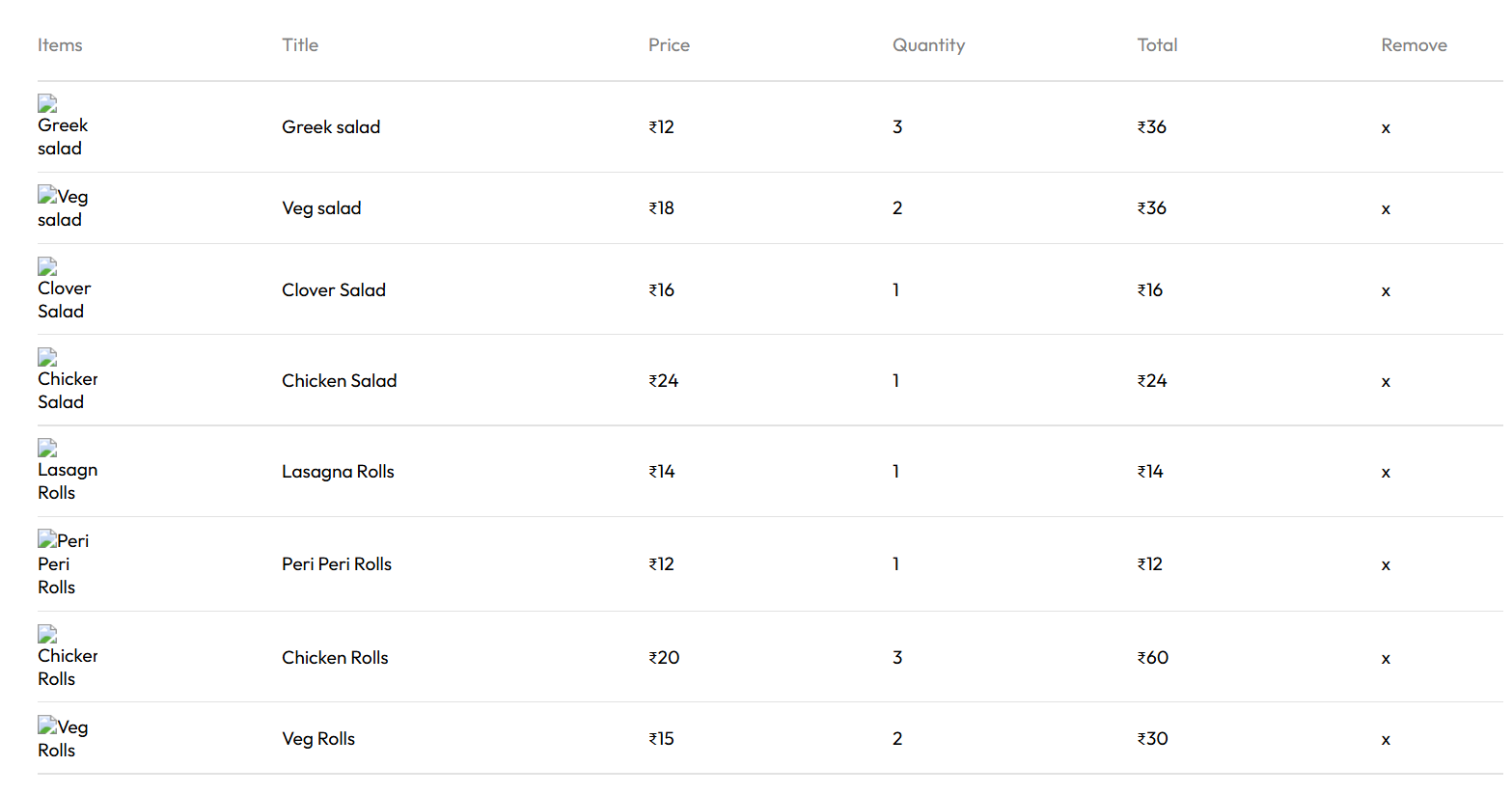
The app is designed without a backend server, using local data stored in an assets.js file. The StoreContext is used to manage global state, providing functionality and data such as food\_list, cartItems, and addToCart across components.

**2. Output**

**Below is an example description of how the app might look with basic Add-to-Cart functionality.**

* **Product List Section:**
  + **Each product displays its name, price, and an "Add to Cart" button.**
* **Cart Section:**
  + **Displays the items in the cart, quantity, and total amount.**

 ****



**Describe Widget/Algorithm Used in Task**

**Widgets/Components Used**

1. **Product List Component**:
   * Displays the list of products from food\_list with:
     + Name
     + Price
     + "Add to Cart" button linked to addToCart.
2. **Cart Summary Component**:
   * Shows items added to the cart, their quantities, and the total price calculated using the getTotalCartAmount function.
3. **StoreContext**:
   * Implements a global state for managing:
     + Product list (food\_list).
     + Cart items (cartItems).
     + Token for authentication.

**Algorithm for Add-to-Cart**

1. **Add Item to Cart**:
   * When a user clicks "Add to Cart":
     + Check if the item exists in cartItems.
     + Increment the quantity by 1 if it exists.
     + Add the item with quantity 1 if it doesn't exist.

const addToCart = (itemId) => {

setCartItems((prev) => {

const newCount = (prev[itemId] || 0) + 1;

return { ...prev, [itemId]: newCount };

});

};

1. **Remove Item from Cart**:
   * When a user removes an item:
     + Decrement the quantity by 1.
     + Remove the item entirely if the quantity reaches 0.

const removeFromCart = (itemId) => {

setCartItems((prev) => {

if (!prev[itemId]) return prev;

const newCount = prev[itemId] - 1;

if (newCount === 0) {

const { [itemId]: \_, ...rest } = prev;

return rest;

}

return { ...prev, [itemId]: newCount };

});

};

1. **Calculate Total Price**:
   * Loop through cartItems to calculate the total cost using the price and quantity of each item.

const getTotalCartAmount = () => {

let totalAmount = 0;

for (const item in cartItems) {

if (cartItems[item] > 0) {

const itemInfo = food\_list.find((product) => product.\_id === item);

if (itemInfo) {

totalAmount += itemInfo.price \* cartItems[item];

}

}

}

return totalAmount;

};