**Student Name and Number as per student card:** Yash Palande (20043004)

**Programme:** Information System with Computing

**Lecturer Name:** Prof. Paul Laird

**Module/Subject Title:** Programming for Information System

**Assignment Title:** Furniture Shop

By submitting this assignment, I am confirming that:

* This assignment is all my own work;
* Any sources used have been referenced;
* I have followed the Generative AI instructions/ scale set out in the Assignment Brief;
* I have read the College rules regarding academic integrity in the QAH Part B Section 3, and the Generative AI Guidelines, and understand that penalties will be applied accordingly if work is found not to be my/our own.
* I understand that all work submitted may be code-matched report to show any similarities with other work.

**Overview:**In this project, I developed an e-commerce platform called Furniture Shop using React for frontend and JSON server for backend.

The goal of the project was to build a fully functional online store where user could browse product and the main goal was for administrators could manage them through an simple, intuitive interface.

The reason I choose React and JSON server was simple:  
React component based architecture provides flexibility and scalability and build dynamic UIs, while JSON server allowed me to quickly setup a backend. By combining this tools, I was able to have a functional backend that could handle CRUD operation for the products.

**Choosing React and JSON Server:**React was the natural choice for building frontend due to its popularity and vast ecosystem with react, I could easily manage the state of the application using hook like useState and useEffect, making the development process more efficient. Additionally, React’s component driven structure allowed me to break down the project into smaller, manageable pieces, enabling faster iteration and easier debugging.

For the backend, I choose JSON server, a mock REST API that provides a simple way to stimulate backend server. JSON server is capable of handling CRUD operations for resources like products.

**Development : Structure and Components**  
The main structure of application is divided into components, pages and routing configuration. The project start with the creation of new react app using create-react-app tool, which provided a ready to use template for React development. The initial setup was cleaned up by removing unnecessary files and configuration index.js to render a simple inline App component displaying a welcome message.

For the layout, I used Bootstrap through a CDN to quickly style the application with a responsive grid system and pre-build components. The Navbar and Footer components were build to ensure consistent navigation and display across all pages. The Navbar include link such as Home, Contact, and Admin product management pages, while the Footer provides basic store information and a consistent look at the bottom of the page.

**Routing and Navigation with React Router:**

To handle page navigation, I integrated React Router DOM. I wrapped the entire application in a BrowserRouter component in index.js and used the Routes component to define the routing structure. The main pages - Home and Contact - were connected to their respective routes (/and/contact). Additionally, for any undefined routes, I added a NotFound page to display a user-friendly error message.

The Navbar link s were updated to use the link component from React Router, ensuring that navigation was handled without reloading the page. This allowed for smooth transitions between pages and a better user experience overall.

**Product Management with JSON Server:**

For managing product data, I implemented JSON Server as the backend, JSON server provided a quick solution for simulating a Restfull API. I created a category , price, image, and creation date.

I also added a public/images folder to store product images, which were accessible via URLs like <http://localhost:4000/images/>. JSON server was configured to serve this data over a local server running on port 4000, allowing me to easily interact with product data via the frontend.

Admin Panel: Product List and CRUD Operations

One of the key features of application in the Admin Panel, where administrators can manage the products. The product management page (/admin/products) was created using React components and a Bootstraps tables to display the products. Each product was displayed with a set of action button edit and delete.

To populate the table, I fetched product data from JSON Server backend using the fetch API. The useEffect hook was used to automatically fetch product data when the page loaded. The data was stored in the products state variable and mapped over to dynamically render rows in the table.

I added functionality to refresh the product list with the “Refresh” button by re-fetching the data from the server. Additionally, I implemented sorting to display the most recently added products first by using the \_sort and \_order query parameters in the fetch URL.

**Adding New Products and Image Handling**

The application allows the creation of new products via a form in the CreateProduct component. The form includes fields for name, brand, category, price description and images. The image file is handled using Multer, a middleware for handling file uploads. I configured Multer in the backend to store the uploaded images, Also validation was applied to ensure that all required filed were filled in.

**Server Configuration with Multer :**

The backend server was configured to handle both product data and image uploads. I used Multer to process file uploads and ensure that images were stored in the correct folder. The server was set up to handle POST request for creating the new product and PATCH requests for updating existing ones. I also added validation to ensure that all the required fields were provided.

**Editing and Deleting Products:**  
The ability to edit existing products was implemented in the EditProduct component. When navigating to the edit page (/admin/products/edit/:id), The product’s current details were fetched from the backend and pre-populated in the form fields. The form was designed to allow changes to the product name, brand, description, and image.For image updates, the user could choose to either upload a new image or keep the exixting one. The update request was sent to the server using patch request, which only updated the fields that were changed.

The DeleteProduct functionality allowed administrators to delete products from the list. By clicking the delete button in the product table, a Delete request was sent to the server to remove the products, and the UI was updated to reflect the changes.

**Conclusion**

This project successfully demonstrates the power of React for building dynamic, component-based user interfaces and JSON Server for quickly setting up a backend. By integrating these tools, I was able to create a fully functionally e-commerce platform with product management, validation, and image handling, all while keeping the development process efficient and manageable.

**Reference:**

[**https://getbootstrap.com/docs/5.3/getting-started/introduction/**](https://getbootstrap.com/docs/5.3/getting-started/introduction/)

[**https://github.com/typicode/json-server/tree/v0**](https://github.com/typicode/json-server/tree/v0)

**Repo\_Link** :

https://github.com/yashpalande26/CA\_ONE\_IS\_Project.git