**Stock Management Application**

**Step 1: Setting Up the Development Environment**

Before you start, make sure you have the following installed on your system:

* Node.js
* MongoDB
* npm or yarn
* Visual Studio Code or any code editor

**Step 2: Set Up the Backend (Node.js + Express + MongoDB)**

**1. Initialize the Node.js Project**

Start by creating a folder for your project and initializing a Node.js project:

bash

Copy code

mkdir stock-management-app

cd stock-management-app

npm init -y

**2. Install Dependencies**

You will need to install Express, MongoDB, and other dependencies:

bash

Copy code

npm install express mongoose cors dotenv body-parser

* express: Framework to create APIs.
* mongoose: MongoDB object modeling.
* cors: Enables cross-origin requests.
* dotenv: For managing environment variables.
* body-parser: To parse incoming request bodies.

**3. Create Folder Structure**

Create the following folder structure for your backend:

bash

Copy code

stock-management-app/

├── server/

│ ├── models/

│ │ └── Product.js

│ ├── routes/

│ │ └── products.js

│ ├── .env

│ ├── server.js

├── package.json

**4. Set Up MongoDB Model**

In the models/Product.js, define the schema for the stock products:

js

Copy code

const mongoose = require('mongoose');

const ProductSchema = new mongoose.Schema({

name: {

type: String,

required: true,

},

description: {

type: String,

required: true,

},

quantity: {

type: Number,

required: true,

default: 0,

},

price: {

type: Number,

required: true,

},

dateAdded: {

type: Date,

default: Date.now,

},

});

module.exports = mongoose.model('Product', ProductSchema);

**5. Set Up Routes for Product Management**

In the routes/products.js file, define the routes for adding, updating, and deleting products:

js

Copy code

const express = require('express');

const Product = require('../models/Product');

const router = express.Router();

// Add a new product

router.post('/', async (req, res) => {

const { name, description, quantity, price } = req.body;

const newProduct = new Product({

name,

description,

quantity,

price,

});

try {

const product = await newProduct.save();

res.json(product);

} catch (err) {

res.status(500).json({ message: err.message });

}

});

// Get all products

router.get('/', async (req, res) => {

try {

const products = await Product.find();

res.json(products);

} catch (err) {

res.status(500).json({ message: err.message });

}

});

// Update product quantity

router.put('/:id', async (req, res) => {

const { quantity } = req.body;

try {

const updatedProduct = await Product.findByIdAndUpdate(req.params.id, { quantity }, { new: true });

res.json(updatedProduct);

} catch (err) {

res.status(500).json({ message: err.message });

}

});

// Delete a product

router.delete('/:id', async (req, res) => {

try {

await Product.findByIdAndDelete(req.params.id);

res.json({ message: 'Product deleted successfully' });

} catch (err) {

res.status(500).json({ message: err.message });

}

});

module.exports = router;

**6. Set Up the Express Server**

In server.js, initialize the Express server and connect to MongoDB:

js

Copy code

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const bodyParser = require('body-parser');

require('dotenv').config();

const productRoutes = require('./routes/products');

const app = express();

const PORT = process.env.PORT || 5000;

app.use(cors());

app.use(bodyParser.json());

app.use('/api/products', productRoutes);

// Connect to MongoDB

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => {

console.log('Connected to MongoDB');

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

})

.catch((err) => {

console.error('MongoDB connection error:', err);

});

In the .env file, add the MongoDB URI:

bash

Copy code

MONGO\_URI=mongodb://localhost:27017/stockManagement

**Step 3: Set Up the Frontend (React)**

**1. Initialize React App**

In the root directory of your project, run:

bash

Copy code

npx create-react-app client

**2. Install Axios for HTTP Requests**

Navigate to the client directory and install Axios:

bash

Copy code

cd client

npm install axios

**3. Set Up React Components**

* **ProductList Component**: This component displays all the products.
* **ProductForm Component**: This component allows adding a new product.
* **ProductCard Component**: This component shows each product's details.

Here is an example of how to create the ProductList component:

src/components/ProductList.js:

js

Copy code

import React, { useEffect, useState } from 'react';

import axios from 'axios';

const ProductList = () => {

const [products, setProducts] = useState([]);

useEffect(() => {

axios.get('http://localhost:5000/api/products')

.then(response => setProducts(response.data))

.catch(error => console.log(error));

}, []);

return (

<div>

<h2>Product List</h2>

<ul>

{products.map(product => (

<li key={product.\_id}>

{product.name} - {product.quantity} in stock

</li>

))}

</ul>

</div>

);

};

export default ProductList;

**4. Create ProductForm Component**

src/components/ProductForm.js:

js

Copy code

import React, { useState } from 'react';

import axios from 'axios';

const ProductForm = () => {

const [name, setName] = useState('');

const [description, setDescription] = useState('');

const [quantity, setQuantity] = useState('');

const [price, setPrice] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

const newProduct = { name, description, quantity, price };

try {

await axios.post('http://localhost:5000/api/products', newProduct);

alert('Product added');

} catch (error) {

console.error(error);

}

};

return (

<form onSubmit={handleSubmit}>

<input type="text" placeholder="Name" onChange={(e) => setName(e.target.value)} />

<input type="text" placeholder="Description" onChange={(e) => setDescription(e.target.value)} />

<input type="number" placeholder="Quantity" onChange={(e) => setQuantity(e.target.value)} />

<input type="number" placeholder="Price" onChange={(e) => setPrice(e.target.value)} />

<button type="submit">Add Product</button>

</form>

);

};

export default ProductForm;

**5. Integrate Components in App**

In src/App.js, import and use the components:

js

Copy code

import React from 'react';

import ProductList from './components/ProductList';

import ProductForm from './components/ProductForm';

const App = () => {

return (

<div>

<h1>Stock Management App</h1>

<ProductForm />

<ProductList />

</div>

);

};

export default App;

**Step 4: Run the Application**

**1. Run the Backend**

From the root directory:

bash

Copy code

node server/server.js

**2. Run the Frontend**

Navigate to the client directory:

bash

Copy code

npm start

**Step 5: Testing and Deployment**

* Test the full functionality of adding products, viewing the list, and updating the stock quantities.
* You can deploy the backend on platforms like Heroku, and the frontend on Netlify or Vercel.