MERN – E-commerce website

# List all hidden files using command line

Ls -a

# Select multiple same word => command + D

# Create react arrow functional component using VS code extension

rafce

# Set default values of props

// set default values for props

Rating.defaultProps ={

color: '#f8e825'

}

# Validate props

//validate proptypes

Rating.proTypes = {

value: PropTypes.number.isRequired,

text: PropTypes.string.isRequired,

color: PropTypes.string

}

# Dev tools – Nodemon and concurrently

Run both server and client together using npm package Nodemon and Concurrently

"scripts": {

"start": "node server/server",

"server": "nodemon server/server",

"client": "npm start --prefix client",

"dev": "concurrently \"npm run server\" \"npm run client\""

}

# Mongo setup

1. Create mongodb account
2. Create cluster
3. Create user using database access tab
4. Create Database
5. Set IP address using network access tab
6. Click on connect cluster>select mogo compass> copy connection url
7. Open mongodb compass(download separately and open) and paste copied url from 6> change password(db password) and test with database name.
8. Select connect cluster again> select connect you application> copy connection string
9. Put connection string from 8 to the .env file of your project

# Connecting to the DB using mongoosejs

import mongoose, { connect } from 'mongoose'

const connectDB = async () => {

try{

const conn = await mongoose.connect(process.env.MONGO\_URI, {

useUnifiedTopology: true,

useNewUrlParser: true,

useCreateIndex: true

})

console.log(`MongoDB connected: ${conn.connection.host} `)

} catch (error){

console.error(`Error: ${ error.message}`)

process.exit(1)

}

}

export default connectDB

# express-async-handler

middleware for handling exceptions inside of async express route and passing them to your express error handlers.

Middleware => Middleware is a function which has access to any request and response body.

# API Custom error message

// Route Not Found error message - custom

const notFound = (req, res, next) => {

const error = new Error(`Not Found - ${req.originalUrl}`)

res.status(404)

next(error)

}

// Custom error message json response

const errorHandler = (err, req, res, next) => {

const statusCode = res.statusCode === 200 ? 500 : res.statusCode

res.status(statusCode)

res.json({

message: err.message,

stack: process.env.NODE\_ENV === 'production' ? null : err.stack

})

}

export {notFound, errorHandler}

# Redux

Component

Server

Actions

Reducer

#React-redux steps to implement

1. Install following packages

*npm i redux react-redux redux-thunk redux-devtools-extension*

1. Create Store file in the root

import {createStore, combineReducers, applyMiddleware} from 'redux'

import thunk from 'redux-thunk'

import {composeWithDevTools} from 'redux-devtools-extension'

import {productListReducer} from './reducers/productReducers'

const reducer = combineReducers({

productList: productListReducer

})

const initialState = {}

const middleware = [thunk]

const store = createStore(reducer, initialState, composeWithDevTools(applyMiddleware(...middleware)))

export default store

1. Create reducer

import {PRODUCT\_LIST\_REQUEST, PRODUCT\_LIST\_SUCCESS, PRODUCT\_LIST\_FAIL} from '../constants/productConstants'

// product lists reducer

export const productListReducer = (state = {products: []}, action) => {

switch(action.type){

case PRODUCT\_LIST\_REQUEST:

return {loading: true, products: []}

case PRODUCT\_LIST\_SUCCESS:

return {loading: false, products: action.payload}

case PRODUCT\_LIST\_FAIL:

return {loadingL: false, error: action.payload }

default:

return state

}

}

1. Add reducer in to store

import {productListReducer} from './reducers/productReducers'

const reducer = combineReducers({

productList: productListReducer

})

1. Create constants

export const PRODUCT\_LIST\_REQUEST = 'PRODUCT\_LIST\_REQUEST'

export const PRODUCT\_LIST\_SUCCESS = 'PRODUCT\_LIST\_SUCCESS'

export const PRODUCT\_LIST\_FAIL = 'PRODUCT\_LIST\_FAIL'

1. Create Actions

import {PRODUCT\_LIST\_REQUEST, PRODUCT\_LIST\_SUCCESS, PRODUCT\_LIST\_FAIL} from '../constants/productConstants'

import axios from 'axios'

export const listProducts = () => async (dispatch) => {

try {

dispatch({type: PRODUCT\_LIST\_REQUEST})

const {data} = await axios.get('/api/products')

dispatch({

type: PRODUCT\_LIST\_SUCCESS,

payload: data

})

} catch (error) {

dispatch({

type: PRODUCT\_LIST\_FAIL,

payload: error.response && error.response.data.message ? error.response.data.message : error.message

})

}

}

1. finally call actions.

# Saving token in postman

# Something is already running on port 3000, but isn’t?

Kill any running port on macOS

*npx kill-port 3000*