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); |
|  | } |
|  | } |

----------------------------

App.test.js

|  |
| --- |
| import { render, screen } from '@testing-library/react'; |
|  | import App from './App'; |
|  |  |
|  | test('renders learn react link', () => { |
|  | render(<App />); |
|  | const linkElement = screen.getByText(/learn react/i); |
|  | expect(linkElement).toBeInTheDocument(); |
|  | }); |

----------------------------------------------

Index.css

|  |
| --- |
| body { |
|  | margin: 0; |
|  | font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen', |
|  | 'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue', |
|  | sans-serif; |
|  | -webkit-font-smoothing: antialiased; |
|  | -moz-osx-font-smoothing: grayscale; |
|  | } |
|  |  |
|  | code { |
|  | font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New', |
|  | monospace; |
|  | } |

-----------------------------------

Index.js

|  |
| --- |
| import React from 'react'; |
|  | import ReactDOM from 'react-dom'; |
|  | import './index.css'; |
|  | import reportWebVitals from './reportWebVitals'; |
|  | import "../node\_modules/bootstrap/dist/css/bootstrap.css"; |
|  | import "../node\_modules/jquery/dist/jquery.min"; |
|  | import "../node\_modules/bootstrap/dist/js/bootstrap.min"; |
|  | import Main from './components/Main' |
|  |  |
|  | ReactDOM.render( |
|  | <React.StrictMode> |
|  | <Main /> |
|  | </React.StrictMode>, |
|  | document.getElementById('root') |
|  | ); |
|  |  |
|  | // If you want to start measuring performance in your app, pass a function |
|  | // to log results (for example: reportWebVitals(console.log)) |
|  | // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals |
|  | reportWebVitals(); |

---------------------------------------------

reportWebVitals.js

|  |
| --- |
| const reportWebVitals = onPerfEntry => { |
|  | if (onPerfEntry && onPerfEntry instanceof Function) { |
|  | import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => { |
|  | getCLS(onPerfEntry); |
|  | getFID(onPerfEntry); |
|  | getFCP(onPerfEntry); |
|  | getLCP(onPerfEntry); |
|  | getTTFB(onPerfEntry); |
|  | }); |
|  | } |
|  | }; |
|  |  |
|  | export default reportWebVitals; |

--------------------------------------------------------------

setupTest.js

import '@testing-library/jest-dom';

index.html

|  |
| --- |
| <!DOCTYPE html> |
|  | <html lang="en"> |
|  |  |
|  | <head> |
|  | <meta charset="utf-8" /> |
|  | <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" /> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1" /> |
|  | <meta name="theme-color" content="#000000" /> |
|  | <meta name="description" content="Web site created using create-react-app" /> |
|  | <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" /> |
|  | <!-- |
|  | manifest.json provides metadata used when your web app is installed on a |
|  | user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/ |
|  | --> |
|  | <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" /> |
|  | <!-- |
|  | Notice the use of %PUBLIC\_URL% in the tags above. |
|  | It will be replaced with the URL of the `public` folder during the build. |
|  | Only files inside the `public` folder can be referenced from the HTML. |
|  |  |
|  | Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will |
|  | work correctly both with client-side routing and a non-root public URL. |
|  | Learn how to configure a non-root public URL by running `npm run build`. |
|  | --> |
|  | <link href='https://fonts.googleapis.com/css?family=Averia Libre' rel='stylesheet'> |
|  | <link href='https://fonts.googleapis.com/css?family=Bellota Text' rel='stylesheet'> |
|  | <title>React App</title> |
|  | </head> |
|  |  |
|  | <body> |
|  | <noscript>You need to enable JavaScript to run this app.</noscript> |
|  | <div id="root"></div> |
|  | <!-- |
|  | This HTML file is a template. |
|  | If you open it directly in the browser, you will see an empty page. |
|  |  |
|  | You can add webfonts, meta tags, or analytics to this file. |
|  | The build step will place the bundled scripts into the <body> tag. |
|  |  |
|  | To begin the development, run `npm start` or `yarn start`. |
|  | To create a production bundle, use `npm run build` or `yarn build`. |
|  | --> |
|  | </body> |
|  |  |
|  | </html> |

---------------------------------------

manifest.json

|  |
| --- |
| { |
|  | "short\_name": "React App", |
|  | "name": "Create React App Sample", |
|  | "icons": [ |
|  | { |
|  | "src": "favicon.ico", |
|  | "sizes": "64x64 32x32 24x24 16x16", |
|  | "type": "image/x-icon" |
|  | }, |
|  | { |
|  | "src": "logo192.png", |
|  | "type": "image/png", |
|  | "sizes": "192x192" |
|  | }, |
|  | { |
|  | "src": "logo512.png", |
|  | "type": "image/png", |
|  | "sizes": "512x512" |
|  | } |
|  | ], |
|  | "start\_url": ".", |
|  | "display": "standalone", |
|  | "theme\_color": "#000000", |
|  | "background\_color": "#ffffff" |
|  | } |

-------------------------------

MODELS

Admin.model.js

|  |
| --- |
| const mongoose = require('mongoose'), Schema = mongoose.Schema; |
|  |  |
|  | const AdminSchema = new mongoose.Schema({ |
|  |  |
|  | name: { |
|  | type: String, |
|  | required: true |
|  | }, |
|  | email: { |
|  | type: String, |
|  | required: true, |
|  | unique: true |
|  | }, |
|  | phone: { |
|  | type: Number, |
|  | required: true |
|  | }, |
|  | username: { |
|  | type: String, |
|  | required: true |
|  | }, |
|  | password: { |
|  | type: String, |
|  | required: true |
|  | } |
|  | }); |
|  |  |
|  | module.exports = mongoose.model('Admin', AdminSchema); |

---------------------------------------------------

foodItem.model.js

|  |
| --- |
| const mongoose = require('mongoose'), Schema = mongoose.Schema; |
|  |  |
|  | const FoodItemSchema = new mongoose.Schema({ |
|  |  |
|  | name: { |
|  | type: String, |
|  | required: true |
|  | }, |
|  | description: { |
|  | type: String, |
|  | required: true |
|  | }, |
|  | category: { |
|  | type: String, |
|  | required: true |
|  | }, |
|  | brand: { |
|  | type: String, |
|  | required: true |
|  | }, |
|  | price: { |
|  | type: Number, |
|  | required: true |
|  | }, |
|  | containerType: { |
|  | type: String, |
|  | required: true |
|  | } |
|  | }); |
|  |  |
|  | module.exports = mongoose.model('FoodItem', FoodItemSchema); |

---------------------------------------------

DEFAULT.JSON

|  |
| --- |
| { |
|  | "url": "mongodb://localhost:27017/kitchenStoryDb", |
|  | "jwtSecret": "betapha45\_765omega" |
|  | } |

---------------------------------

ADMIN.CONTROLERS.JS

|  |
| --- |
| const Admin = require('../Models/admin.model'); |
|  | const FoodItem = require('../Models/foodItem.model'); |
|  | const bcrypt = require('bcryptjs'); |
|  | const config = require('config'); |
|  | const jwt = require('jsonwebtoken'); |
|  | const { has } = require('config'); |
|  |  |
|  |  |
|  | exports.signUpAdmin = (req, res, next) => { |
|  |  |
|  | const name = req.body.name; |
|  | const email = req.body.email; |
|  | const phone = req.body.phone; |
|  | const username = req.body.username; |
|  | const password = req.body.password; |
|  |  |
|  | //basic check |
|  | if (!name || !email || !phone || !username || !password) { |
|  | res.status(400).json({ |
|  | message: "Please fill all the fields !" |
|  | }) |
|  | } |
|  |  |
|  | var hashPassword = null; |
|  | //check for existing email |
|  | Admin.findOne({ email: email }) |
|  | .then(admin => { |
|  | if (admin) { |
|  | res.status(403).json({ |
|  | message: "Email already exists !" |
|  | }) |
|  | } |
|  |  |
|  |  |
|  | bcrypt.genSalt(10, (err, salt) => { |
|  | bcrypt.hash(password, salt, (err, hash) => { |
|  | if (err) throw err; |
|  | const newAdmin = new Admin({ name: name, email: email, phone: phone, username: username, password: hash }); |
|  | newAdmin.save() |
|  | .then(admin => { |
|  | res.status(201).json({ |
|  | message: "Signed Up Successfully !", |
|  | admin: { |
|  | id: admin.\_id, |
|  | name: admin.name, |
|  | username: admin.username |
|  | } |
|  | }) |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to signup due to error : " + err |
|  | })) |
|  | }) |
|  | }); |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to signup due to error : " + err |
|  | })) |
|  |  |
|  | } |
|  |  |
|  |  |
|  | exports.getAllAdmins = (req, res, next) => { |
|  |  |
|  | Admin.find() |
|  | .then(admins => { |
|  | if (!admins) { |
|  | res.status(404).json({ |
|  | message: "No Admins found !" |
|  | }) |
|  | } |
|  | res.status(200).json(admins); |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to get admins due to error : " + err |
|  | })) |
|  |  |
|  | } |
|  |  |
|  |  |
|  | exports.changePassword = (req, res, next) => { |
|  |  |
|  |  |
|  | const sentOldPass = req.body.sentOldPass; |
|  | const newPass = req.body.newPass; |
|  | const id = req.params.id; |
|  |  |
|  |  |
|  | if (!sentOldPass || !newPass) { |
|  | res.status(400).json({ |
|  | message: "Please fill all the fields !" |
|  | }) |
|  | } |
|  |  |
|  | Admin.findById(id) |
|  | .then(admin => { |
|  | if (!admin) { |
|  | res.status(404).json({ |
|  | message: "Admin not found !" |
|  | }) |
|  | } |
|  |  |
|  |  |
|  | // console.log(savedPass); |
|  | // console.log(sentOldPass); |
|  |  |
|  | bcrypt.compare(sentOldPass, admin.password) |
|  | .then(isMatch => { |
|  | if (!isMatch) { |
|  | res.status(400).json({ |
|  | message: "Old password is incorrect !!" |
|  | }) |
|  | } |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to change password due to error : " + err |
|  | })) |
|  |  |
|  | const name = admin.name; |
|  | const email = admin.email; |
|  | const phone = admin.phone; |
|  | const username = admin.username; |
|  |  |
|  |  |
|  | bcrypt.genSalt(10, (err, salt) => { |
|  | bcrypt.hash(newPass, salt, (err, hash) => { |
|  | if (err) throw err; |
|  |  |
|  | Admin.findByIdAndUpdate(id, { |
|  | name: name, |
|  | email: email, |
|  | phone: phone, |
|  | username: username, |
|  | password: hash |
|  | }, { new: true }) |
|  | .then(admin => { |
|  | if (!admin) { |
|  | res.status(404).json({ |
|  | message: "Admin not found !" |
|  | }) |
|  | } |
|  | res.status(200).json({ |
|  | message: "Password changed successfully !!" |
|  | }); |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to update password due to error : " + err |
|  | })) |
|  | }); |
|  | }) |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to signup due to error : " + err |
|  | })) |
|  | } |

-------------------------------------------------------------

auth.controller.js

|  |
| --- |
| const Admin = require('../Models/admin.model'); |
|  | const bcrypt = require('bcryptjs'); |
|  | const config = require('config'); |
|  | const jwt = require('jsonwebtoken'); |
|  |  |
|  |  |
|  | exports.loginAdmin = (req, res, next) => { |
|  |  |
|  | const email = req.body.email; |
|  | const password = req.body.password; |
|  |  |
|  | //basic check |
|  | if (!email || !password) { |
|  | res.status(400).json({ |
|  | message: "Please fill all the fields !" |
|  | }) |
|  | } |
|  |  |
|  | //Check if user with sent email exist |
|  | Admin.findOne({ email: email }) |
|  | .then(admin => { |
|  | if (!admin) { |
|  | res.status(400).json({ |
|  | message: "Admin doesn't exists !" |
|  | }) |
|  | } |
|  |  |
|  | bcrypt.compare(password, admin.password) |
|  | .then(isMatch => { |
|  | if (!isMatch) { |
|  | res.status(400).json({ |
|  | message: "Invalid Credentials !!" |
|  | }) |
|  | } |
|  |  |
|  | jwt.sign({ |
|  | id: admin.\_id |
|  | }, config.get('jwtSecret'), { expiresIn: 3600 }, (err, token) => { |
|  | if (err) throw err; |
|  | res.status(201).json({ |
|  | token: token, |
|  | admin: { |
|  | id: admin.\_id, |
|  | name: admin.name, |
|  | username: admin.username |
|  | } |
|  | }); |
|  | }) |
|  |  |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to login due to error : " + err |
|  | })) |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to login due to error : " + err |
|  | })) |
|  |  |
|  | } |

-----------------------------------------------

foodItem.controllers.js

|  |
| --- |
| const FoodItem = require('../Models/foodItem.model'); |
|  | const foodItem = require('../routes/foodItem'); |
|  |  |
|  | exports.addFoodItem = (req, res, next) => { |
|  |  |
|  | const name = req.body.name; |
|  | const description = req.body.description; |
|  | const category = req.body.category; |
|  | const brand = req.body.brand; |
|  | const price = req.body.price; |
|  | const containerType = req.body.containerType; |
|  |  |
|  | if (!name || !description || !category || !brand || !containerType || price < 0) { |
|  | res.status(400).json({ |
|  | message: "Please fill all the fields and check that price is not negative !" |
|  | }) |
|  | } |
|  |  |
|  | const newFoodItem = new FoodItem({ |
|  | name: name, |
|  | description: description, |
|  | category: category, |
|  | brand: brand, |
|  | price: price, |
|  | containerType: containerType |
|  | }); |
|  |  |
|  | newFoodItem.save() |
|  | .then(foodItem => { |
|  | res.status(201).json({ |
|  | message: "Food item added Successfully !", |
|  | foodItem: { |
|  | name: foodItem.name, |
|  | category: foodItem.category, |
|  | brand: foodItem.brand |
|  | } |
|  | }) |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to add food item due to error : " + err |
|  | })) |
|  |  |
|  | }; |
|  |  |
|  | exports.getFoodItemById = (req, res, next) => { |
|  |  |
|  | const id = req.params.id; |
|  |  |
|  | FoodItem.findById(id) |
|  | .then(foodItem => { |
|  | if (!foodItem) { |
|  | res.status(404).json({ |
|  | message: "Food item not found !" |
|  | }) |
|  | } |
|  | res.status(200).json(foodItem); |
|  | }) |
|  | .catch(err => { |
|  | return res.status(500).json({ |
|  | message: "Unable to find food item due to error : " + err |
|  | }) |
|  | }) |
|  | } |
|  |  |
|  |  |
|  | exports.deleteFoodItemById = (req, res, next) => { |
|  |  |
|  | const id = req.params.id; |
|  |  |
|  | FoodItem.findById(id) |
|  | .then(foodItem => { |
|  | if (!foodItem) { |
|  | res.status(404).json({ |
|  | message: "Food item not found !" |
|  | }) |
|  | } |
|  | return FoodItem.findByIdAndRemove(id); |
|  | }) |
|  | .then(result => { |
|  | res.status(200).json({ |
|  | message: "Food item with id: " + id + " deleted successfully !" |
|  | }) |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to delete food item due to error : " + err |
|  | })) |
|  | }; |
|  |  |
|  |  |
|  | exports.updateFoodItemById = (req, res, next) => { |
|  |  |
|  | const id = req.params.id; |
|  | const name = req.body.name; |
|  | const description = req.body.description; |
|  | const category = req.body.category; |
|  | const brand = req.body.brand; |
|  | const price = req.body.price; |
|  | const containerType = req.body.containerType; |
|  |  |
|  |  |
|  |  |
|  | if (!name || !description || !category || !brand || !containerType || price < 0) { |
|  | res.status(400).json({ |
|  | message: "Please fill all the fields and check that price is not negative !" |
|  | }) |
|  | } |
|  |  |
|  | FoodItem.findById(id) |
|  | .then(foodItem => { |
|  | if (!foodItem) { |
|  | res.status(404).json({ |
|  | message: "Food item not found !" |
|  | }) |
|  | } |
|  |  |
|  | return FoodItem.findByIdAndUpdate(id, { |
|  | name: name, |
|  | description: description, |
|  | category: category, |
|  | brand: brand, |
|  | price: price, |
|  | containerType: containerType |
|  | }, { new: true }); |
|  | }) |
|  | .then(foodItem => { |
|  | if (!foodItem) { |
|  | res.status(404).json({ |
|  | message: "Food item not found !" |
|  | }) |
|  | } |
|  |  |
|  | res.status(200).json({ |
|  | message: "Updation Successful !!", |
|  | foodItem: foodItem |
|  | }); |
|  | }) |
|  | .catch(err => res.status(500).json({ |
|  | message: "Unable to update food item due to error : " + err |
|  | })) |
|  |  |
|  | }; |
|  |  |
|  |  |
|  | exports.getAllFoodItems = (req, res, next) => { |
|  |  |
|  | FoodItem.find() |
|  | .then(foodItems => { |
|  | if (!foodItems) { |
|  | res.status(404).json({ |
|  | message: "Food items not found !" |
|  | }) |
|  | } |
|  | res.status(200).json(foodItems); |
|  | }) |
|  | .catch(err => { |
|  | if (err) { |
|  | return res.status(500).json({ |
|  | message: "Unable to find food items due to error : " + err |
|  | }) |
|  | } |
|  | }) |
|  | } |
|  |  |
|  |  |
|  | exports.getFoodItemsByFilter = (req, res, next) => { |
|  |  |
|  | const category = req.body.category; |
|  | const maxPrice = req.body.maxPrice; |
|  | const containerType = req.body.containerType; |
|  |  |
|  | FoodItem.find({ |
|  | category: category, |
|  | price: { $lt: maxPrice }, |
|  | containerType: containerType |
|  | }) |
|  | .then(foodItems => { |
|  | if (foodItems.length <= 0) { |
|  | res.status(404).json({ |
|  | message: "Food items not found !" |
|  | }) |
|  | } |
|  | res.status(200).json(foodItems); |
|  | }) |
|  | .catch(err => { |
|  | if (err) { |
|  | res.status(500).json({ |
|  | message: "Unable to find food items due to error : " + err |
|  | }) |
|  | } |
|  | }) |
|  | } |

-------------------------------------------------------------------------------------

admin.js

|  |
| --- |
| const authMiddleware = require('../middleware/auth'); |
|  |  |
|  | module.exports = function (app) { |
|  |  |
|  | const adminController = require('../controllers/admin.controller'); |
|  |  |
|  | // signup |
|  | app.post("/admin/signup", adminController.signUpAdmin); |
|  |  |
|  | app.get("/admins", adminController.getAllAdmins); |
|  |  |
|  | app.put("/admin/changepassword/:id", authMiddleware, adminController.changePassword); |
|  |  |
|  | } |

---------------------------------------------------

auth.js

|  |
| --- |
| module.exports = function (app) { |
|  |  |
|  | const authController = require('../controllers/auth.controller'); |
|  |  |
|  | //login |
|  | app.post("/auth/admin/login", authController.loginAdmin); |
|  |  |
|  | } |

---------------------------------------

foodItems.js

|  |
| --- |
| const authMiddleware = require('../middleware/auth'); |
|  |  |
|  | module.exports = function (app) { |
|  |  |
|  | const foodItemController = require('../controllers/foodItem.controller'); |
|  |  |
|  |  |
|  | // \*\*\*\*\* Normal CRUD operations\*\*\*\*\* |
|  |  |
|  | // add food item |
|  | app.post("/foodItem", authMiddleware, foodItemController.addFoodItem); |
|  |  |
|  | // delete food item by id |
|  | app.delete("/foodItem/:id", authMiddleware, foodItemController.deleteFoodItemById); |
|  |  |
|  | // get food item by id |
|  | app.get("/foodItem/:id", foodItemController.getFoodItemById); |
|  |  |
|  | // update food item by id |
|  | app.put("/foodItem/:id", authMiddleware, foodItemController.updateFoodItemById); |
|  |  |
|  | //get all the items in the db |
|  | app.get("/foodItems", foodItemController.getAllFoodItems); |
|  |  |
|  | // \*\*\*\*\*\* Advance searches \*\*\*\*\* |
|  | app.post("/foodItems/filter", foodItemController.getFoodItemsByFilter); |
|  |  |
|  | } |

-----------------------------------------------------------