Title : Create Full stack web app to perform CRUD operations on the data stored in MongoDB database.

Problem description: Implement insert, find, update, and delete operations using the Node.js MongoDB driver. Insert data into the products collection using Mongoose. Create an Express API with a /products endpoint to fetch all products. Use fetch in React to call the /products endpoint and display the list of products. Add a POST /products endpoint in Express to insert a new product. Update the Product List, After adding a product, update the list of products displayed in React.

Method: Install the MongoDB driver for Node.js. Create a Node.js script to connect to the shop database. . Define a product schema using Mongoose.

Step 1: Setup Your Environment

1. **Install Node.js (please refer to previous experiment)**
2. **Initialize a Node.js Project**

   mkdir experiment8;

   cd experiment8;

   mkdir server;

   cd server

3. **Install Required Dependencies in server folder**

   npm init -y

   npm install cors express mongoose

   npm install dotenv

   Note: Edit package.json add `"type":"module"`

Step 2: Set Up MongoDB

1. **Install MongoDB and MongoDB Shell**

Step 3: Create following file under experiment8/server folder

server.js

```
import express from "express";
import mongoose from "mongoose";
import cors from "cors";
import dotenv from "dotenv";
dotenv.config();

const app = express();
app.use(cors()); // allow frontend requests
app.use(express.json());

// MongoDB connection
const DB_URL = process.env.MONGO_URI || "mongodb://127.0.0.1:27017/shop";
mongoose.connect(DB_URL, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

const db = mongoose.connection;
db.on("error", console.error.bind(console, "Connection error:"));
db.once("open", () => console.log("Connected to MongoDB"));

// Product schema & model
const productSchema = new mongoose.Schema({
  name: String,
  description: String,
  price: Number,
});

const Product = mongoose.model("Product", productSchema);

// Routes

// Get all products
app.get("/products", async (req, res) => {
  try {
    const products = await Product.find();
    res.json(products);
  } catch (err) {
    res.status(500).send(err.message);
  }
});

// Add a product
app.post("/add", async (req, res) => {
  try {
    const newProduct = new Product(req.body);
    await newProduct.save();
    res.status(201).json(newProduct);
  } catch (err) {
    res.status(500).send(err.message);
  }
});

// Update a product
app.put("/update/:id", async (req, res) => {
  try {
    const updatedProduct = await Product.findByIdAndUpdate(
      req.params.id,
      req.body,
      { new: true }
    );
    res.json(updatedProduct);
  } catch (err) {
    res.status(500).send(err.message);
  }
});

// Delete a product
app.delete("/delete/:id", async (req, res) => {
  try {
    await Product.findByIdAndDelete(req.params.id);
    res.status(204).send();
  } catch (err) {
    res.status(500).send(err.message);
  }
});

// Start server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Step 4: Create a React Frontend

1. **Go back to** `experiment8` **folder**
2. **Run**

   ```
   npm create vite@latest client
   ```

Step 5: Create/modify following files under experiment/client/src as below:

ProductList.jsx

```
import React from "react";

const ProductList = ({ products }) => (
  <table style={{ marginTop: "20px", width: "100%", borderCollapse: "collapse" }}>
    <thead>
      <tr>
        <th style={{ border: "1px solid black", padding: "5px" }}>Name</th>
        <th style={{ border: "1px solid black", padding: "5px" }}>Description</th>
        <th style={{ border: "1px solid black", padding: "5px" }}>Price</th>
      </tr>
    </thead>
    <tbody>
      {products.length === 0 ? (
        <tr>
          <td colSpan="3" style={{ textAlign: "center" }}>No products available</td>
        </tr>
      ) : (
        products.map((product) => (
          <tr key={product._id}>
            <td style={{ border: "1px solid black", padding: "5px"
}}>{product.name}</td>
            <td style={{ border: "1px solid black", padding: "5px"
```

```
}}>{product.description}</td>
            <td style={{ border: "1px solid black", padding: "5px"
}}>{product.price}</td>
          </tr>
        ))
      )}
    </tbody>
  </table>
);

export default ProductList;
```

Addproduct.jsx

```
import React, { useState } from "react";

const AddProduct = ({ fetchProducts }) => {
  const [name, setName] = useState("");
  const [description, setDescription] = useState("");
  const [price, setPrice] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await fetch("http://localhost:5000/add", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ name, description, price: Number(price) }),
      });

      if (!res.ok) throw new Error("Failed to add product");

      setName("");
      setDescription("");
      setPrice("");
      fetchProducts(); // refresh list
    } catch (err) {
      console.error(err.message);
    }
  };

  return (
    <form onSubmit={handleSubmit} style={{ marginTop: "20px" }}>
      <h3>Add New Product</h3>
      <input
        type="text"
        placeholder="Name"
        value={name}
        onChange={(e) => setName(e.target.value)}
        required
      />
      <input
        type="text"
        placeholder="Description"
        value={description}
        onChange={(e) => setDescription(e.target.value)}
        required
      />
      <input
        type="number"
        placeholder="Price"
        value={price}
        onChange={(e) => setPrice(e.target.value)}
        required
      />
      <button type="submit">Add Product</button>
    </form>
  );
};
export default AddProduct;
```

App.jsx (note: replace the whole content)

```
import React, { useState, useEffect } from "react";
import ProductList from "./ProductList";
import AddProduct from "./AddProduct";

const App = () => {
  const [products, setProducts] = useState([]);

  const fetchProducts = async () => {
    try {
      const res = await fetch("http://localhost:5000/products");
      const data = await res.json();
      setProducts(data);
    } catch (err) {
      console.error("Error fetching products:", err);
    }
  };

  useEffect(() => {
    fetchProducts();
  }, []);

  return (
    <div style={{ padding: "20px" }}>
      <h1>Product Management</h1>
      <ProductList products={products} />
      <AddProduct fetchProducts={fetchProducts} />
    </div>
  );
};

export default App;
```

App.css (note: remove the file)

Step 6: Run the Application

1. **Start the Backend Server**

   cd experiment/server; node server.js

2. **Start the React Frontend**

   cd experiment/client npm run dev

3. **Access the Application**
   - Backend API: http://localhost:5000/products
   - React App: http://localhost:5173

Outcome

1. The React app displays a list of products fetched from the /products endpoint.

   Using Postman or Browser

   If your Node.js API is running, you can fetch the products using your GET endpoint:

   Open Postman or your browser. http://localhost:5000/products

2. Adding a new product updates the list dynamically without refreshing the page. You now have a fully functional MongoDB-Node.js-Express-React app.

How to check which data are there in MongoDB

Using MongoDB Shell

1. Start the MongoDB Shell:
   - Run mongosh (for modern versions) or mongo (for older versions) in your terminal.
2. Switch to the shop Database: use shop
3. List All Collections:

   show collections

   You should see products.
4. View All Documents in the products Collection: db.products.find().pretty()

This will display all the products in a readable format.