Assignment # 4

Due Date Wednesday, April 7, 2021, @11: 59 PM

In this Assignment, you will build **the product management app** app's backend using express and nodejs

Preparation

1. Sync cmps350-lab repo to get the Assignment files
2. Open the **assignment4/ProductManagementApp** project
3. The project should be organized as follows:
    a. public folder contains the HTML pages, templates, CSS, and client-side JavaScript
    b. data folder contains **products.json** files to be used in this Assignment.
4. Run **npm install express** to install express package.

PART A – Creating the Server and Product APIs

- Use the given **data/products. json** file for persisting and reading the data for all the below methods.
- Create a **product-repo.js** file and implement ProdcutRepo class with the following methods listed in the table below.

Method	Description
getProducts()	Return all products inside the products.json file
getProduct(pid)	Return a single product that has the same product id(pid)
addProduct(newProdcut)	Adds new product to the products.json file
updateProdcut(updatedProduct)	Updates existing product that matches one of the product inside products.json file
deleteProduct(pid)	Deletes specific product from the **products.json** file using the product id (pid)
deleteAllProducts()	Remove all the products saved inside products.json file

getStatistics()	Returns the an object containing the total number of products in the supermarket and their total price.
	Example object should look like
	{
	totalNumberOfProducts : 5,
	totalPrice : 500
	}

Note : Test your **ProdcutRepo** class before moving to the next step to make sure everything works. You can create an object of the **ProdcutRepo** class and test all the methods or use mocha and chai. Once you finish the testing comment out or remove the test code.

- Create a **prodcut-service.js** file and implement **ProductService** class with methods to handle the requests listed in the table 1 below. The handlers should use the provided methods from the ProductRepo class.
- Create a **router.js** file and implement handlers for the routes listed in the table below. The routes handlers should use the methods from the **ProductService** class.

HTPP Verb	URL	Functionality
Get	/api/products	Return all products inside the **products.json** file
Get	/api/products?pid={**product id**}	Return a single product that has the same product id(pid)
Post	/api/products	Adds new product to the **products.json** file
Put	/api/products	Updates existing product that matches one of the product inside **products.json** file
Delete	/api/products/:pid	Deletes specific product from the **products.json** file
Delete	/api/products	Remove all the products saved inside **products.json** file

Get	/api/products/stats	Returns the an object containing the total number of products in the supermarket and their total price.
		Example object should look like
		{
		totalNumberOfProducts : 5,
		totalPrice : 500
		}

- Finally, create **app.js** file, and import express package, instantiate an express app, configure it then start it at port 9090.
- Test your API's using **PostMan**

Connecting the Client App to the Server

Your task is to **replace** the Indexed DB with API calls to the server APIs that you created above. The application should work exactly the same way as Assignment 3 with one additional method called **Read**. The complete code is given to you. You only need to write the todos of the **public/js/repository/product-repo**.js file on the client side. And make the **product management app** app communicate with the server.

The **Product Management App should** have the following CRUD features:

1. **Create Product**: when the user clicks on the **create new product** button in figure 1, you should open the add product page in figure 2 that allows the user to add a new product. Once they click on the **save** button, you should navigate them to the product page and show the newly added Product, as shown in figure

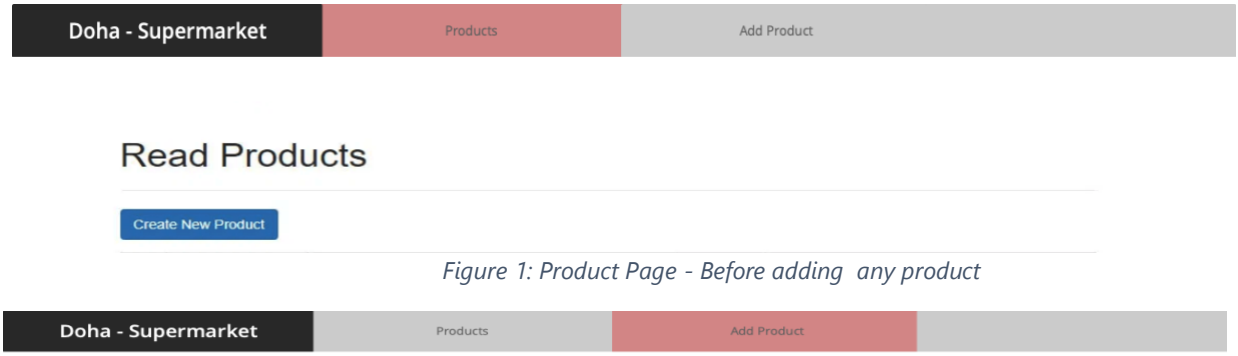

Figure 1: Product Page - Before adding any product

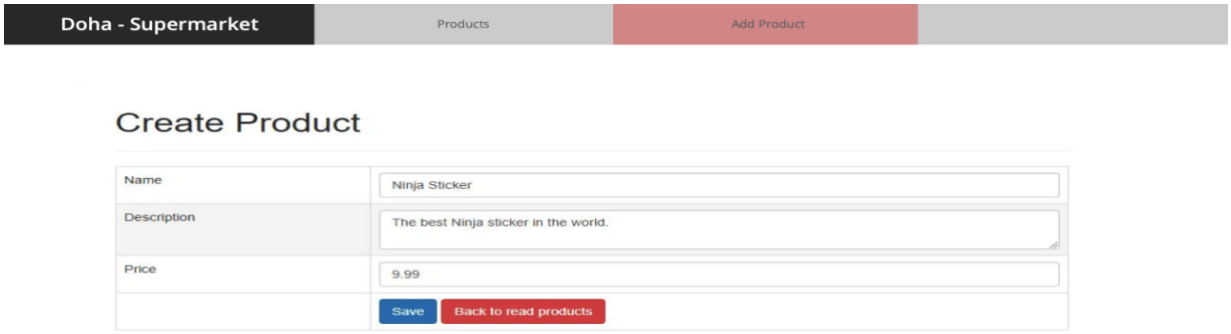

Figure 2: Add Product page

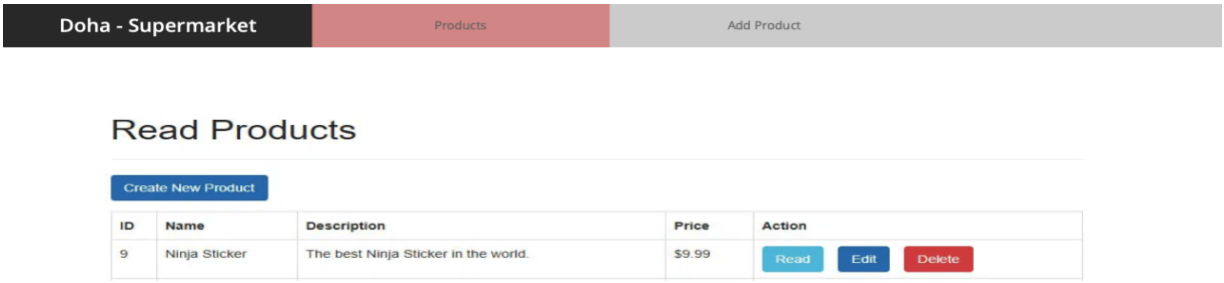

Figure 3:Product Page - After adding Product

2. **List Products:** as soon as the application loads, you show all the products saved in the server, as shown in figure 4

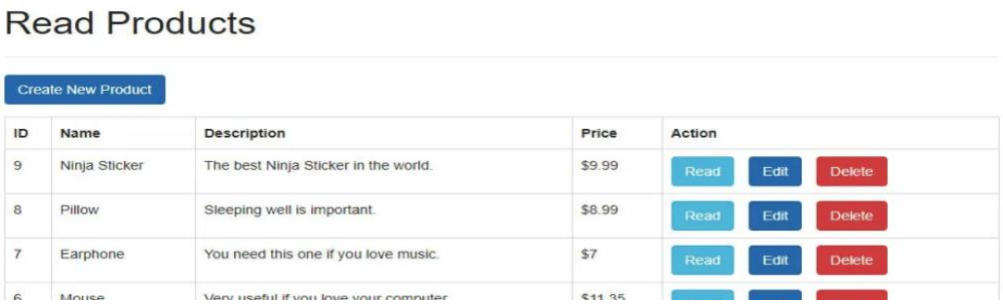

Figure 4: Products page

3. **Read Product:** Displays a single product with its information on a separate page
4. **Edit Products:** If the user clicks on the edit product button, you should take the
5. selected product details to allow the user to update the information of the selected Product, as shown in Figure 5.

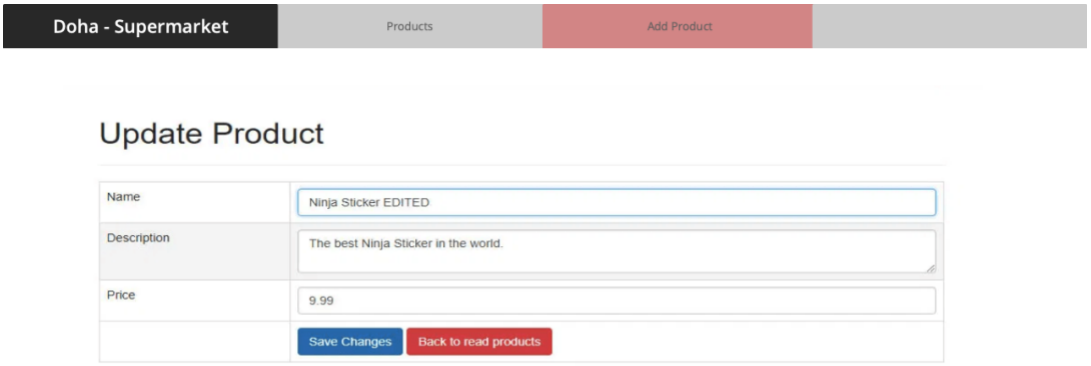

Figure 5 Update Product

6. **Delete Product:** Delete the selected Product from the database and reload the page.

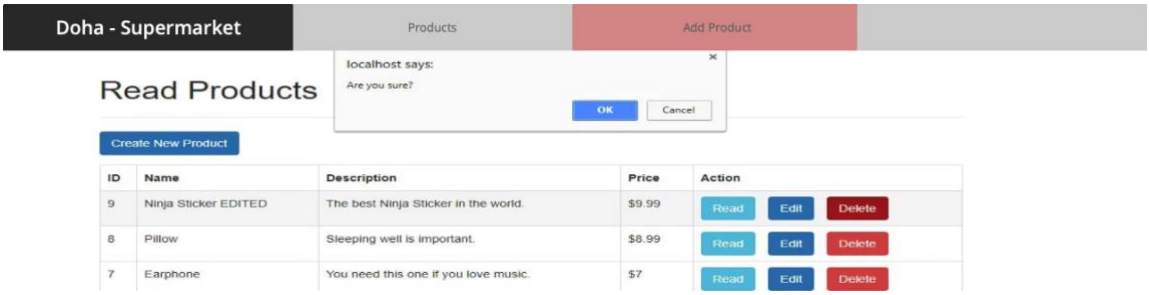

Figure 6 : Delete Product

After you complete the Assignment, fill in the ***TestingDoc-Grading-Sheet.docx*** and save it inside the ***Assignment4*** folder. Push your work to the Github repository.