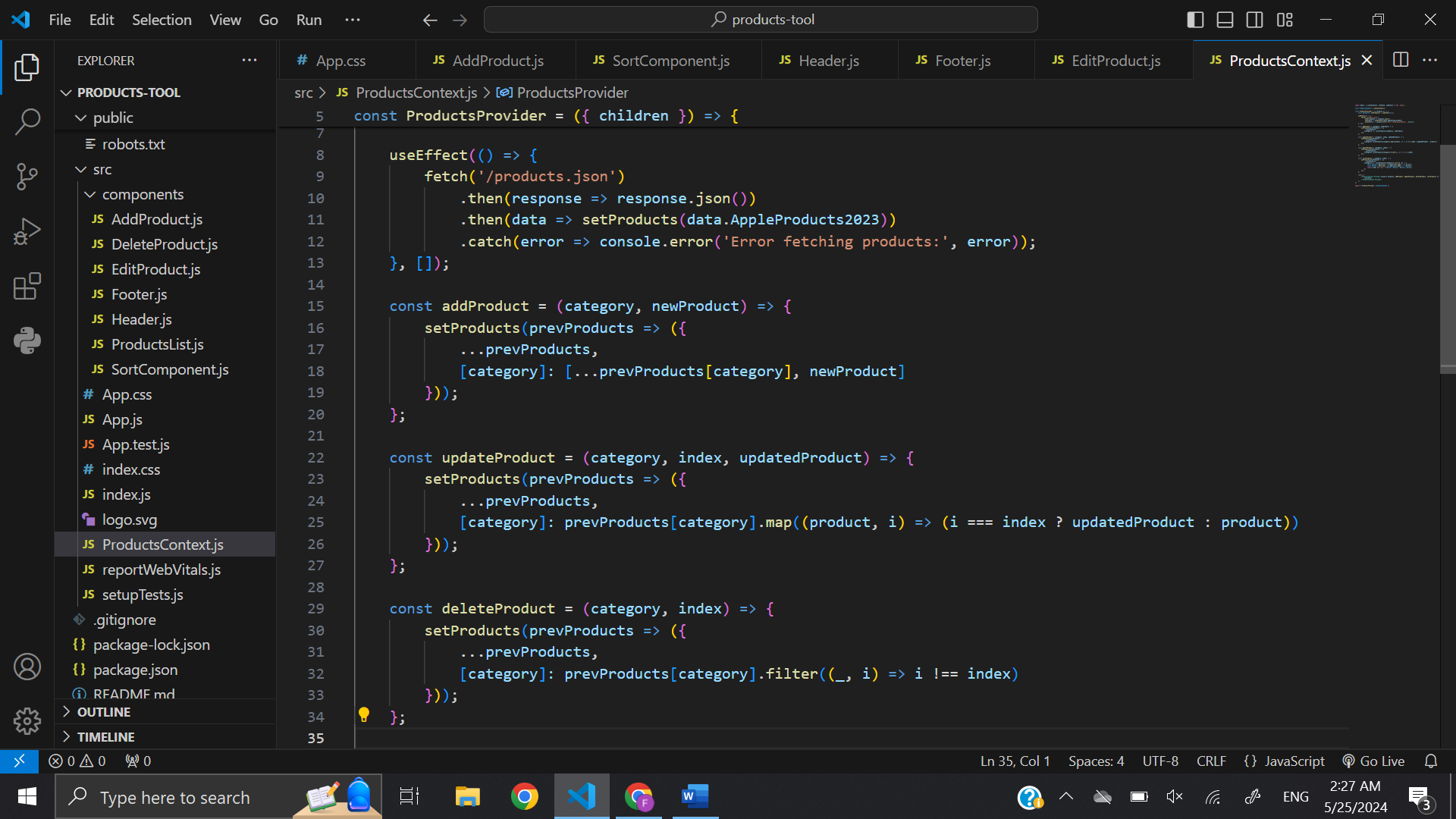
**Introduction:**

This React project is designed to display and manage a list of Apple products, including iPhones, iPads, and MacBooks. It utilizes a JSON file to store product data and provides functionalities to view, add, and sort products by price. The project includes several key components:

1. **ProductsContext**

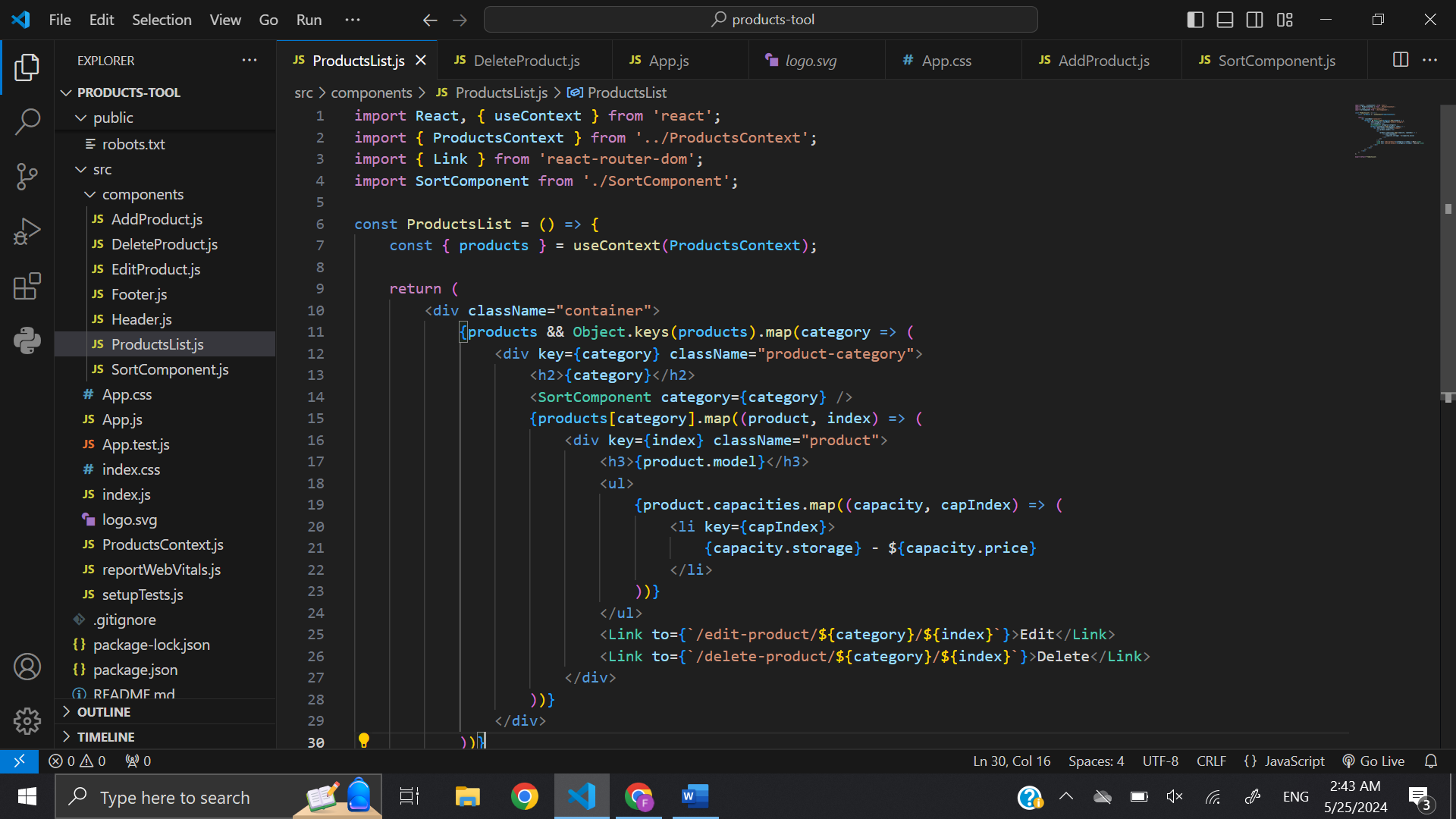
* This component uses the Context API to manage the state of the product data. It provides a central place to store and update the product information.



* **ProductsContext**: This is a React context that will be used to share the product data and manipulation functions across different components in the application.
* **ProductsProvider**: This component wraps the application and provides the context to its children. It manages the state of the products and provides functions to add, update, delete, and sort products.
* **State Initialization**: The **useState** hook initializes the **products** state to **null**.
* **Fetching Products**: The **useEffect** hook fetches product data from **products.json** on component mount and updates the **products** state.
* **addProduct**: This function takes a **category** and a **newProduct**, and updates the state by adding the new product to the specified category.
* **updateProduct**: This function takes a **category**, an **index**, and an **updatedProduct**. It updates the product at the specified index within the specified category.
* **deleteProduct**: This function takes a **category** and an **index**, and removes the product at the specified index from the specified category.
* **sortProducts**: This function takes a **category** and an **order** ('asc' for ascending or 'desc' for descending) and sorts the products within the specified category based on their minimum price.
* **ProductsContext.Provider**: This provider component passes down the current **products** state and the functions (**addProduct**, **updateProduct**, **deleteProduct**, **sortProducts**) to its children via context.

1. **ProductsList**

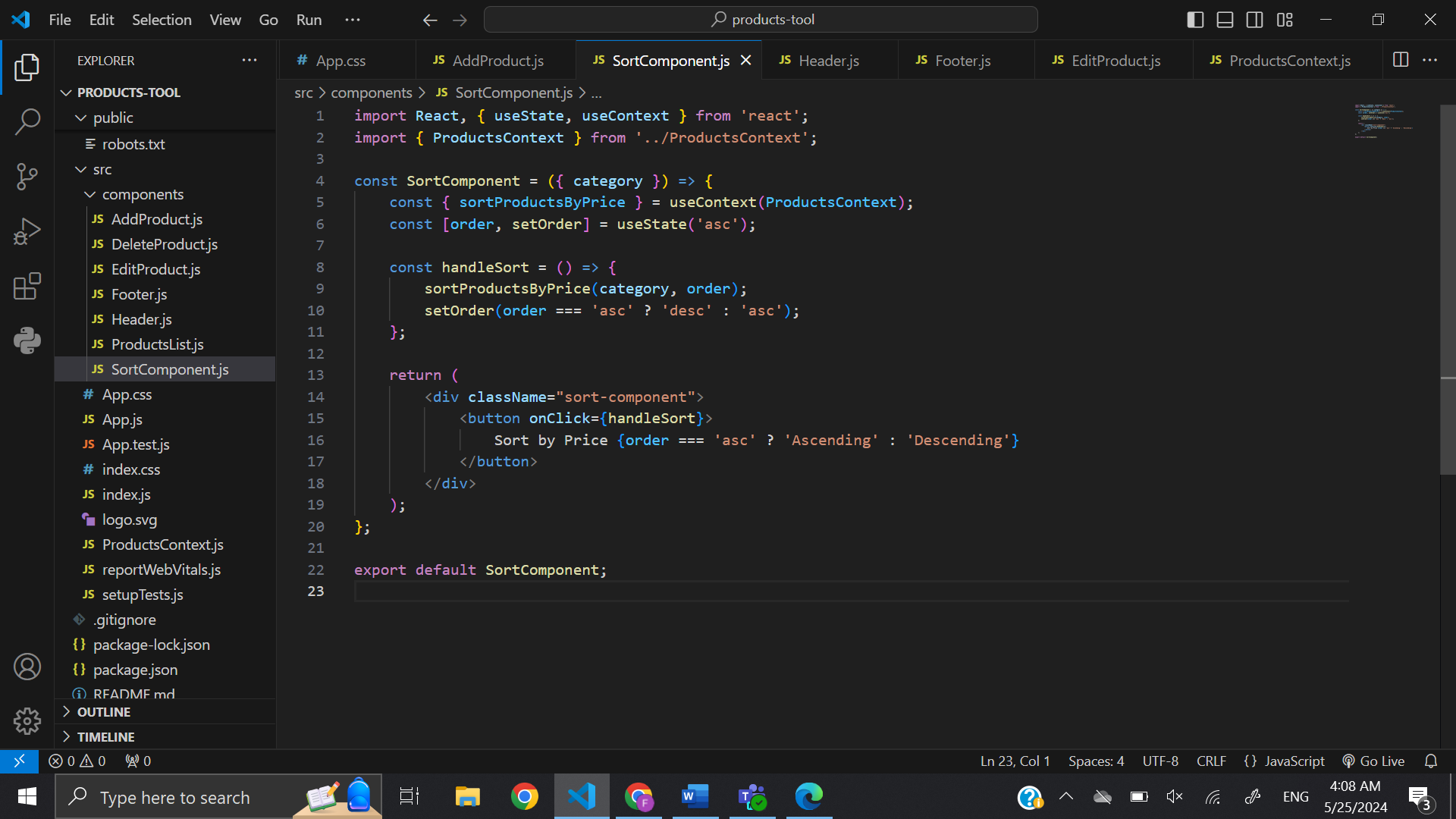
* This component displays a list of products, categorized by device type (iPhone, iPad, MacBook). It fetches product data from the context and renders it.



* Check if products data exists. If so, map over each category of products.
* For each category, render a section with the category name (<h2>) and a SortComponent for sorting products within that category.
* Display each capacity of the product (storage and price) in an list (<ul>).
* Provide links to edit and delete the product, using the Link component from 'react-router-dom'.

1. **SortComponent**

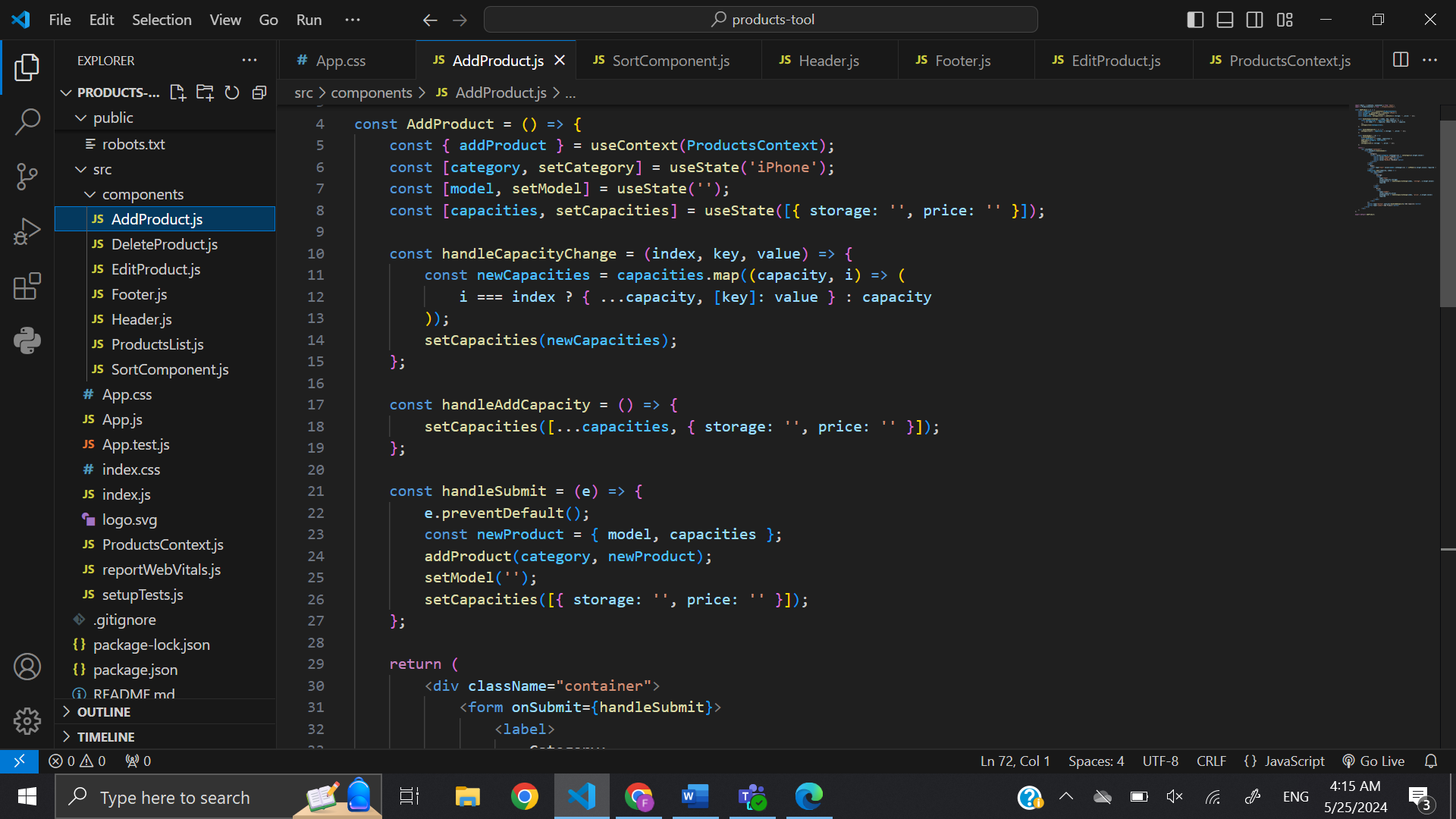
* This component provides functionality to sort the list of products by price, either in ascending or descending order.



* Initialize state with useState to track the sorting order ('asc' for ascending or 'desc' for descending).
* Define a handleSort function that toggles the sorting order and calls the sortProductsByPrice function with the current category and sorting order.
* Update the state to reflect the new sorting order after sorting.

1. **AddProductForm**

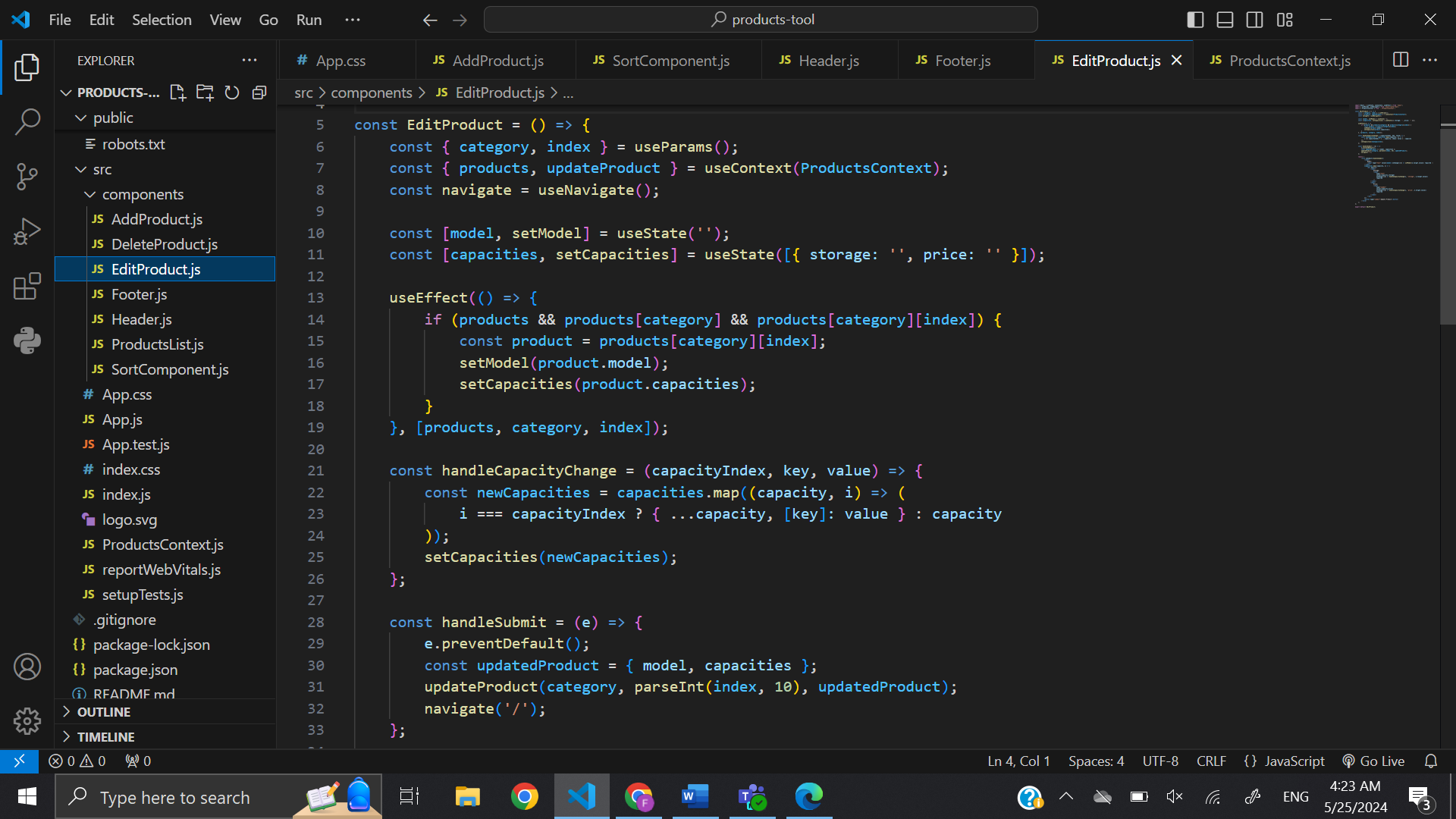
* This component renders a form that allows users to add new products to the list. It updates the product context with the new product data.



* handleCapacityChange: Updates the capacity (storage or price) of the product when input values change.
* handleAddCapacity: Adds a new capacity object to the capacities array.
* handleSubmit: Handles submission. It constructs a new product object with the entered model and capacities, then calls the addProduct function with the selected category and the new product.

1. **EditProduct**

* This component provides a form to update an existing product's details.



* Use the useEffect hook to fetch and populate the product data when the component mounts.
* handleCapacityChange: Updates the capacity (storage or price) of the product when input values change.
* handleSubmit: Handles form submission. It constructs an updated product object with the modified model and capacities, then calls the updateProduct function with the category, index, and updated product.

1. **DeleteProduct**

* This component handles the deletion of a product. It removes the specified product from the list.

A screenshot of a computer program

Description automatically generated

* Use the useEffect to perform the deletion operation.
* Check if the product exists in the products state based on the provided category and index.
* If the product exists, call the deleteProduct function to remove it from the state.
* After deletion, navigate the user back to the home page.

**Resources and References:**

* CPIT-405 Course website:
* Had to make a reference to in class activities and labs to know how to make the SortComponent for sorting the prices, also to deal with specific handle JS events.
* Used for understanding React's core concepts, hooks, and context API.
* React Router DOM: Referred to for implementing client-side routing in the application.
* Stack OverFlow:
* Used for troubleshooting and finding solutions to specific coding issues.
* Used to know the process to create a context that holds the products data and provides functions to manipulate this data.
* AI-Model:
* Used for finding solutions to specific coding issues.
* Fix a few issues with the forms, Also we had a problem with handle events in the EditProduct component that need to be fixed!

**Conclusion:**

This project demonstrates effective use of modern React practices, promoting clean code, reusability, and maintainability.

Looking forward, it would be beneficial to integrate a backend server to persist data changes in a JSON file or a database, thereby enhancing the application's functionality and reliability. This upgrade would involve setting up a server with endpoints to handle CRUD operations, ensuring that data remains consistent and secure across sessions.