Backend and frontend

**Backend/controller/itemHandlers.js**

const Items = require("../models/item")

const handlers = {

getItems: async (req, res, next) => {

let data = [];

try{

data = await Items.find();

res.status(200).json(data);

} catch(err) {

res.status(500).json({

error: err

});

}

},

getItem: async (req, res, next) => {

try{

await Items.findOne({item\_no: req.params.item\_no})

.then((item) => {

if(item !== null)

res.json(item);

else

res.json({

"message": "No item found with item number: " + req.params.item\_no

});

});

} catch (err) {

res.status(500).json({

error: err

});

}

},

createItem: async (req, res, next) => {

await Items.create(req.body)

.then((data) => {

res.json(data);

})

.catch((err) => {

if(err.errors)

res.json({

error: ""

});

else

res.json({

error: "Item number must be unique"

});

})

},

deleteItem: async (req, res, next) => {

try{

await Items.findOne({item\_no: req.params.item\_no})

.then((item) => {

if(item !== null)

Items.deleteOne({item\_no: req.params.item\_no})

.then(async (data) => {

let items = [];

items = await Items.find();

res.json(items);

});

else

res.json({

"message": "No item found with item number: " + req.params.item\_no

});

});

} catch(err) {

res.status(500).json({

error: err

});

}

},

updateItem: async (req, res, next) => {

try{

await Items.findOne({item\_no: req.body.item\_no})

.then((item) => {

if(item !== null)

Items.updateOne({item\_no: req.body.item\_no}, req.body)

.then((data) => {

res.json({

message: "Edited the item with item number: " + req.body.item\_no + " successfully"

});

});

else

res.json({

"message": "No item found with item number: " + req.params.item\_no

});

});

} catch(err) {

res.status(500).json({

error: err

});

}

}

}

module.exports = handlers;

**models/items.js**

const { Schema, default: mongoose } = require("mongoose");

const Item = new Schema({

item\_no: {type: String, required: true, unique: true},

name: {type: String, required: true},

price: {type: Number, required: true},

quantity: {type: Number, required: true}

});

module.exports = mongoose.model("item", Item);

**server.js**

const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");

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

const handlers = require("./controller/itemHandlers");

const server = express();

server.use(cors());

server.use(bodyParser());

server.get("/", (req, res, next) => {

res.send("hello world");

})

server.get("/items", handlers.getItems);

server.get("/item/:item\_no", handlers.getItem);

server.post("/item", handlers.createItem);

server.put("/item", handlers.updateItem);

server.delete("/item/:item\_no", handlers.deleteItem);

mongoose.connect("mongodb://127.0.0.1:27017/itemsdb")

.then((d) => {

server.listen(8080, () => {console.log("Server listening at port: 8080")});

})

.catch((err) => {

console.log("ERROR: " + err);

});

**Frontend/App.js**

import './App.css';

import { Button, createTheme, Dialog, DialogActions, DialogContent, DialogTitle, Grid, OutlinedInput, Table, TableBody, TableCell, TableHead, TableRow, ThemeProvider, Typography } from '@mui/material';

import { useEffect, useState } from 'react';

import axios from 'axios';

function App() {

const theme = createTheme({

palette: {

primary: {

main: "#BE6B2C"

},

secondary: {

main: "#ffffff"

},

info: {

main: "#444444"

}

}

});

const [items, setItems] = useState([]);

const [edit, setEdit] = useState(false);

const [create, setCreate] = useState(false);

const [item, setItem] = useState(null);

//for create

const [item\_no, setItem\_no] = useState("");

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

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

const [quantity, setQuantity] = useState(0);

useEffect(() => {

axios.get("http://localhost:8080/items")

.then((res) => {

setItems(res.data);

})

.catch((err) => {

console.log(err);

})

}, []);

const handleCreate = () => {

setItem(null);

setCreate(true);

}

const createCloseHandler = () => {

setCreate(false);

}

const handleEdit = (e) => {

setItem(e);

setEdit(true);

}

const handleDelete = (item\_no) => {

console.log(item\_no)

axios.delete("http://localhost:8080/item/"+item\_no)

.then((res) => {

setItems(res.data);

})

.catch((err) => {

console.log(err);

})

}

const editCloseHandler = () => {

setEdit(false);

}

const handleEditSubmit = () => {

axios.put("http://localhost:8080/item", item)

.then((res) => {

axios.get("http://localhost:8080/items")

.then((res) => {

setItems(res.data);

editCloseHandler();

})

.catch((err) => {

console.log(err);

})

})

.catch((err) => {

console.log(err);

})

}

const handleCreateSubmit = () => {

console.log(item);

axios.post("http://localhost:8080/item", {

item\_no: item\_no,

name: name,

price: price,

quantity: quantity

})

.then((res) => {

axios.get("http://localhost:8080/items")

.then((res) => {

setItems(res.data);

createCloseHandler();

})

.catch((err) => {

console.log(err);

})

})

.catch((err) => {

console.log(err);

})

}

return (

<div className="App">

<ThemeProvider theme={theme}>

<Grid container>

<Grid item alignItems="center" className='width'>

<Typography variant="h4" style={{margin: "1em", fontWeight: "bold"}} color="primary">INVENTORY MANAGEMENT SYSTEM</Typography>

</Grid>

<Grid item className='width' alignItems="center">

<Button variant='outlined' color='primary' style={{backgroundColor: "white"}} onClick={handleCreate}>CREATE</Button>

</Grid>

<Grid item className='width' style={{padding: "3em"}} alignItems="center">

<Table style={{backgroundColor: "white", borderRadius: "8 px"}}>

<TableHead>

<TableRow>

<TableCell style={{fontWeight: "bold"}}>Item No</TableCell>

<TableCell style={{fontWeight: "bold"}}>Name</TableCell>

<TableCell style={{fontWeight: "bold"}}>Price</TableCell>

<TableCell style={{fontWeight: "bold"}}>Quantity</TableCell>

<TableCell style={{fontWeight: "bold"}}>Edit</TableCell>

<TableCell style={{fontWeight: "bold"}}>Delete</TableCell>

</TableRow>

</TableHead>

<TableBody>

{

items.map((e) => (

<TableRow>

<TableCell>{e.item\_no}</TableCell>

<TableCell>{e.name}</TableCell>

<TableCell>{e.price}</TableCell>

<TableCell>{e.quantity}</TableCell>

<TableCell>

<Button

variant="contained"

color="primary"

onClick={() => handleEdit(e)}

>

EDIT

</Button>

</TableCell>

<TableCell>

<Button variant="contained" color="error" onClick={() => handleDelete(e.item\_no)}>

DELETE

</Button>

</TableCell>

</TableRow>

))

}

</TableBody>

</Table>

</Grid>

</Grid>

<Dialog

open={edit}

onClose={editCloseHandler}

>

<DialogTitle>Edit Item</DialogTitle>

<DialogContent style={{width: "300px"}}>

{

(item !== null)?

<Grid container spacing="10px">

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="Item Number" value={item.item\_no} onChange={(e) => {

setItem({

item\_no: e.target.value,

name: item.name,

price: item.price,

quantity: item.quantity

});

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="Name" value={item.name} onChange={(e) => {

setItem({

item\_no: item.item\_no,

name: e.target.value,

price: item.price,

quantity: item.quantity

});

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="number" placeholder="Price" value={item.price} onChange={(e) => {

setItem({

item\_no: item.item\_no,

name: item.name,

price: e.target.value,

quantity: item.quantity

});

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="number" placeholder="quantity" value={item.quantity} onChange={(e) => {

setItem({

item\_no: item.item\_no,

name: item.name,

price: item.price,

quantity: e.target.value

});

}}></OutlinedInput>

</Grid>

</Grid>

:

<></>

}

</DialogContent>

<DialogActions style={{paddingRight: "1.6em", marginBottom: "1em"}}>

<Button variant='contained' color='primary' onClick={handleEditSubmit}>EDIT</Button>

<Button variant='outlined' color='info' onClick={editCloseHandler}>CANCEL</Button>

</DialogActions>

</Dialog>

<Dialog

open={create}

onClose={createCloseHandler}

>

<DialogTitle>Add Item</DialogTitle>

<DialogContent style={{width: "300px"}}>

<Grid container spacing="10px">

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="Item Number" onChange={(e) => {

setItem\_no(e.target.value);

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="text" placeholder="Name" onChange={(e) => {

setName(e.target.value);

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="number" placeholder="Price" onChange={(e) => {

setPrice(e.target.value);

}}></OutlinedInput>

</Grid>

<Grid item xs="12">

<OutlinedInput fullWidth type="number" placeholder="quantity" onChange={(e) => {

setQuantity(e.target.value);

}}></OutlinedInput>

</Grid>

</Grid>

</DialogContent>

<DialogActions style={{paddingRight: "1.6em", marginBottom: "1em"}}>

<Button variant='contained' color='primary' onClick={handleCreateSubmit}>ADD</Button>

<Button variant='outlined' color='info' onClick={createCloseHandler}>CANCEL</Button>

</DialogActions>

</Dialog>

</ThemeProvider>

</div>

);

}

export default App;

**app.css**

.App {

text-align: center;

}

.App-logo {

height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.App-logo {

animation: App-logo-spin infinite 20s linear;

}

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.App-link {

color: #61dafb;

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}