**Waves**

This project uses nodejs, mongo db, React js

* First we will create server(i.e. Node js)

Npm init 🡪 run the default

Npm install –g nodemon

Create two folders – client, server

Install package – see package.json

Inside server/server.js (entry point of server)

Package.json – start: “node server/server.json”

COMMANDS RUNNING

Mongo db – bin – mongod, mongo, robomongo

Server – waves – npm run server

Server: User models (server/models/user.js)

const mongoose = require("mongoose");

const userSchema = mongoose.Schema({

  email: {

    type: String,

    required: true,

    trim: true,

    unique: 1

  },

  password: {

    type: String,

    required: true,

    minlength: 5

  },

  name: {

    type: String,

    required: true,

    maxlength: 100

  },

  cart: {

    type: Array,

    default: []

  },

  history: {

    type: Array,

    default: []

  },

  role: {

    type: Number,

    default: 0

  },

  token: {

    type: String

  }

});

const User = mongoose.model("User", userSchema);

module.exports = { User };

Generate hash password – using bcrypt

Path – server/models/user.js

const mongoose = require("mongoose");

const bcrypt = require("bcrypt");

const SALT\_I = 10;

const userSchema = mongoose.Schema({

  email: {

    type: String,

    required: true,

    trim: true,

    unique: 1

  },

  password: {

    type: String,

    required: true,

    minlength: 5

  },

  name: {

    type: String,

    required: true,

    maxlength: 100

  },

  lastname: {

    type: String,

    required: true,

    maxlength: 100

  },

  cart: {

    type: Array,

    default: []

  },

  history: {

    type: Array,

    default: []

  },

  role: {

    type: Number,

    default: 0

  },

  token: {

    type: String

  }

});

userSchema.pre("save", function(next) {

  var user = this;

  bcrypt.genSalt(SALT\_I, function(err, salt) {

    if (err) return next(err);

    bcrypt.hash(user.password, salt, function(err, hash) {

      if (err) return next(err);

      user.password = hash;

      next();

    });

  });

});

// when user has modified password

app.post("/api/users/register", (req, res) => {

  const user = new User(req.body);

  if (user.isModified("password")) { // part of mongo db

    user.save((err, doc) => {

      if (err) return res.json({ success: false, err });

      res.status(200).json({

        success: true,

        userdata: doc

      });

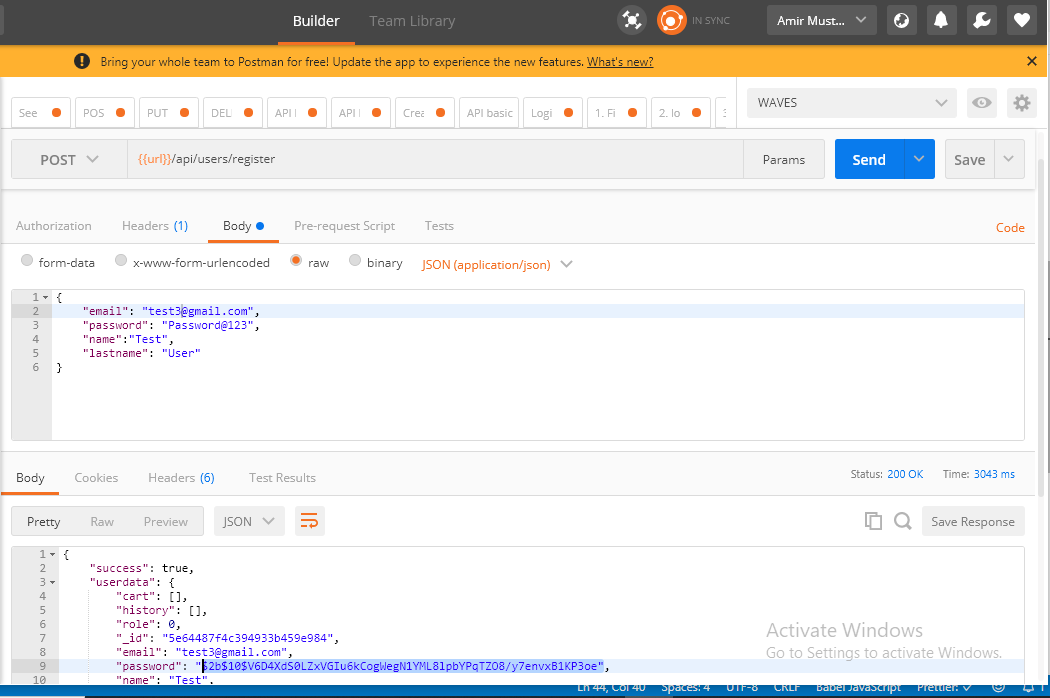
    });

  } else {

    next();

  }

});



Eg2: Brycrypt hashing

Resource - <https://www.npmjs.com/package/bcrypt>

const bcrypt **=** require('bcrypt');

const saltRounds **=** 10;

const myPlaintextPassword **=** 's0/\/\P4$$w0rD';

const someOtherPlaintextPassword **=** 'not\_bacon';

**To hash a password:**

Technique 1 (generate a salt and hash on separate function calls):

bcrypt.genSalt(saltRounds, function(err, salt) {

    bcrypt.hash(myPlaintextPassword, salt, function(err, hash) {

*// Store hash in your password DB.*

    });

});

#### To check a password:

*// Load hash from your password DB.*

bcrypt.compare(myPlaintextPassword, hash, function(err, result) {

*// result == true*

});

bcrypt.compare(someOtherPlaintextPassword, hash, function(err, result) {

*// result == false*

});

* If we are in a function inside function

User = this

User. // use like this

But when we are inside single level function this is ok.

Register Postman

**Server/server.js**

app.post("/api/users/register", (req, res) => {

  const user = new User(req.body);

  if (user.isModified("password")) {

    user.save((err, doc) => {

      if (err) return res.json({ success: false, err });

      res.status(200).json({

        success: true,

        userdata: doc

      });

    });

  } else {

    next();

  }

});

**Models/user.js**

userSchema.pre("save", function(next) {

  var user = this;

  bcrypt.genSalt(SALT\_I, function(err, salt) {

    if (err) return next(err);

    bcrypt.hash(user.password, salt, function(err, hash) {

      if (err) return next(err);

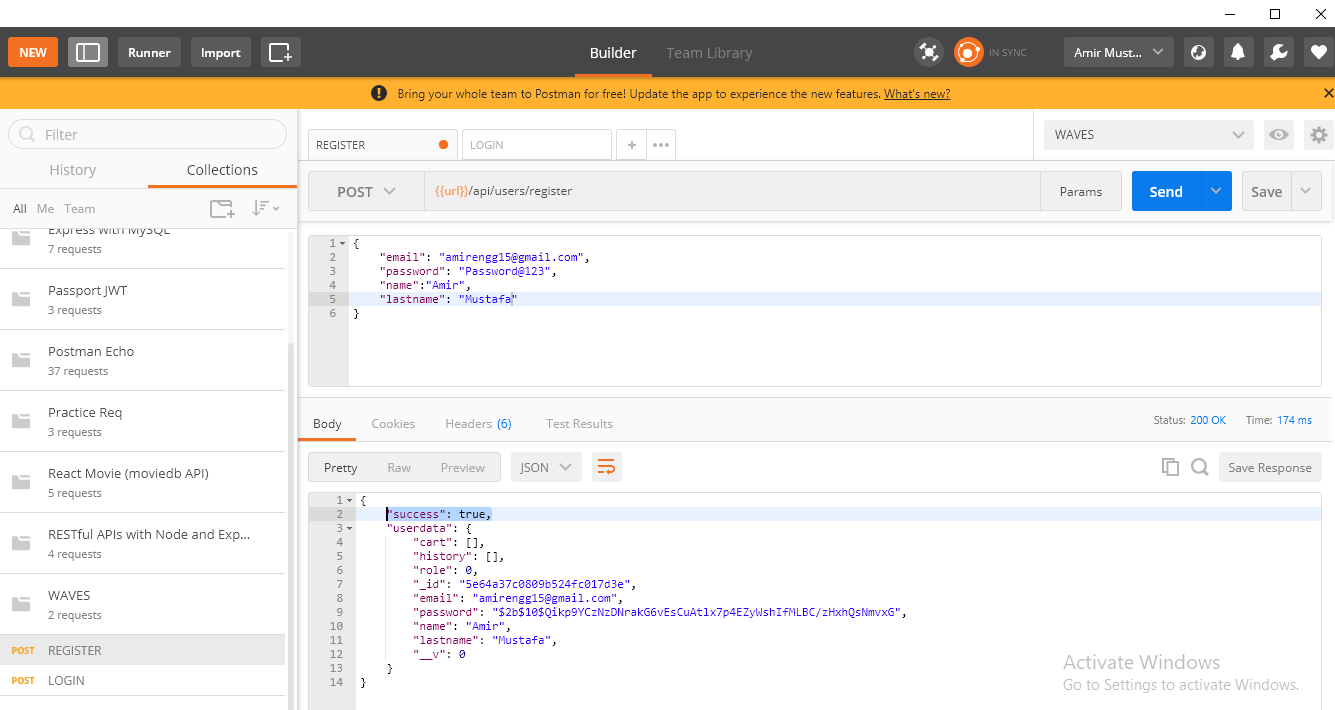
      user.password = hash;

      next();

    });

  });

});



Login

Server/server.js

app.post("/api/users/login", (req, res) => {

  User.findOne({ email: req.body.email }, (err, user) => {

    if (!user)

      return res.json({

        loginSuccess: false,

        message: "Auth failed, email not found"

      });

    // we are using user.comparePassword and not User.comparePassword, so email id already found from mongodb,

    // therefore in funcn this.password gives right saved hash password

    user.comparePassword(req.body.password, (err, isMatch) => {

      if (!isMatch)

        return res.json({

          loginSuccess: false,

          message: "Wrong password"

        });

      user.generateToken((err, user) => {

        if (err) return res.status(400).send(err);

        res

          .cookie("w\_auth", user.token)

          .status(200)

          .json({ loginSuccess: true });

      });

    });

  });

});

Models/user.js

// Method for comparing password

userSchema.methods.comparePassword = function(candidatePassword, cb) {

  bcrypt.compare(candidatePassword, this.password, function(err, isMatch) {

    if (err) return cb(err);

    cb(null, isMatch);

  });

};

// generate token

userSchema.methods.generateToken = function(cb) {

  var user = this;

// strong token = userid + env secret password

var token = jwt.sign(user.\_id.toHexString(), process.env.SECRET);

  user.token = token;

  user.save(function(err, user) {

    if (err) return cb(err);

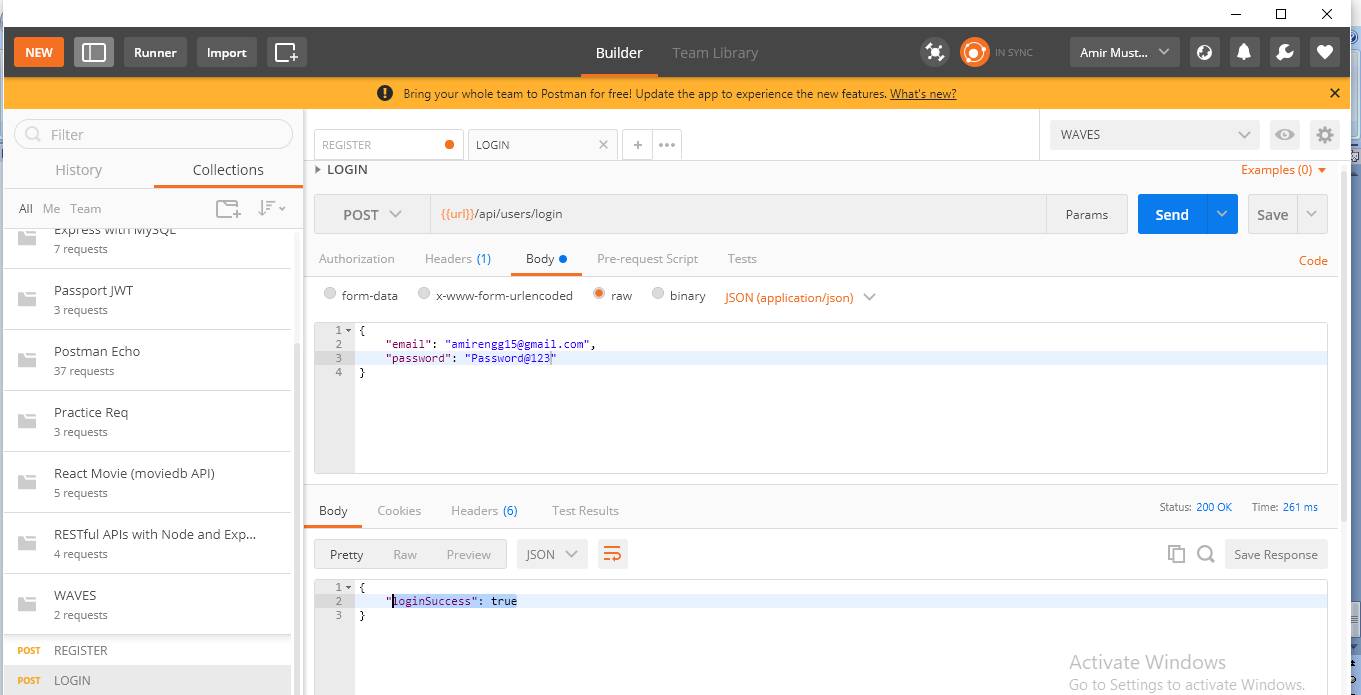
    cb(null, user);

  });

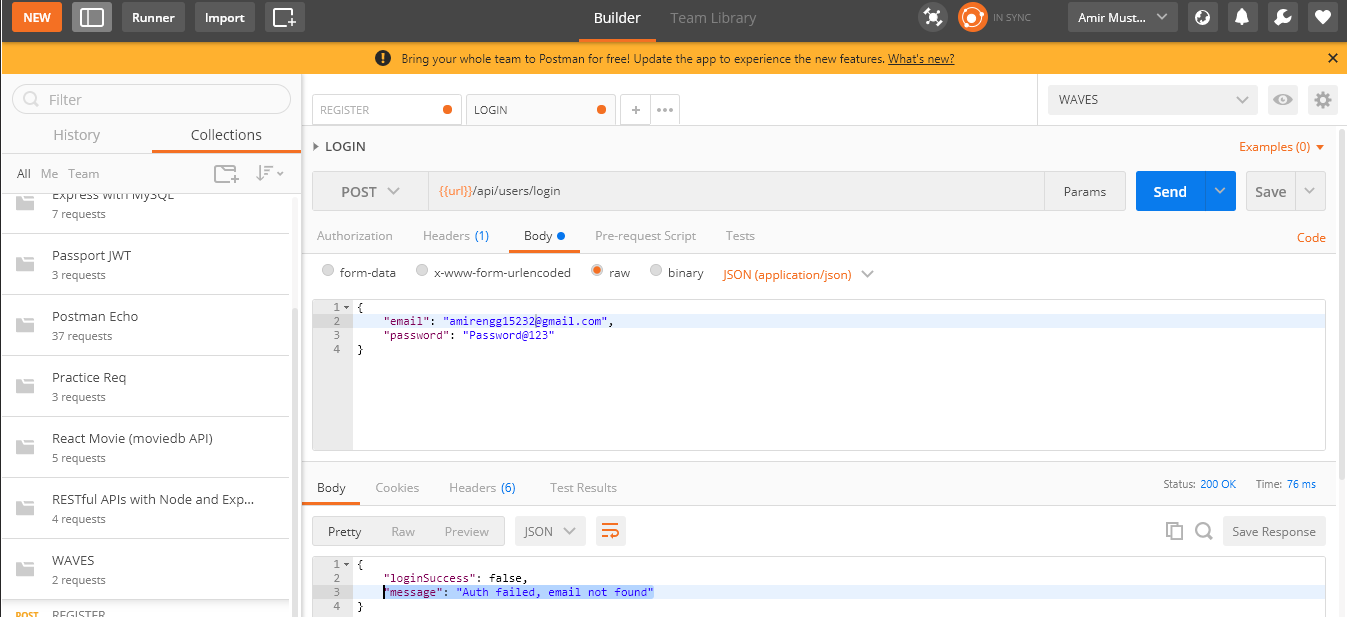
};

const User = mongoose.model("User", userSchema);

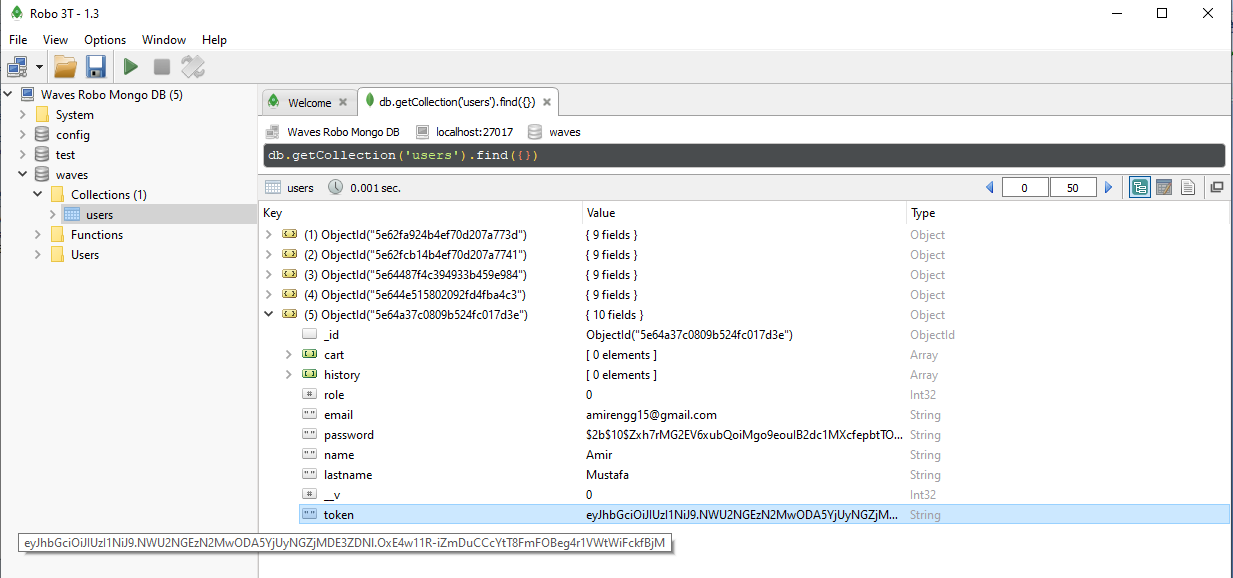
module.exports = { User };



For wrong email



DB



MIddlewares - <https://expressjs.com/en/guide/using-middleware.html>

Use

// Regular route

app.get("/api/users/auth", (req, res)=>{

})

// Route with middleware

server/server.js

const { auth } from “/api/users/auth”;

app.get("/api/users/auth", auth, (req, res)=>{

});

server/middleware/auth.js

const { User } = require("../models/user");

let auth = (req, res, next) => {

// we wite some logic here

// if pass we write next(), otherwise some message

};

module.exports = { auth };

* So in any route if auth middleware is added we check if it has valid tokens. Tokens are generated once we have used login request route.

Server/server.js

app.get("/api/users/auth", auth, (req, res) => {

// this route is used only when middleware pass – i.e. it has next()

  res.status(200).json({

    user: req.user

  });

});

Server/middleware/auth.js

const { User } = require("../models/user");

let auth = (req, res, next) => {

  let token = req.cookies.w\_auth;

  // custom method

  User.findByToken(token, (err, user) => {

    if (err) throw err;

    if (!user)

      return res.json({

        isAuth: false,

        error: true

      });

    req.token = token; // if validation pass, append token in request

    req.user = user;

    next(); // writing this is important to use route

  });

};

module.exports = { auth };

server/models/user.js

userSchema.statics.findByToken = function(token, cb) {

  var user = this;

  jwt.verify(token, process.env.SECRET, function(err, decode) {

    // decode has \_id of mongodb

    user.findOne({ \_id: decode, token: token }, function(err, user) {

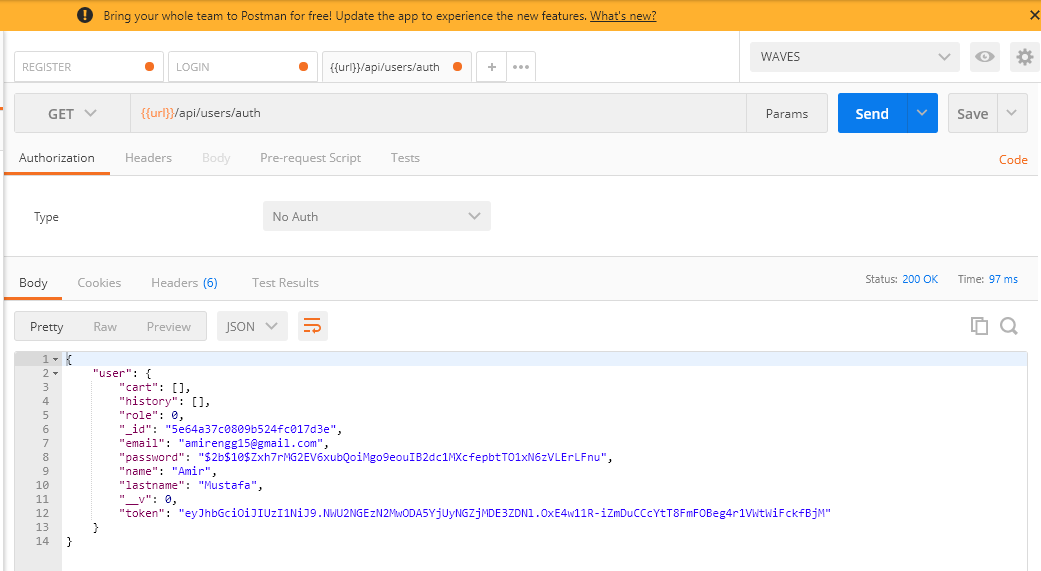
      if (err) return cb(err);

      cb(null, user);

    });

  });

};



app.get("/api/users/auth", auth, (req, res) => {

  res.status(200).json({

    // req.user has full user object

    isAdmin: req.user.role === 1 ? true : false,

    isAuth: true,

    email: req.user.email,

    name: req.user.name,

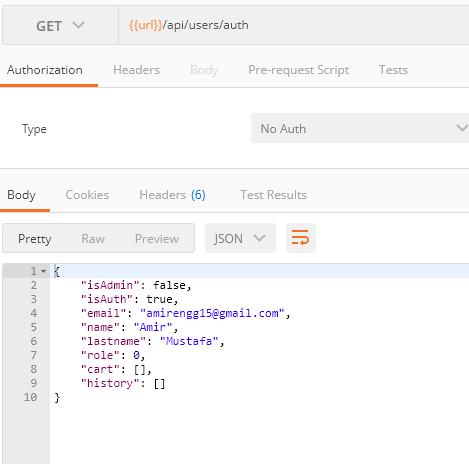
    lastname: req.user.lastname,

    role: req.user.role,

    cart: req.user.cart,

    history: req.user.history

  });



Logout – empty the token from mongo database

Server/server.js

// logout case - empty the token, auth middleware used as login is must for logout

app.get("/api/user/logout", auth, (req, res) => {

  User.findOneAndUpdate({ \_id: req.user.\_id }, { token: "" }, (err, doc) => {

    if (err) return res.json({ success: false, err });

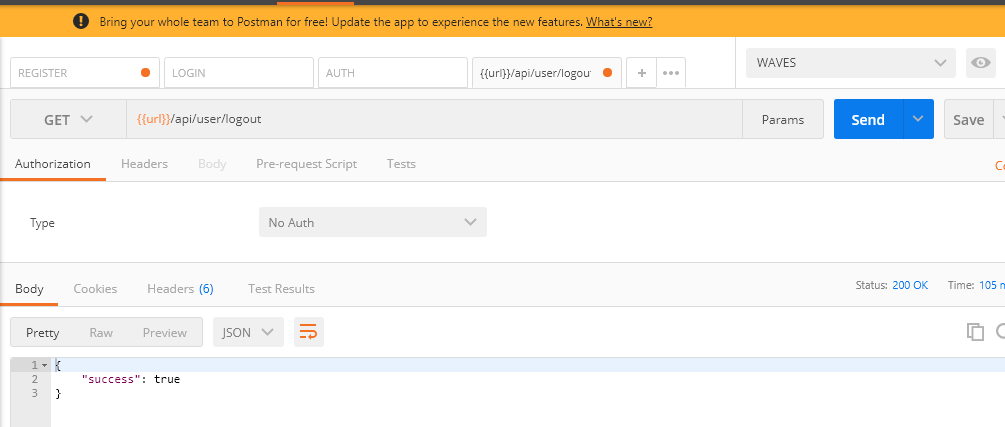
    return res.status(200).send({

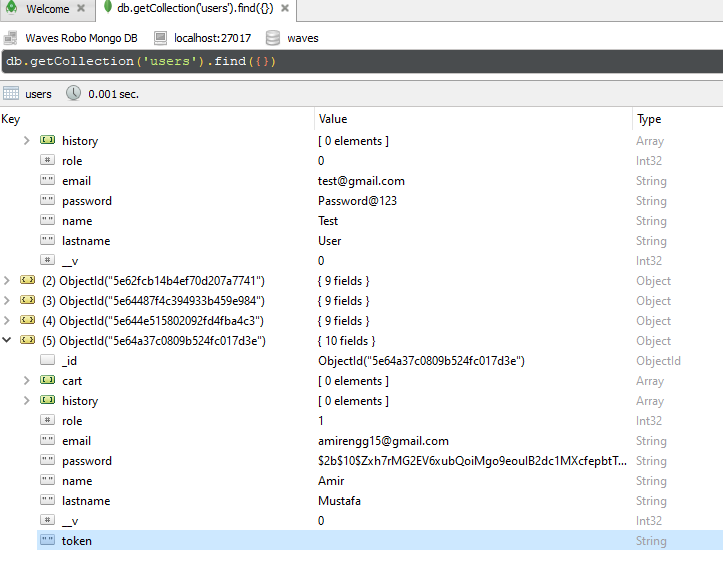
      success: true

    });

  });

});





Woods and Brand (Let’s see woods)

Server/server.js

//=============================================

//              BRAND

//=============================================

app.post("/api/product/brand", auth, (req, res) => {

  const brand = new Brand(req.body);

  brand.save((err, doc) => {

    if (err) return res.json({ success: false, err });

    res.send(200).json({

      success: true,

      brand: doc

    });

  });

});

Models/wood.js

const mongoose = require("mongoose");

const woodSchema = mongoose.Schema({

  name: {

    required: true,

    type: String,

    unique: 1,

    maxlength: 100

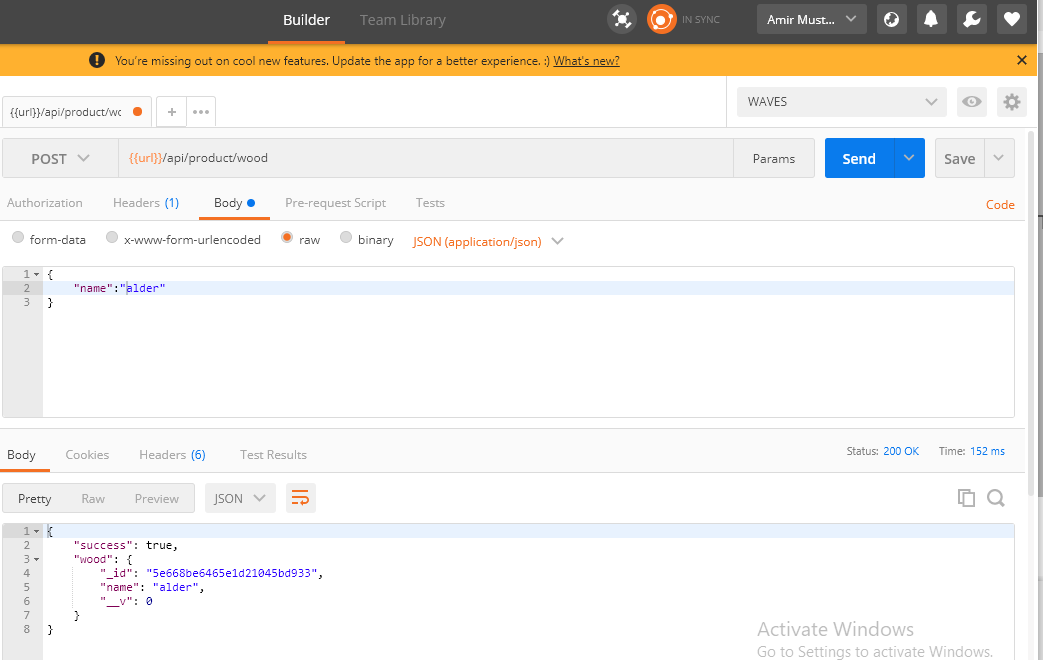
  }

});

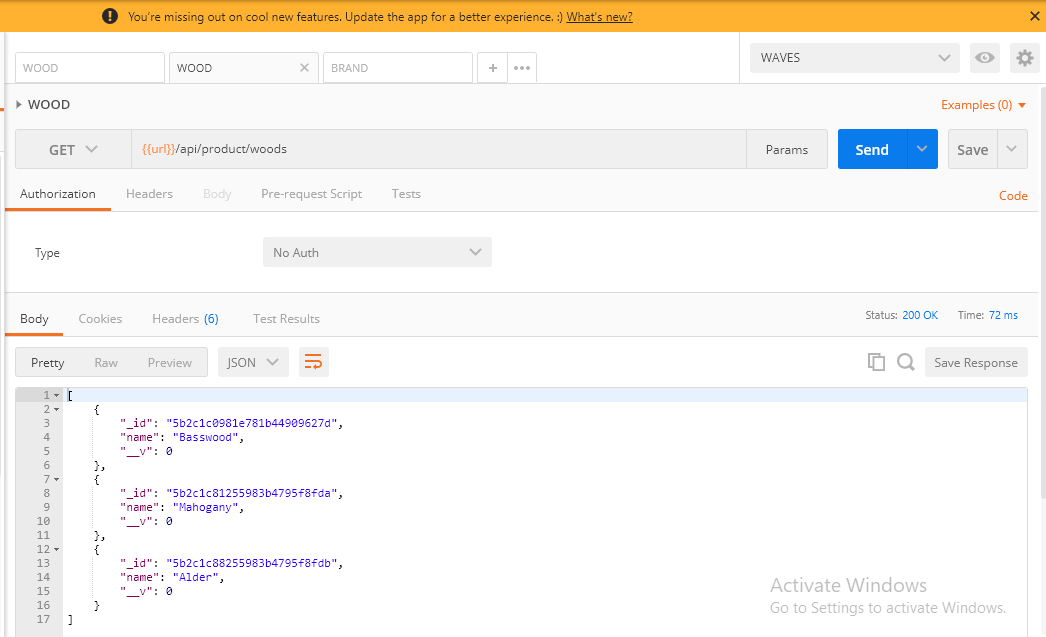
const Wood = mongoose.model("Wood", woodSchema);

module.exports = { Wood };

INSERT WOOD



GET WOODS



Products:

1. Insert product – POST req

Server/server.js

app.post("/api/product/article", auth, admin, (req, res) => {

  const product = new Product(req.body);

  product.save((err, doc) => {

    if (err) return res.json({ success: false, err });

    res.status(200).json({

      success: true,

      article: doc

    });

  });

});

Server/models/product.js

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const productSchema = mongoose.Schema(

  {

    name: {

      required: true,

      type: String,

      unique: 1,

      maxlength: 100

    },

    description: {

      required: true,

      type: String,

      maxlength: 100000

    },

    price: {

      required: true,

      type: Number,

      maxlength: 255

    },

    brand: {

      type: Schema.Types.ObjectId,

      ref: "Brand", // for making reference

      required: true

    },

    shipping: {

      required: true,

      type: Boolean

    },

    available: {

      required: true,

      type: Boolean

    },

    wood: {

      type: Schema.Types.ObjectId,

      ref: "Wood",// for making reference

      required: true

    },

    frets: {

      required: true,

      type: Number

    },

    sold: {

      type: Number,

      maxlength: 100,

      default: 0

    },

    publish: {

      required: true,

      type: Boolean

    },

    images: {

      type: Array,

      default: []

    }

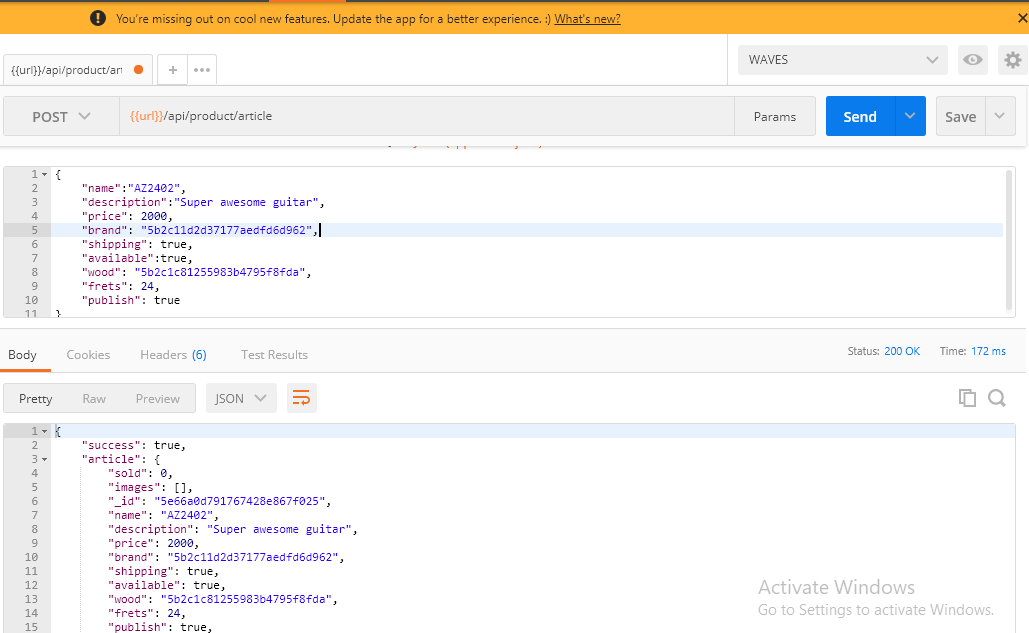
  },

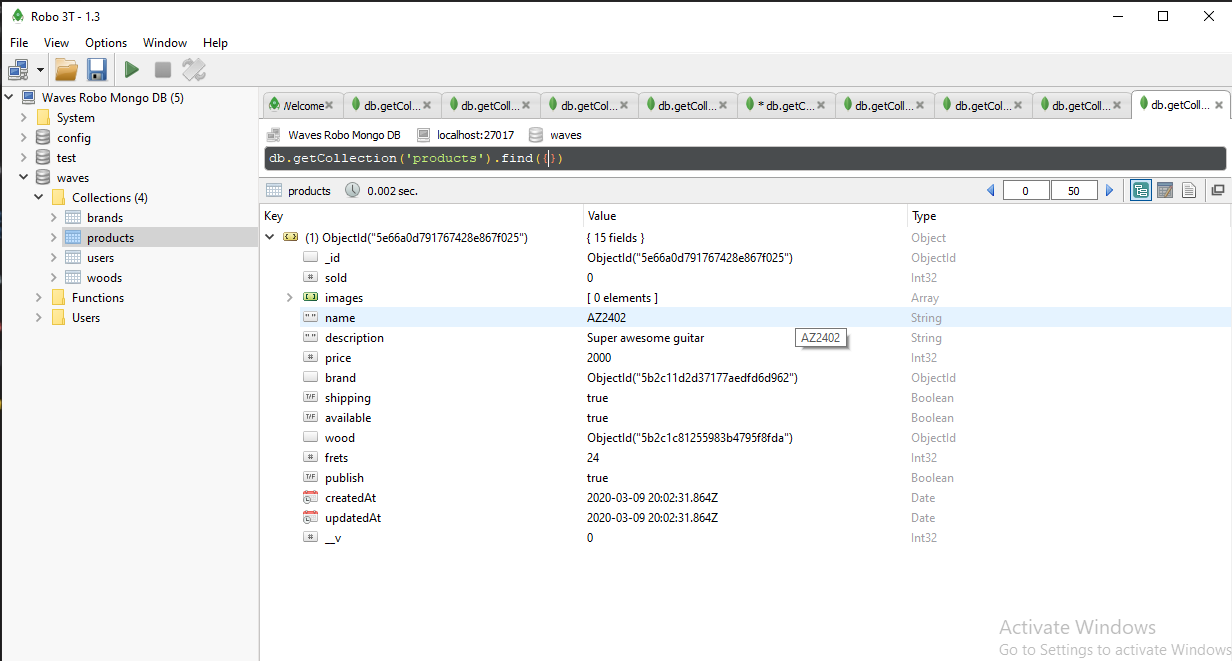
  { timestamps: true }```// auto adds created at and updated at in the end of document

);

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

module.exports = { Product };





1. Get product by id

app.get("/api/product/articles\_by\_id", (req, res) => {

  let type = req.query.type;

  let items = req.query.id;

  if (type === "array") {

    let ids = req.query.id.split(",");

    items = [];

    items = ids.map(item => {

      return mongoose.Types.ObjectId(item);

    });

  }

  // $in check single or multiple record in mongo db

  Product.find({ \_id: { $in: items } })

    .populate("brand") // have made reference in model above

    .populate("wood") // get that collection object as well based on id

    .exec((err, docs) => {

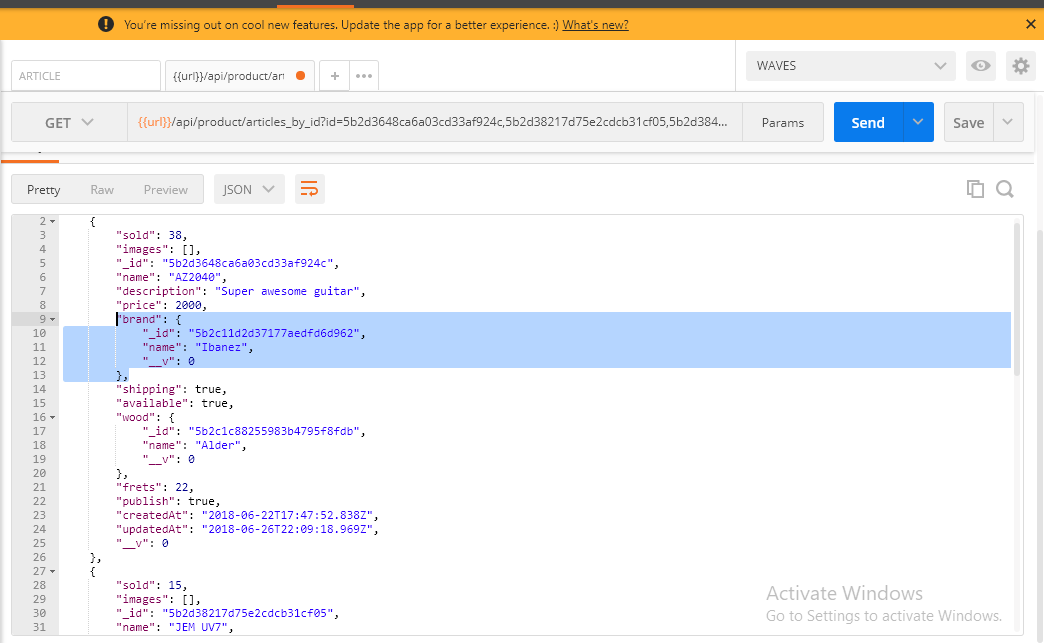
      if (err) return res, json({ success: false, err });

      return res.status(200).send(docs);

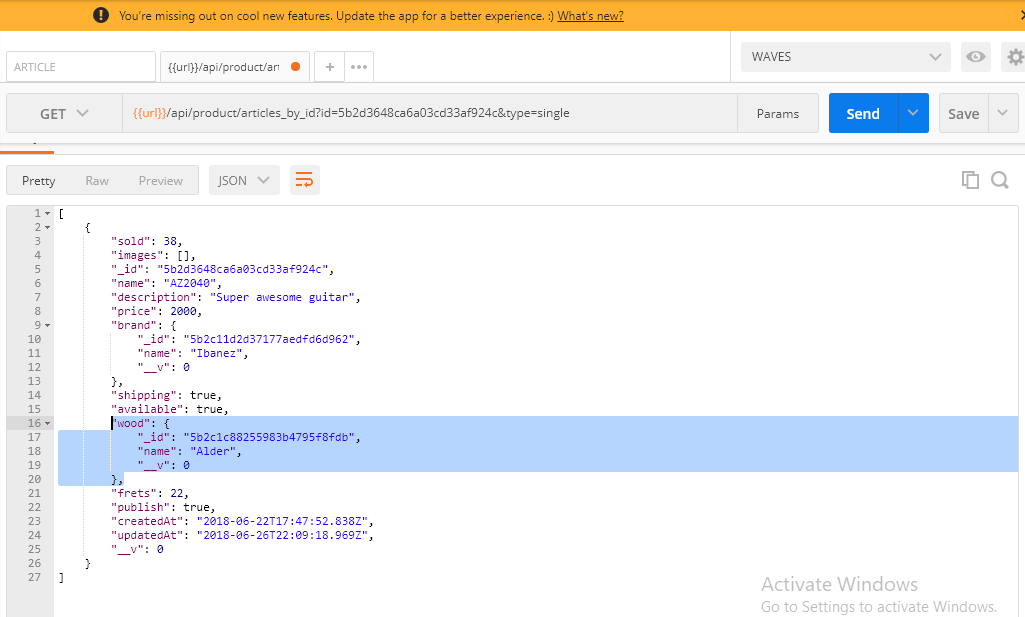
    });

});

**Request (query string) -** {{url}}/api/product/articles\_by\_id?id=5b2d3648ca6a03cd33af924c,5b2d38217d75e2cdcb31cf05,5b2d384b7d75e2cdcb31cf06&type=array



{{url}}/api/product/articles\_by\_id?id=5b2d3648ca6a03cd33af924c&type=single



1. Product by orders (i.e. arrival (i.e. created\_at) or sell (i.e. sold))

Server/server.js

// BY ARRIVAL

// articles?sortBy=createdAt&order=desc&limit=4

// BY SELL

// articles?sortBy=sold&order=desc&limit=4

app.get("/api/product/articles", (req, res) => {

  let order = req.query.order ? req.query.order : "asc";

  let sortBy = req.query.sortBy ? req.query.sortBy : "\_id";

  let limit = req.query.limit ? parseInt(req.query.limit) : 100; // parseInt because no is converted to string by default in query string

  Product.find()

    .populate("brand")

    .populate("wood")

    .sort([[sortBy, order]])

    .limit(limit)

    .exec((err, articles) => {

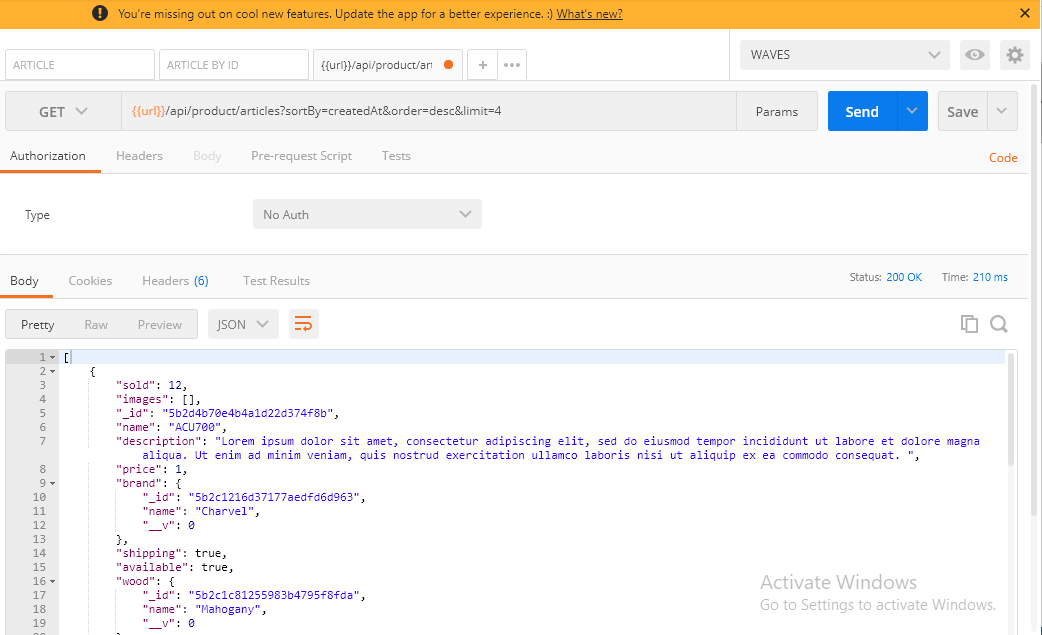
      if (err) return res.status(400).send(err);

      res.status(200).send(articles);

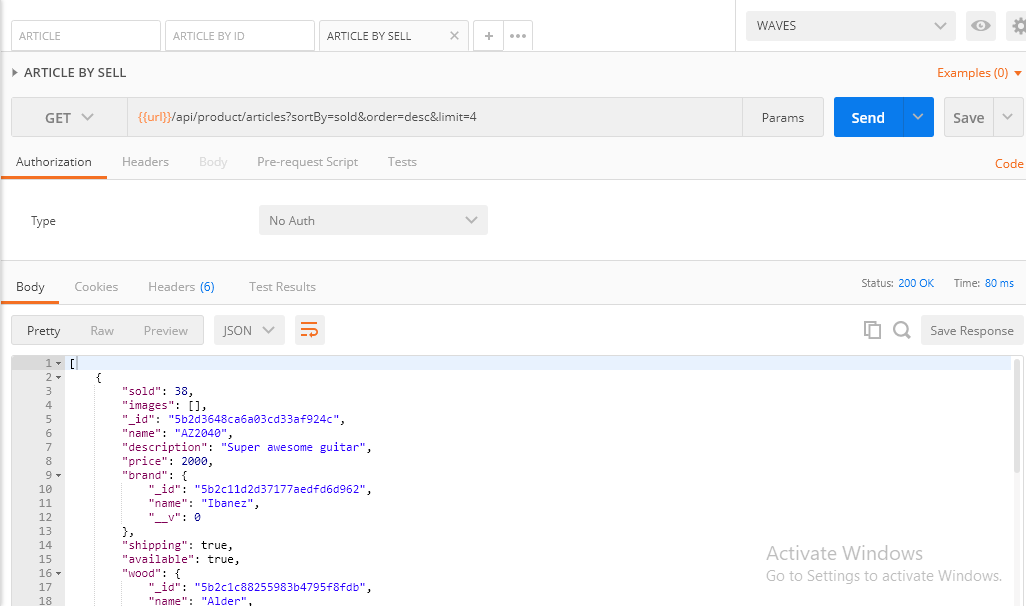
    });

});

// by arrival



// by sold



Client start

Go to waves/client

Create-react-app .

Dependencies packages

npm install @fortawesome/fontawesome@1.1.8 @fortawesome/fontawesome-free-solid@5.0.13 @fortawesome/react-fontawesome@0.0.20 @material-ui/core@1.2.2 axios@0.18.0 react-images@0.5.17 react-redux@5.0.7 react-router-dom@4.3.1 react-slick@0.23.1 redux@4.0.0 redux-promise@0.6.0 redux-thunk@2.3.0 react-dropzone@4.2.12 react-moment@0.7.7 react-paypal-express-checkout@1.0.4 –save

Add in client/public/index.html

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet" />

<link href="https://fonts.googleapis.com/css?family=Monoton|Oswald:300,400,500" rel="stylesheet">

<link rel="stylesheet" type="text/css" charset="UTF-8" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css" />

Client/src/app.js

Sending client to server request (port 3000 to 3002) will show error

import React, { Component } from "react";

import axios from "axios";

class App extends Component {

  componentDidMount() {

    axios.get("http://localhost:3002/api/product/brands").then(response => {

      console.log(response);

    });

  }

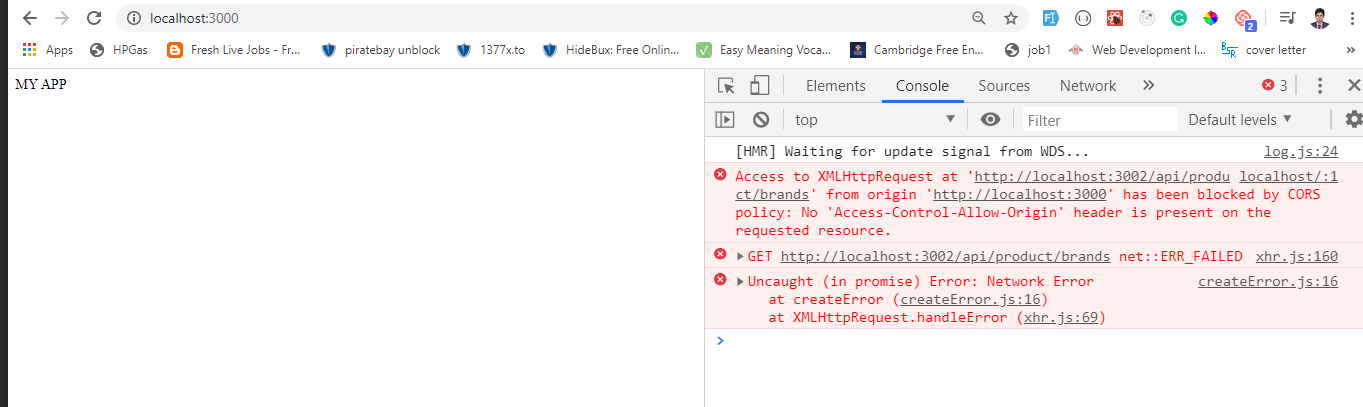
  render() {

    return <div className="App">MY APP</div>;

  }

}

export default App;



Quick fix

Go to react’s package.json (i.e. client/package.json)

i.e. // in the end

"proxy": “<http://localhost:3002>”

Client/src/App.js

import React, { Component } from "react";

import axios from "axios";

class App extends Component {

  componentDidMount() { // no need to write 3002 node server

    axios.get("/api/product/brands").then(response => {

      console.log(response);

    });

  }

  render() {

    return <div className="App">MY APP</div>;

  }

}

export default App;

Run node and react in same terminal

* For this concurrently package is installed for development

Open project package.json file

Waves/package.json

"scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "node server/server.js",

    "server": "nodemon server/server.js",

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

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

  },

Kill both react and node server

Open one terminal in waves (project root) – npm run dev

Header and Footer - Higher Order Component(HOC)

Rename app.js to router.js

Client/src/hoc/layout.js - hoc

import React, { Component } from "react";

class Layout extends Component {

  render() {

    return (

      <div>

        HEADER

        <div className="page\_container">{this.props.children}</div>

        FOOTER

      </div>

    );

  }

}

export default Layout;

client/src/router.js – component using hoc

import React from "react";

import { Switch, Route } from "react-router-dom";

import Home from "./components/Home";

import Layout from "./hoc/layout";

const Routes = () => {

  return (

    <Layout>

      <Switch>

        <Route path="/" component={Home} />

      </Switch>

    </Layout>

  );

};

export default Routes;

client/src/component/Home/index.js – home component

import React, { Component } from "react";

class Home extends Component {

  render() {

    return <div>home</div>;

  }

}

export default Home;

Redux setup

Inside src create two folders

Reducers and action

Src/index.js – Redux connectivity, chrome dev tools

import React from "react";

import ReactDOM from "react-dom";

import Routes from "./routes";

import "./Resources/css/styles.css";

import { BrowserRouter } from "react-router-dom";

import { Provider } from "react-redux";

import { createStore, applyMiddleware } from "redux";

import promiseMiddleware from "redux-promise";

import ReduxThunk from "redux-thunk";

import Reducer from "./reducers";

const createStoreWithMiddleware = applyMiddleware(

  promiseMiddleware,

  ReduxThunk

)(createStore);

const chromeDebugTool =

  window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_();

ReactDOM.render(

  <Provider store={createStoreWithMiddleware(Reducer, chromeDebugTool)}>

    <BrowserRouter>

      <Routes />

    </BrowserRouter>

  </Provider>,

  document.getElementById("root")

);

Src/reducers/index.js

import { combineReducers } from "redux";

import user from "./user\_reducer"; // list of reducers

const rootReducer = combineReducers({ // combining all reducers

  user

});

export default rootReducer;

Src/reducers/user\_reducer.js

export default function(state = {}, action) {

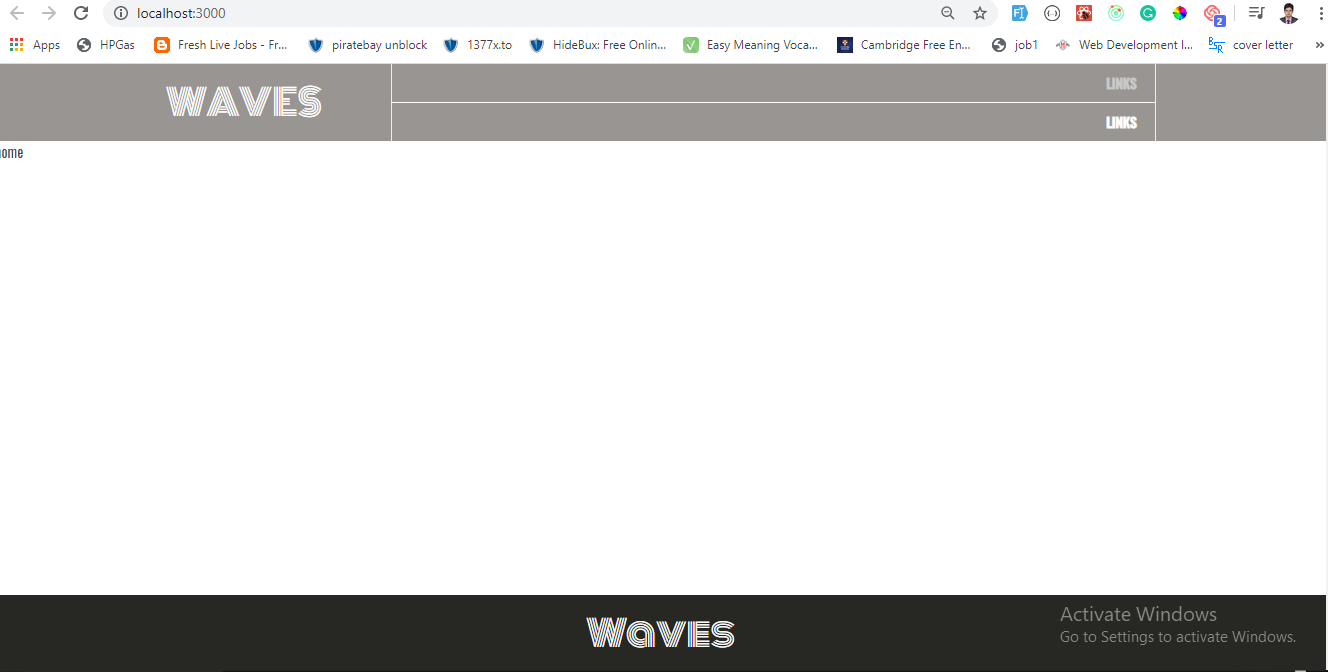
  switch (action.type) {

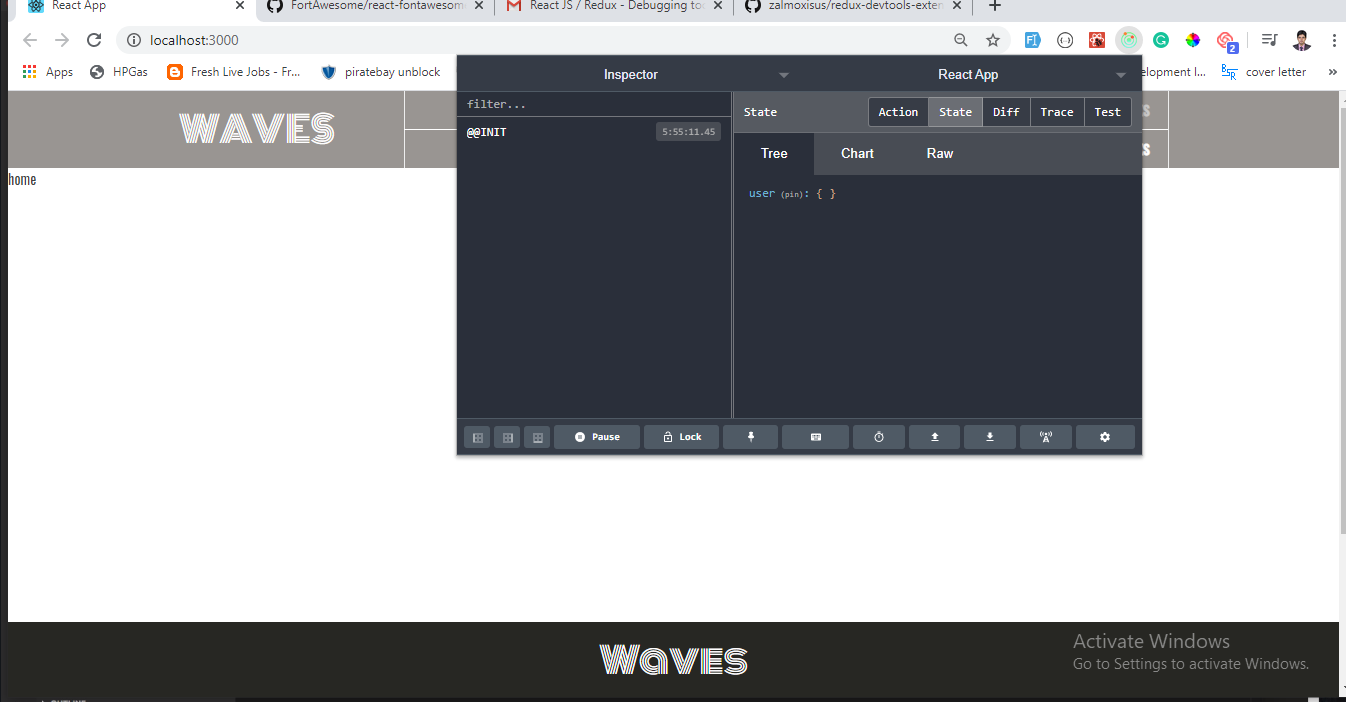
    default:

      return state;

  }

}





Create end point

src/components/utils/misc.js

//SERVER ROUTES

export const USER\_SERVER = "/api/users";

export const PRODUCT\_SERVER = "/api/product";

action creator- From here we will send requests to node server

src/actions/user\_actions.js

import axios from "axios";

import { USER\_SERVER } from "../components/utils/misc";

Create Register and login functionality

Src/routes

<Route path="/register\_login" component={RegisterLogin} /> - Add this line

Src/Components/Register\_login/index.js - for buttons created reusable component

import React from "react";

import MyButton from "../utils/button";

const RegisterLogin = () => {

  return (

    <div className="page\_wrapper">

      <div className="container">

        <div className="register\_login\_container">

          <div className="left">

            <h1>New Customers</h1>

            <p>

              Lorem Ipsum is simply dummy text of the printing and typesetting

              industry. Lorem Ipsum has been the industry's standard dummy text

              ever since the 1500s,

            </p>

            <MyButton

              type="default"

              title="Create and account"

              linkTo="/register"

              addStyles={{

                margin: "10px 0 0 0 "

              }}

            />

          </div>

          <div className="right"></div>

        </div>

      </div>

    </div>

  );

};

export default RegisterLogin;

utils/button.js

import React from "react";

import { Link } from "react-router-dom";

const MyButton = props => {

  const buttons = () => {

    let template = "";

    switch (props.type) {

      case "default":

        template = (

          <Link className="link\_default" to={props.linkTo} {...props.addStyles}>

            {props.title}

          </Link>

        );

        break;

      default:

        template = "";

    }

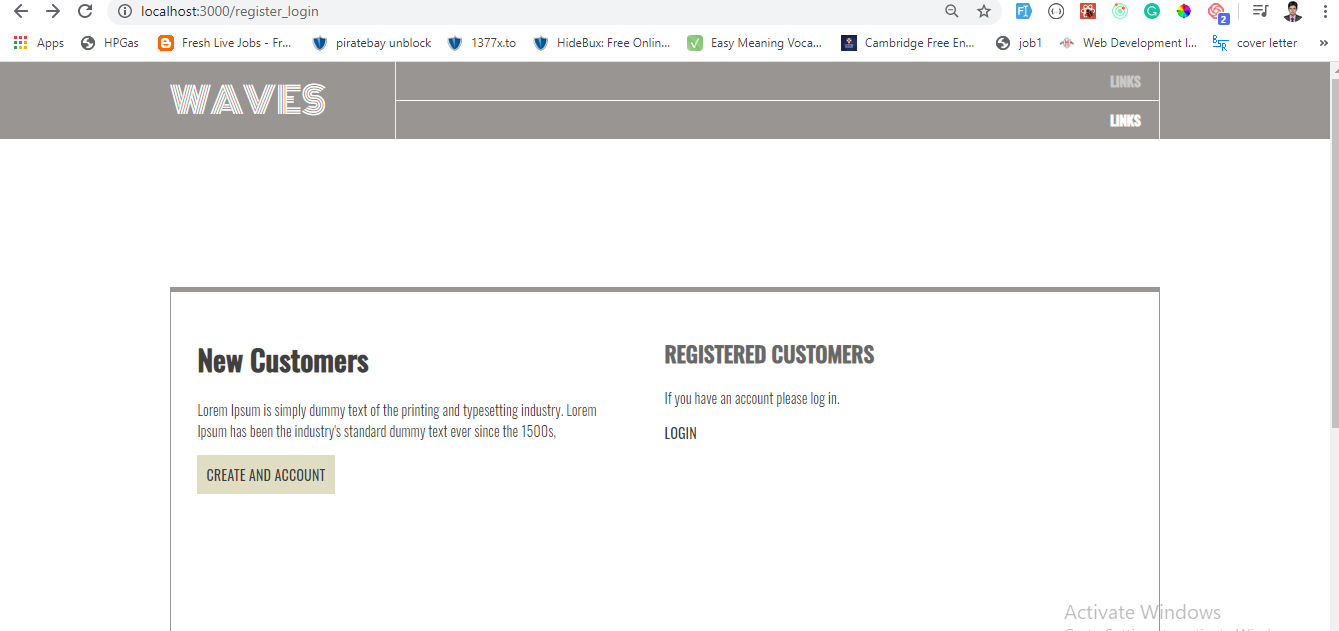
    return template;

  };

  return <div className="my\_link">{buttons()}</div>;

};

export default MyButton;



React Forms (login) – Component using reusable component

Component/Register\_login/login.js

import React, { Component } from "react";

import FormField from "../utils/Form/formfield";

import { update, generateData, isFormValid } from "../utils/Form/formActions";

class Login extends Component {

  state = {

    formError: false,

    formSuccess: "",

    formdata: {

      email: {

        element: "input",

        value: "",

        config: {

          name: "email\_input",

          type: "email",

          placeholder: "Enter your email"

        },

        validation: {

          required: true,

          email: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      },

      password: {

        element: "input",

        value: "",

        config: {

          name: "password\_input",

          type: "password",

          placeholder: "Enter your password"

        },

        validation: {

          required: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      }

    }

  };

  updateForm = element => {

    const newFormdata = update(element, this.state.formdata, "login");

    this.setState({

      formError: false,

      formdata: newFormdata

    });

  };

  submitForm = event => {

    event.preventDefault();

    let dataToSubmit = generateData(this.state.formdata, "login"); // get key-value pair to data to save

    let formIsValid = isFormValid(this.state.formdata, "login"); // check validations before submit

    if (formIsValid) {

      console.log(dataToSubmit);

    } else {

      this.setState({

        formError: true

      });

    }

  };

  render() {

    return (

      <div className="signin\_wrapper">

        <form onSubmit={() => this.submitForm()}>

          <FormField

            id={"email"}

            formdata={this.state.formdata.email}

            change={element => this.updateForm(element)}

          />

          <FormField

            id={"password"}

            formdata={this.state.formdata.password}

            change={element => this.updateForm(element)}

          />

          {this.state.formError ? (

            <div className="error\_label">Please check your data</div>

          ) : null}

          <button onClick={event => this.submitForm(event)}>Log in</button>

        </form>

      </div>

    );

  }

}

export default Login;

components/utils/login/formfields.js (Reusable component) – Written HTML here

/\* Actual input field HTML code is written here  \*/

import React from "react";

const FormField = ({ formdata, change, id }) => {

  // Displaying error message if any

  const showError = () => {

    let errorMessage = null;

    if (formdata.validation && !formdata.valid) {

      errorMessage = (

        <div className="error\_label">{formdata.validationMessage}</div>

      );

    }

    return errorMessage;

  };

  const renderTemplate = () => {

    let formTemplate = null;

    switch (formdata.element) {

      case "input":

        formTemplate = (

          <div className="formBlock">

            <input

              {...formdata.config}

              value={formdata.value}

              onBlur={event => change({ event, id, blur: true })}

              onChange={event => change({ event, id })}

            />

            {showError()}

          </div>

        );

        break;

      default:

        formTemplate = null;

    }

    return formTemplate;

  };

  return <div>{renderTemplate()}</div>;

};

export default FormField;

components/utils/form/formActions.js – On input update – Reusable Component

// check form validation for displaying messages

export const validate = (element, formdata = []) => {

  let error = [true, ""];

  if (element.validation.email) {

    const valid = /\S+@\S+\.\S+/.test(element.value);

    const message = `${!valid ? "Must be a valid email" : ""}`;

    error = !valid ? [valid, message] : error;

  }

  if (element.validation.required) {

    const valid = element.value.trim() !== "";

    const message = `${!valid ? "This field is required" : ""}`;

    error = !valid ? [valid, message] : error;

  }

  return error;

};

// On input typing

export const update = (element, formdata, formName) => {

  // original formdata

  const newFormdata = {

    ...formdata

  };

  // making the copy of original formdata and mutating there i.e. updating values

  const newElement = {

    ...newFormdata[element.id]

  };

  newElement.value = element.event.target.value;

  if (element.blur) {

    let validData = validate(newElement, formdata);

    newElement.valid = validData[0];

    newElement.validationMessage = validData[1];

  }

  newElement.touched = element.blur;

  // Updating the original formdata

  newFormdata[element.id] = newElement;

  return newFormdata;

};

// Getting key-value of input type : eg. email: amir@gmail.com, password: Test@123 - save in mongo DB

export const generateData = (formdata, formName) => {

  let dataToSubmit = {};

  for (let key in formdata) {

    dataToSubmit[key] = formdata[key].value;

  }

  return dataToSubmit;

};

// check form validation on submit

export const isFormValid = (formdata, formName) => {

  let formIsValid = true;

  for (let key in formdata) {

    formIsValid = formdata[key].valid && formIsValid;

  }

  return formIsValid;

};

Redux for sending request using Axios and getting data

Now for submitting data we will use redux action creator. Here we will dispatch the data to redux action and get the response back from response.

**Why use types?**

We use types.js in action – we pass variables instead of string because even if we make mistakes, data is not stored in central state but we are not able to see error message or warning. So we pass variables

Login.js

**Src/components/Register\_login**

import React, { Component } from "react";

import FormField from "../utils/Form/formfield";

import { update, generateData, isFormValid } from "../utils/Form/formActions";

import { withRouter } from "react-router-dom"; // can be used in all child components props

import { connect } from "react-redux";

import { loginUser } from "../../actions/user\_actions";

class Login extends Component {

  state = {

    formError: false,

    formSuccess: "",

    formdata: {

      email: {

        element: "input",

        value: "",

        config: {

          name: "email\_input",

          type: "email",

          placeholder: "Enter your email"

        },

        validation: {

          required: true,

          email: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      },

      password: {

        element: "input",

        value: "",

        config: {

          name: "password\_input",

          type: "password",

          placeholder: "Enter your password"

        },

        validation: {

          required: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      }

    }

  };

  updateForm = element => {

    const newFormdata = update(element, this.state.formdata, "login");

    this.setState({

      formError: false,

      formdata: newFormdata

    });

  };

  submitForm = event => {

    event.preventDefault();

    let dataToSubmit = generateData(this.state.formdata, "login"); // get key-value pair to data to save

    let formIsValid = isFormValid(this.state.formdata, "login"); // check validations before submit

    if (formIsValid) {

      this.props.dispatch(loginUser(dataToSubmit)).then(response => {

        if (response.payload.loginSuccess) {

          console.log(response.payload);

          this.props.history.push("/user/dashboard");

        } else {

          this.setState({

            formError: true

          });

        }

        console.log(response.payload);

      });

      // console.log(dataToSubmit);     {email: "amirengg15@gmail.com", password: "Admin@123"}

    } else {

      this.setState({

        formError: true

      });

    }

  };

  render() {

    return (

      <div className="signin\_wrapper">

        <form onSubmit={() => this.submitForm()}>

          <FormField

            id={"email"}

            formdata={this.state.formdata.email}

            change={element => this.updateForm(element)}

          />

          <FormField

            id={"password"}

            formdata={this.state.formdata.password}

            change={element => this.updateForm(element)}

          />

          {this.state.formError ? (

            <div className="error\_label">Please check your data</div>

          ) : null}

          <button onClick={event => this.submitForm(event)}>Log in</button>

        </form>

      </div>

    );

  }

}

export default connect()(withRouter(Login));

useractions.js

path – src/actions/user\_actions.js

import axios from "axios";

import { USER\_SERVER } from "../components/utils/misc";

import { LOGIN\_USER } from "./types";

export function loginUser(dataToSubmit) { // sending login request

  const request = axios

    .post(`${USER\_SERVER}/login`, dataToSubmit)

    .then(response => response.data);

  return {

    type: LOGIN\_USER,

    payload: request

  };

}

Userreducers.js

Path – arc/actions/user\_reducers.js

import { LOGIN\_USER } from "../actions/types";

export default function(state = {}, action) {

  switch (action.type) {

    case LOGIN\_USER:

      return { ...state, loginSuccess: action.payload };

    default:

      return state;

  }

}

Register functionality

Register.js – file using reusable component

import React, { Component } from "react";

import FormField from "../utils/Form/formfield";

import { update, generateData, isFormValid } from "../utils/Form/formActions"; // Reusable component

import Dialog from "@material-ui/core/Dialog"; // for successful register msg

import { connect } from "react-redux"; // Redux for sending register request to node server

import { registerUser } from "../../actions/user\_actions";

class Register extends Component {

  state = {

    formError: false,

    formSuccess: false,

    formdata: {

      name: {

        element: "input",

        value: "",

        config: {

          name: "name\_input",

          type: "text",

          placeholder: "Enter your name"

        },

        validation: {

          required: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      },

      lastname: {

        element: "input",

        value: "",

        config: {

          name: "lastname\_input",

          type: "text",

          placeholder: "Enter your lastname"

        },

        validation: {

          required: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      },

      email: {

        element: "input",

        value: "",

        config: {

          name: "email\_input",

          type: "email",

          placeholder: "Enter your email"

        },

        validation: {

          required: true,

          email: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      },

      password: {

        element: "input",

        value: "",

        config: {

          name: "password\_input",

          type: "password",

          placeholder: "Enter your password"

        },

        validation: {

          required: true

        },

        valid: false,

        touched: false,

        validationMessage: ""

      },

      confirmPassword: {

        element: "input",

        value: "",

        config: {

          name: "confirm\_password\_input",

          type: "password",

          placeholder: "Confirm your password"

        },

        validation: {

          required: true,

          confirm: "password"

        },

        valid: false,

        touched: false,

        validationMessage: ""

      }

    }

  };

  updateForm = element => {

    const newFormdata = update(element, this.state.formdata, "register");

    this.setState({

      formError: false,

      formdata: newFormdata

    });

  };

  submitForm = event => {

    event.preventDefault();

    let dataToSubmit = generateData(this.state.formdata, "register");

    let formIsValid = isFormValid(this.state.formdata, "register");

    if (formIsValid) {

      this.props // from component to action creator - redux

        .dispatch(registerUser(dataToSubmit))

        .then(response => {

          if (response.payload.success) {

            this.setState({

              formError: false,

              formSuccess: true

            });

            setTimeout(() => {

              this.props.history.push("/register\_login");

            }, 3000);

          } else {

            this.setState({ formError: true });

          }

        })

        .catch(e => {

          this.setState({ formError: true });

        });

    } else {

      this.setState({

        formError: true

      });

    }

  };

  render() {

    return (

      <div className="page\_wrapper">

        <div className="container">

          <div className="register\_login\_container">

            <div className="left">

              <form onSubmit={event => this.submitForm(event)}>

                <h2>Personal information</h2>

                <div className="form\_block\_two">

                  <div className="block">

                    <FormField

                      id={"name"}

                      formdata={this.state.formdata.name}

                      change={element => this.updateForm(element)}

                    />

                  </div>

                  <div className="block">

                    <FormField

                      id={"lastname"}

                      formdata={this.state.formdata.lastname}

                      change={element => this.updateForm(element)}

                    />

                  </div>

                </div>

                <div>

                  <FormField

                    id={"email"}

                    formdata={this.state.formdata.email}

                    change={element => this.updateForm(element)}

                  />

                </div>

                <h2>Verify password</h2>

                <div className="form\_block\_two">

                  <div className="block">

                    <FormField

                      id={"password"}

                      formdata={this.state.formdata.password}

                      change={element => this.updateForm(element)}

                    />

                  </div>

                  <div className="block">

                    <FormField

                      id={"confirmPassword"}

                      formdata={this.state.formdata.confirmPassword}

                      change={element => this.updateForm(element)}

                    />

                  </div>

                </div>

                <div>

                  {this.state.formError ? (

                    <div className="error\_label">Please check your data</div>

                  ) : null}

                  <button onClick={event => this.submitForm(event)}>

                    Create an account

                  </button>

                </div>

              </form>

            </div>

          </div>

        </div>

        <Dialog open={this.state.formSuccess}>

          <div className="dialog\_alert">

            <div>Congratulations !!</div>

            <div>You will be redirected to LOGIN in a couple seconds...</div>

          </div>

        </Dialog>

      </div>

    );

  }

}

export default connect()(Register);

formAction.js – Reusable component – on update

// check form validation for displaying messages

export const validate = (element, formdata = []) => {

  let error = [true, ""];

  // email validation rule

  if (element.validation.email) {

    const valid = /\S+@\S+\.\S+/.test(element.value);

    const message = `${!valid ? "Must be a valid email" : ""}`;

    error = !valid ? [valid, message] : error;

  }

  // confirm password rule

  if (element.validation.confirm) {

    const valid =

      element.value.trim() == formdata[element.validation.confirm].value;

    const message = `${!valid ? "Passwords do not match." : ""}`;

    error = !valid ? [valid, message] : error;

  }

  // required field rule

  if (element.validation.required) {

    const valid = element.value.trim() !== "";

    const message = `${!valid ? "This field is required" : ""}`;

    error = !valid ? [valid, message] : error;

  }

  return error;

};

// On input typing

export const update = (element, formdata, formName) => {

  // original formdata

  const newFormdata = {

    ...formdata

  };

  // making the copy of original formdata and mutating there i.e. updating values

  const newElement = {

    ...newFormdata[element.id]

  };

  newElement.value = element.event.target.value;

  if (element.blur) {

    let validData = validate(newElement, formdata);

    newElement.valid = validData[0];

    newElement.validationMessage = validData[1];

  }

  newElement.touched = element.blur;

  // Updating the original formdata

  newFormdata[element.id] = newElement;

  return newFormdata;

};

// Getting key-value of input type : eg. email: amir@gmail.com, password: Test@123 - save in mongo DB

export const generateData = (formdata, formName) => {

  let dataToSubmit = {};

  for (let key in formdata) {

    // we do not want to write confirm password in mongo db

    if (key !== "confirmPassword") {

      dataToSubmit[key] = formdata[key].value;

    }

  }

  return dataToSubmit;

};

// check form validation on submit

export const isFormValid = (formdata, formName) => {

  let formIsValid = true;

  for (let key in formdata) {

    formIsValid = formdata[key].valid && formIsValid;

  }

  return formIsValid;

};

formfield.js – Reusable Component

/\* Actual input field HTML code is written here  \*/

import React from "react";

const FormField = ({ formdata, change, id }) => {

  // Displaying error message if any

  const showError = () => {

    let errorMessage = null;

    if (formdata.validation && !formdata.valid) {

      errorMessage = (

        <div className="error\_label">{formdata.validationMessage}</div>

      );

    }

    return errorMessage;

  };

  const renderTemplate = () => {

    let formTemplate = null;

    switch (formdata.element) {

      case "input":

        formTemplate = (

          <div className="formBlock">

            <input

              {...formdata.config}

              value={formdata.value}

              onBlur={event => change({ event, id, blur: true })}

              onChange={event => change({ event, id })}

            />

            {showError()}

          </div>

        );

        break;

      default:

        formTemplate = null;

    }

    return formTemplate;

  };

  return <div>{renderTemplate()}</div>;

};

export default FormField;

// Redux for sending request

userAction.js

path – src/actions/userActions.js

import axios from "axios";

import { USER\_SERVER } from "../components/utils/misc";

import { LOGIN\_USER, REGISTER\_USER } from "./types";

export function registerUser(dataToSubmit) {

  const request = axios

    .post(`${USER\_SERVER}/register`, dataToSubmit)

    .then(response => response.data);

  return {

    type: REGISTER\_USER,

    payload: request

  };

}

export function loginUser(dataToSubmit) {

  const request = axios

    .post(`${USER\_SERVER}/login`, dataToSubmit)

    .then(response => response.data);

  return {

    type: LOGIN\_USER,

    payload: request

  };

}

userReducers.js

path – src/reducers/userReducers.js

import { LOGIN\_USER, REGISTER\_USER } from "../actions/types";

export default function(state = {}, action) {

  switch (action.type) {

    case REGISTER\_USER:

      return { ...state, register: action.payload };

    case LOGIN\_USER:

      return { ...state, loginSuccess: action.payload };

    default:

      return state;

  }

}

Preparing Admin dashboard

HOC – for left admin sidebar

Path – src/hoc/user.js

import React from "react";

import { Link } from "react-router-dom";

const links = [

  {

    name: "My account",

    linkTo: "/user/dashboard"

  },

  {

    name: "User information",

    linkTo: "/user/user\_profile"

  },

  {

    name: "My Cart",

    linkTo: "/user/cart"

  }

];

const UserLayout = props => {

  const generateLinks = links =>

    links.map((item, i) => (

      <Link to={item.linkTo} key={i}>

        {item.name}

      </Link>

    ));

  return (

    <div className="container">

      <div className="user\_container">

        <div className="user\_left\_nav">

          <h2>My account</h2>

          <div className="links">{generateLinks(links)}</div>

        </div>

        <div className="user\_right">{props.children}</div>

      </div>

    </div>

  );

};

export default UserLayout;

routes.js

path – waves/client/src/routes.js

import React from "react";

import { Switch, Route } from "react-router-dom";

import Home from "./components/Home";

import Layout from "./hoc/layout";

import RegisterLogin from "./components/Register\_login";

import Register from "./components/Register\_login/register";

import UserDashboard from "./components/User";

const Routes = () => {

  return (

    <Layout>

      <Switch>

        <Route path="/user/dashboard" component={UserDashboard} />

        <Route path="/register" component={Register} />

        <Route path="/register\_login" component={RegisterLogin} />

        <Route path="/" component={Home} />

      </Switch>

    </Layout>

  );

};

export default Routes;

Dashboard Component

Path – src/components/User/index.js

import React from "react";

import UserLayout from "../../hoc/user";

const UserDashboard = () => {

  return (

    <UserLayout>

      <div>dashboard</div>

    </UserLayout>

  );

};

export default UserDashboard;

// Private, semi public and public routes

// Auth – Where ever we need server url we will use redux (axios in server - action creator)

Composed component - a function that returns another component

Path - Client/src/routes.js

<Route path="/user/dashboard" component={Auth(UserDashboard, true)} />

And in component instead of component directly we write inside export function

/\* Here we hit server's auth route i.e. /api/users/auth to check authentication \*/

import React, { Component } from "react";

import { connect } from "react-redux";

import { auth } from "../actions/user\_actions";

import CircularProgress from "@material-ui/core/CircularProgress";

// composed component - A function that returns another function

export default function(ComposedClass, reload, adminRoute = null) {

  class AuthenticationCheck extends Component {

    render() {

      return <div>Auth</div>;

    }

  }

  return connect()(AuthenticationCheck); // instead of export default of component

}

}

Same above with specific

/\* Here we hit server's auth route i.e. /api/users/auth to check authentication \*/

import React, { Component } from "react";

import { connect } from "react-redux";

import { auth } from "../actions/user\_actions";

import CircularProgress from "@material-ui/core/CircularProgress";

// composed component - A function that returns another function

export default function(ComposedClass, reload, adminRoute = null) {

  class AuthenticationCheck extends Component {

    state = {

      loading: true

    };

    render() {

      if (this.state.loading) {

        return (

          <div className="main\_loader">

            <CircularProgress style={{ color: "#2196F3" }} thickness={7} />

          </div>

        );

      }

      // Parameter 1 is component

      return <ComposedClass />;

    }

  }

  function mapStateToProps(state) {

    return {

      user: state.props

    };

  }

  return connect(mapStateToProps)(AuthenticationCheck);

}

For making private, public and semi public route we used composed component –

check client/hoc/auth.js, userreducer.js, useraction.js, routes.js

React JS Slider

import React from "react";

import Slider from "react-slick";

import MyButton from "../utils/button";

const HomeSlider = props => {

  const slides = [

    {

      img: "/images/featured/featured\_home.jpg",

      lineOne: "Fender",

      lineTwo: "Custom shop",

      linkTitle: "Shop now",

      linkTo: "/shop"

    },

    {

      img: "/images/featured/featured\_home\_2.jpg",

      lineOne: "B-Stock",

      lineTwo: "Awesome discounts",

      linkTitle: "View offers",

      linkTo: "/shop"

    }

  ];

  const settings = {

    dots: false,

    infinite: true,

    speed: 500,

    slidesToShow: 1,

    slidesToScroll: 1,

    arrows: false

  };

  const generateSlides = () =>

    slides

      ? slides.map((item, i) => (

          <div key={i}>

            <div

              className="featured\_image"

              style={{

                background: `url(${item.img})`,

                height: `${window.innerHeight}px`

              }}

            >

              <div className="featured\_action">

                <div className="tag title">{item.lineOne}</div>

                <div className="tag low\_title">{item.lineTwo}</div>

                <div>

                  <MyButton

                    type="default"

                    title={item.linkTitle}

                    linkTo={item.linkTo}

                    addStyles={{

                      margin: "10px 0 0 0"

                    }}

                  />

                </div>

              </div>

            </div>

          </div>

        ))

      : null;

  return (

    <div className="featured\_container">

      <Slider {...settings}>{generateSlides()}</Slider>

    </div>

  );

};

export default HomeSlider;

Home layout

Home/index.js

import React, { Component } from "react";

import HomeSlider from "./home\_slider";

import HomePromotion from "./home\_promotion";

import CardBlock from "../utils/card\_block";

import { connect } from "react-redux";

import {

  getProductsBySell,

  getProductsByArrival

} from "../../actions/products\_actions";

class Home extends Component {

  componentDidMount() {

    // hitting same node server request with two different arguments

    this.props.dispatch(getProductsBySell());

    this.props.dispatch(getProductsByArrival());

  }

  render() {

    return (

      <div>

        <HomeSlider />

        <CardBlock

          list={this.props.products.bySell}

          title="Best Selling guitars"

        />

        <HomePromotion />

        <CardBlock list={this.props.products.byArrival} title="New arrivals" />

      </div>

    );

  }

}

const mapStateToProps = state => {

  return {

    products: state.products

  };

};

export default connect(mapStateToProps)(Home);

actions/products\_actions.js (action creator)

import axios from "axios";

import { GET\_PRODUCTS\_BY\_SELL, GET\_PRODUCTS\_BY\_ARRIVAL } from "./types";

import { PRODUCT\_SERVER } from "../components/utils/misc";

export function getProductsBySell() {

  //?sortBy=sold&order=desc&limit=100

  const request = axios

    .get(`${PRODUCT\_SERVER}/articles?sortBy=sold&order=desc&limit=4`)

    .then(response => response.data);

  return {

    type: GET\_PRODUCTS\_BY\_SELL,

    payload: request

  };

}

export function getProductsByArrival() {

  //?sortBy=createdAt&order=desc&limit=100

  const request = axios

    .get(`${PRODUCT\_SERVER}/articles?sortBy=createdAt&order=desc&limit=4`)

    .then(response => response.data);

  return {

    type: GET\_PRODUCTS\_BY\_ARRIVAL,

    payload: request

  };

}

Reducers/products\_reducer.js

import {

  GET\_PRODUCTS\_BY\_SELL,

  GET\_PRODUCTS\_BY\_ARRIVAL

} from "../actions/types";

export default function(state = {}, action) {

  switch (action.type) {

    case GET\_PRODUCTS\_BY\_SELL:

      return { ...state, bySell: action.payload };

    case GET\_PRODUCTS\_BY\_ARRIVAL:

      return { ...state, byArrival: action.payload };

    default:

      return state;

  }

}

SHOP

Components/shop/index.js

import React, { Component } from "react";

import PageTop from "../utils/page\_top";

import { connect } from "react-redux";

class Shop extends Component {

  render() {

    return (

        <div>

            <PageTop

                title="Browse Products"

            />

        </div>

    );

  }

}

const mapStateToProps = state => {

  return {

    products: state.products

  };

};

export default connect(mapStateToProps)(Shop);

routes.js

import React from "react";

import { Switch, Route } from "react-router-dom";

import Layout from "./hoc/layout";

import Auth from "./hoc/auth";

import Home from "./components/Home";

import RegisterLogin from "./components/Register\_login";

import Register from "./components/Register\_login/register";

import Shop from "./components/Shop";

import UserDashboard from "./components/User";

const Routes = () => {

  return (

    <Layout>

      <Switch>

        {/\* private routes \*/}

        <Route path="/user/dashboard" component={Auth(UserDashboard, true)} />

        {/\* semi public routes - should not go to this route after login \*/}

        <Route path="/register" component={Auth(Register, false)} />

        <Route path="/register\_login" component={Auth(RegisterLogin, false)} />

        {/\* public routes \*/}

        <Route path="/shop" component={Auth(Shop, null)} />

        <Route path="/" component={Auth(Home, null)} />

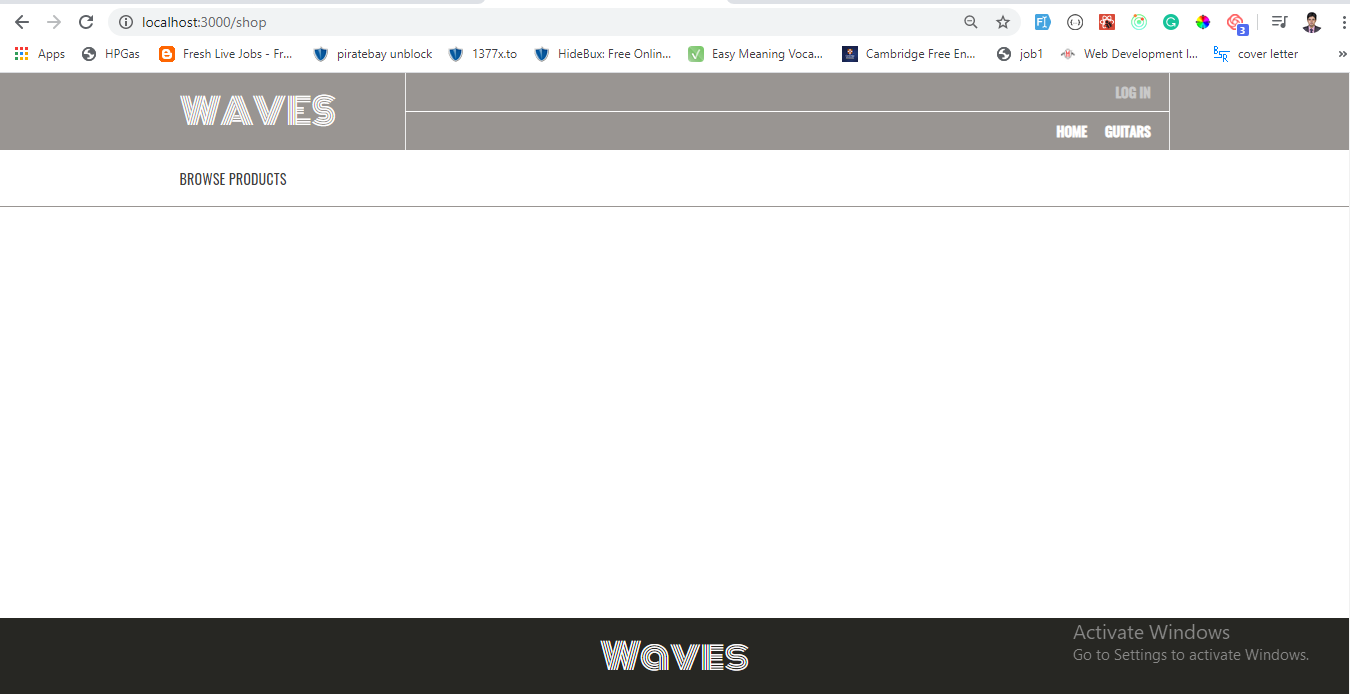
      </Switch>

    </Layout>

  );

};

export default Routes;



Using redux for getting shop data – get woods and get brands

Components/shop/index.js

import React, { Component } from "react";

import PageTop from "../utils/page\_top";

import { connect } from "react-redux";

import { getBrands, getWoods } from "../../actions/products\_actions";

class Shop extends Component {

  componentDidMount() {

    // getting woods data

    this.props.dispatch(getBrands());

    this.props.dispatch(getWoods());

  }

  render() {

    return (

      <div>

        <PageTop title="Browse Products" />

        <div className="container">

          <div className="shop\_wrapper">

            <div className="left">left</div>

            <div className="right">right</div>

          </div>

        </div>

      </div>

    );

  }

}

const mapStateToProps = state => {

  return {

    products: state.products

  };

};

export default connect(mapStateToProps)(Shop);

components/actions/productactions.js

import axios from "axios";

import {

  GET\_PRODUCTS\_BY\_SELL,

  GET\_PRODUCTS\_BY\_ARRIVAL,

  GET\_BRANDS,

  GET\_WOODS

} from "./types";

import { PRODUCT\_SERVER } from "../components/utils/misc";

export function getProductsBySell() {

  //?sortBy=sold&order=desc&limit=100

  const request = axios

    .get(`${PRODUCT\_SERVER}/articles?sortBy=sold&order=desc&limit=4`)

    .then(response => response.data);

  return {

    type: GET\_PRODUCTS\_BY\_SELL,

    payload: request

  };

}

export function getProductsByArrival() {

  //?sortBy=createdAt&order=desc&limit=100

  const request = axios

    .get(`${PRODUCT\_SERVER}/articles?sortBy=createdAt&order=desc&limit=4`)

    .then(response => response.data);

  return {

    type: GET\_PRODUCTS\_BY\_ARRIVAL,

    payload: request

  };

}

////////////////////////////////////////////////

////////        CATEGORIES

////////////////////////////////////////////////

export function getBrands() {

  const request = axios

    .get(`${PRODUCT\_SERVER}/brands`)

    .then(response => response.data);

  return {

    type: GET\_BRANDS,

    payload: request

  };

}

export function getWoods() {

  const request = axios

    .get(`${PRODUCT\_SERVER}/woods`)

    .then(response => response.data);

  return {

    type: GET\_WOODS,

    payload: request

  };

}

Reducers/products\_reducers.js

import {

  GET\_PRODUCTS\_BY\_SELL,

  GET\_PRODUCTS\_BY\_ARRIVAL,

  GET\_BRANDS,

  GET\_WOODS

} from "../actions/types";

export default function(state = {}, action) {

  switch (action.type) {

    case GET\_PRODUCTS\_BY\_SELL:

      return { ...state, bySell: action.payload };

    case GET\_PRODUCTS\_BY\_ARRIVAL:

      return { ...state, byArrival: action.payload };

    case GET\_BRANDS:

      return { ...state, brands: action.payload };

    case GET\_WOODS:

      return { ...state, woods: action.payload };

    default:

      return state;

  }

}

Actions/types.js

export const LOGIN\_USER = "login\_user";

export const REGISTER\_USER = "register\_user";

export const AUTH\_USER = "auth\_user";

export const LOGOUT\_USER = "logout\_user";

export const GET\_PRODUCTS\_BY\_SELL = "get\_products\_by\_sell";

export const GET\_PRODUCTS\_BY\_ARRIVAL = "get\_products\_by\_arrival";

export const GET\_BRANDS = "get\_brands";

export const GET\_WOODS = "get\_woods";

COLLAPSE CHECKBOX – Material UI

Shop/index.js

import React, { Component } from "react";

import PageTop from "../utils/page\_top";

import { connect } from "react-redux";

import { getBrands, getWoods } from "../../actions/products\_actions";

import CollapseCheckbox from "../utils/collapseCheckbox.js";

class Shop extends Component {

  componentDidMount() {

    // getting woods data

    this.props.dispatch(getWoods());

    this.props.dispatch(getBrands());

  }

  handleFilters = () => {};

  render() {

    const products = this.props.products;

    return (

      <div>

        <PageTop title="Browse Products" />

        <div className="container">

          <div className="shop\_wrapper">

            <div className="left">

              <CollapseCheckbox

                initState={true}

                title="Brands"

                list={products.brands}

                handleFilters={filters => this.handleFilters(filters, "brand")}

              />

            </div>

            <div className="right">right</div>

          </div>

        </div>

      </div>

    );

  }

}

const mapStateToProps = state => {

  return {

    products: state.products

  };

};

export default connect(mapStateToProps)(Shop);

collapseCheckbox.js – reusable component

path – components/utils/collapseCheckbox

import React, { Component } from "react";

import FontAwesomeIcon from "@fortawesome/react-fontawesome";

import faAngleDown from "@fortawesome/fontawesome-free-solid/faAngleDown";

import faAngleUp from "@fortawesome/fontawesome-free-solid/faAngleUp";

import List from "@material-ui/core/List";

import ListItem from "@material-ui/core/ListItem";

import ListItemSecondaryAction from "@material-ui/core/ListItemSecondaryAction";

import ListItemText from "@material-ui/core/ListItemText";

import Checkbox from "@material-ui/core/Checkbox";

import Collapse from "@material-ui/core/Collapse";

class CollapseCheckbox extends Component {

  state = {

    open: false,

    checked: []

  };

  componentDidMount() {

    if (this.props.initState) {

      this.setState({

        open: this.props.initState

      });

    }

  }

  handleClick = () => {

    this.setState({

      open: !this.state.open

    });

  };

  handleAngle = () =>

    this.state.open ? (

      <FontAwesomeIcon icon={faAngleUp} className="icon" />

    ) : (

      <FontAwesomeIcon icon={faAngleDown} className="icon" />

    );

  render() {

    return (

      <div className="collapse\_items\_wrapper">

        <List style={{ borderBottom: "1px solid #dbdbdb" }}>

          <ListItem

            onClick={this.handleClick}

            style={{ padding: "10px 23px 10px 0" }}

          >

            <ListItemText

              primary={this.props.title}

              className="collapse\_title"

            />

            {this.handleAngle()}

          </ListItem>

          <Collapse in={this.state.open} timeout="auto" ummountOnExit>

            <ListItemText

              primary={this.props.title}

              className="collapse\_title"

            />

          </Collapse>

        </List>

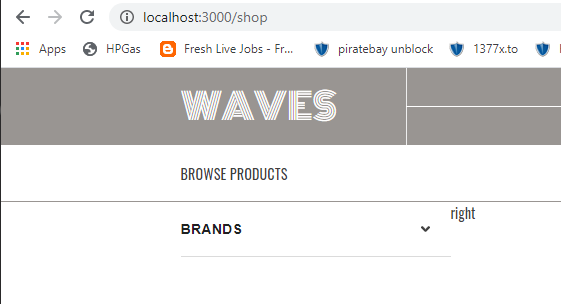
      </div>

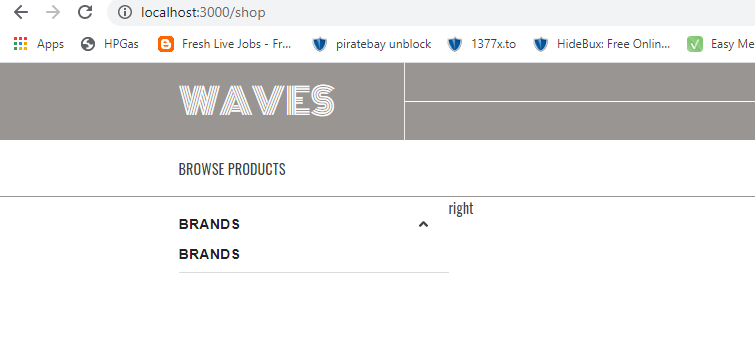
    );

  }

}

export default CollapseCheckbox;





Material UI Checkboxes

<https://material-ui.com/components/lists/>

see lists with image and checkbox

collapseCheckbox.js – reusable component

path – component/utils/collapseCheckbox.js

**renderList** = () =>

    this.props.list

      ? this.props.list.**map**(value => (

          <**ListItem** *key*={value.\_id} *style*={{ padding: "10px 0" }}>

            <**ListItemText** *primary*={value.name} />

            <**ListItemSecondaryAction**> <!-- Material UI 🡪

              <**Checkbox**

*color*="primary"

*onChange*={this.**handleToggle**(value.\_id)}

*checked*={this.state.checked.**indexOf**(value.\_id) !== -1}

              />

            </**ListItemSecondaryAction**>

          </**ListItem**>

        ))

      : null;

*// Here we are getting this.state.checked data - push/pop data from array (i.e. \_id value), copied the original state array and finally updating state*

**handleToggle** = value => () => {

    const { checked } = this.state;

    const currentIndex = checked.**indexOf**(value);

    const newChecked = [...checked];

    if (currentIndex === -1) {

      newChecked.**push**(value);

    } else {

      newChecked.**splice**(currentIndex, 1);

    }

    this.**setState**({

      checked: newChecked

    });

  };

**Way to pass function as props and receive it back to parent**

**Shop/index.js – parent component**

**handleFilters** = (filters, categories) => {

    console.**log**(filters);

  };

**render**() {

    const products = this.props.products;

    return (

      <div>

        <**PageTop** *title*="Browse Products" />

        <div *className*="container">

          <div *className*="shop\_wrapper">

            <div *className*="left">

              <**CollapseCheckbox**

*initState*={true}

*title*="Brands"

*list*={products.brands}

*handleFilters*={filters => this.**handleFilters**(filters, "brand")}

              />

            </div>

            <div *className*="right">right</div>

          </div>

        </div>

      </div>

    );

  }

collapseCheckbox.js

// render list is called in render()

**renderList** = () =>

    this.props.list

      ? this.props.list.**map**(value => (

          <**ListItem** *key*={value.\_id} *style*={{ padding: "10px 0" }}>

            <**ListItemText** *primary*={value.name} />

            <**ListItemSecondaryAction**>

              <**Checkbox**

*color*="primary"

*onChange*={this.**handleToggle**(value.\_id)}

*checked*={this.state.checked.**indexOf**(value.\_id) !== -1}

              />

            </**ListItemSecondaryAction**>

          </**ListItem**>

        ))

      : null;

*// Here we are getting this.state.checked data - push/pop data from array (i.e. \_id value), copied the original state array and finally updating state*

**handleToggle** = value => () => {

    const { checked } = this.state;

    const currentIndex = checked.**indexOf**(value);

    const newChecked = [...checked];

    if (currentIndex === -1) {

      newChecked.**push**(value);

    } else {

      newChecked.**splice**(currentIndex, 1);

    }

    this.**setState**(

      {

        checked: newChecked

      },

      () => {

      // do your logic of child component and invoke parent method

this.props.**handleFilters**(newChecked);

      }

    );

  };

CLIENT SIDE- SHOP

Shop/index.js

import React, { Component } from "react";

import PageTop from "../utils/page\_top";

import { connect } from "react-redux";

import {

  getProductsToShop,

  getBrands,

  getWoods

} from "../../actions/products\_actions";

import CollapseCheckbox from "../utils/collapseCheckbox.js";

import CollapseRadio from "../utils/collapseRadio.js";

import { frets, price } from "../utils/Form/fixed\_categories";

class **Shop** extends **Component** {

  state = {

    grid: "",

    limit: 6,

    skip: 0,

    filters: {

      brand: [],

      frets: [],

      wood: [],

      price: []

    }

  };

**componentDidMount**() {

*// getting woods data*

    this.props.**dispatch**(**getWoods**());

    this.props.**dispatch**(**getBrands**());

    this.props.**dispatch**(

**getProductsToShop**(this.state.limit, this.state.skip, this.state.filters)

    );

  }

**……**

**……**

export default **connect**(**mapStateToProps**)(**Shop**);

components/reducers/product\_reducers.js

import {

  GET\_PRODUCTS\_BY\_SELL,

  GET\_PRODUCTS\_BY\_ARRIVAL,

  GET\_BRANDS,

  GET\_WOODS,

  GET\_PRODUCTS\_TO\_SHOP

} from "../actions/types";

export default function(state = {}, action) {

  switch (action.type) {

    case GET\_PRODUCTS\_BY\_SELL:

      return { ...state, bySell: action.payload };

    case GET\_PRODUCTS\_BY\_ARRIVAL:

      return { ...state, byArrival: action.payload };

    case GET\_BRANDS:

      return { ...state, brands: action.payload };

    case GET\_WOODS:

      return { ...state, woods: action.payload };

    case GET\_PRODUCTS\_TO\_SHOP:

      return {

        ...state,

        toShop: action.payload.articles,

        toShopSize: action.payload.size

      };

    default:

      return state;

  }

}

Components/action/product\_actions.js

import axios from "axios";

import {

  GET\_PRODUCTS\_BY\_SELL,

  GET\_PRODUCTS\_BY\_ARRIVAL,

  GET\_BRANDS,

  GET\_WOODS,

  GET\_PRODUCTS\_TO\_SHOP

} from "./types";

import { PRODUCT\_SERVER } from "../components/utils/misc";

export function **getProductsToShop**(

  skip,

  limit,

  filters = [],

  previousState = []

) {

  const data = {

    limit,

    skip,

    filters

  };

  const request = axios.**post**(`${PRODUCT\_SERVER}/shop`, data).**then**(response => {

    return {

      size: response.data.size,

      articles: response.data.articles

    };

  });

  return {

    type: GET\_PRODUCTS\_TO\_SHOP,

    payload: request

  };

}

SERVER – shop

Server/server.js

Node route for shop filter

app.**post**("/api/product/shop", (req, res) => {

  let order = req.body.order ? req.body.order : "desc";

  let sortBy = req.body.sortBy ? req.body.sortBy : "\_id";

  let limit = req.body.limit ? **parseInt**(req.body.limit) : 100; *// mongo db need this in int data type*

  let skip = **parseInt**(req.body.skip);

  let findArgs = {};

  for (let key in req.body.filters) {

    if (req.body.filters[key].length > 0) {

      if (key === "price") {

        findArgs[key] = {

          $gte: req.body.filters[key][0], // greater than equal to

          $lte: req.body.filters[key][1] // less than or equal to

        };

      } else {

*// brand, frets, wood*

        findArgs[key] = req.body.filters[key];

      }

    }

  }

*// searching in Mongo DB via schema*

  Product.**find**(findArgs)

    .**populate**("brand")

    .**populate**("wood")

    .**sort**([[sortBy, order]])

    .**skip**(skip)

    .**limit**(limit)

    .**exec**((err, articles) => {

      if (err) return res.**status**(400).**send**(err);

      res.**status**(200).**json**({

*// React Reducer expects size and articles - see reducer/produxt reducers*

        size: articles.length,

        articles

      });

    });

});

Product\_reducer.js

Path - Client/src/component/reducers/product\_reducer.js

import {

  GET\_PRODUCTS\_BY\_SELL,

  GET\_PRODUCTS\_BY\_ARRIVAL,

  GET\_BRANDS,

  GET\_WOODS,

  GET\_PRODUCTS\_TO\_SHOP

} from "../actions/types";

export default function(state = {}, action) {

  switch (action.type) {

    case GET\_PRODUCTS\_BY\_SELL:

      return { ...state, bySell: action.payload };

    case GET\_PRODUCTS\_BY\_ARRIVAL:

      return { ...state, byArrival: action.payload };

    case GET\_BRANDS:

      return { ...state, brands: action.payload };

    case GET\_WOODS:

      return { ...state, woods: action.payload };

    case GET\_PRODUCTS\_TO\_SHOP:

      return {

        ...state,

        toShop: action.payload.articles,

        toShopSize: action.payload.size

      };

    default:

      return state;

  }

}

Product\_actions.js – (redux)

Path - Client/src/component/actions/product\_action.js

export function **getProductsToShop**(

  skip,

  limit,

  filters = [],

  previousState = []

) {

  const data = {

    limit,

    skip,

    filters

  };

  const request = axios.**post**(`${PRODUCT\_SERVER}/shop`, data).**then**(response => {

    return {

      size: response.data.size,

      articles: response.data.articles

    };

  });

  return {

    type: GET\_PRODUCTS\_TO\_SHOP,

    payload: request

  };

}

Shopping cards

Components/shop/index.js

import React, { Component } from "react";

import PageTop from "../utils/page\_top";

import { connect } from "react-redux";

import {

  getProductsToShop,

  getBrands,

  getWoods

} from "../../actions/products\_actions";

import CollapseCheckbox from "../utils/collapseCheckbox.js";

import CollapseRadio from "../utils/collapseRadio.js";

import { frets, price } from "../utils/Form/fixed\_categories";

import LoadmoreCards from "./loadmoreCards";

class Shop extends Component {

  state = {

    grid: "",

    limit: 6,

    skip: 0,

    filters: {

      brand: [],

      frets: [],

      wood: [],

      price: []

    }

  };

…

…

  // will load next 6 products

  loadMoreCards = () => {

    let skip = this.state.skip + this.state.limit;

    this.props.dispatch(

      getProductsToShop(

        skip,

        this.state.limit,

        this.state.filters,

        this.props.products.toShop

      )).then(()=>{

        this.setState({

          skip

        });

      })

  }

  render() {

  …

…

          <div className="right">

              <div className="shop\_options">

                <div className="shop\_grids clear">grids</div>

              </div>

              <div>

                <LoadmoreCards

                  grid={this.state.grid}

                  limit={this.state.limit}

                  size={products.toShopSize}

                  products={products.toShop}

                  loadMore={ () => this.loadMoreCards() }

                />

              </div>

            </div>

          </div>

        </div>

      </div>

    );

  }

}

const mapStateToProps = state => {

  return {

    products: state.products

  };

};

export default connect(mapStateToProps)(Shop);

components/shop/loadmorecards.js

import React from "react";

import CardBlockShop from "../utils/card\_block\_shop";

const LoadmoreCards = props => {

  return (

    <div>

      <div>

        <CardBlockShop grid={props.grid} list={props.products} />

      </div>

      {

        props.size > 0 && props.size >= props.limit ?

          <div className="load\_more\_container">

              <span onClick={ ()=> props.loadMore() }>Load More</span>

          </div>

        :null

      }

    </div>

  );

};

export default LoadmoreCards;

src/components/utils/card\_block\_shop.js // reusable block card

import React from "react";

import Card from "../utils/card"; // reusable card

const CardBlockShop = props => {

  const renderCards = () => (

    props.list ?

      props.list.map(card => (

        <Card

          key={card.\_id}

          {...card}

          grid={props.grid}

        />

      ))

    :null

  )

  return (

    <div className="card\_block\_shop">

      <div>

        <div>

          {

            props.list ?

              props.list.length === 0 ?

                <div className="no\_result">

                    Sorry, no results

                </div>

              :null

            :null

          }

          { renderCards(props.list) }

        </div>

      </div>

    </div>

  );

};

export default CardBlockShop;

**CLoudinary – Upload image as a service**

video - <https://www.loom.com/share/86a14dfdf402416396d6d903d9bec05b>

add\_product.js

path – component/User/Admin/add\_product.js

import React, { Component } from "react";

import UserLayout from "../../../hoc/user";

import FormField from "../../utils/Form/formfield";

import {

  update,

  generateData,

  isFormValid,

  populateOptionFields,

  resetFields

} from "../../utils/Form/formActions";

import FileUpload from "../../utils/Form/fileupload";

import { connect } from "react-redux";

import {

  getBrands,

  getWoods,

  addProduct,

  clearProduct

} from "../../../actions/products\_actions";

…

…

imagesHandler = () => {};

  render() {

    return (

      <UserLayout>

        <div>

          <h1>Add Product</h1>

          <form onSubmit={event => this.submitForm(event)}>

            <FileUpload

              imagesHandler={images => this.imagesHandler(images)}

              reset={this.state.formSuccess}

            />

FormData - <https://javascript.info/formdata>

Dropzone

Fileupload.js

Path – component/utils/form/fileupload.js

import React, { Component } from "react";

import DropZone from "react-dropzone";

import axios from "axios";

import FontAwesomeIcon from "@fortawesome/react-fontawesome";

import faPlusCircle from "@fortawesome/fontawesome-free-solid/faPlusCircle";

import CircularProgress from "@material-ui/core/CircularProgress";

import { response } from "express";

class Fileupload extends Component {

  constructor() {

    super();

    this.state = {

      uploadedFiles: [],

      uploading: false

    };

  }

  // custom function - called when we upload file

  onDrop = files => {

    this.setState({ uploading: true });

    let formData = new FormData();

    const config = {

      header: { "content-type": "multipart/form-data" }

    };

    formData.append("file", files[0]);

    axios.post("/api/users/uploadimage", formData, config).then(response => {

      this.setState(

        {

          uploading: false,

          uploadedFiles: [...this.state.uploadedFiles, response.data]

        },

        () => {

          this.props.imagesHandler(this.state.uploadedFiles); // calling parent method

        }

      );

    });

  };

  showUploadedImages = () => {};

  render() {

    return (

      <div>

        <section>

          <div className="dropzone clear">

            <DropZone

              onDrop={e => this.onDrop(e)}

              multiple={false}

              className="dropzone\_box"

            >

              <div className="wrap">

                <FontAwesomeIcon icon={faPlusCircle} />

              </div>

            </DropZone>

            {this.showUploadedImages()}

            {this.state.uploading ? (

              <div

                className="dropzone\_box"

                style={{

                  textAlign: "center",

                  paddingTop: "60px"

                }}

              >

                <CircularProgress style={{ color: "#00bcd4" }} thickness={7} />

              </div>

            ) : null}

          </div>

        </section>

      </div>

    );

  }

}

export default Fileupload;

Cloudinary – accepting above axios request

Server.js

const express = require("express");

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

const cookieParser = require("cookie-parser");

const formidable = require("express-formidable");       // whenever request has file inside

const cloudinary = require("cloudinary");               // For using image as a service - storing in 3rd party

const app = express();

const mongoose = require("mongoose");

require("dotenv").config();

mongoose.Promise = global.Promise;

mongoose.connect(process.env.DATABASE);

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.use(cookieParser());

// in .env file set the value cloudinary gave us

cloudinary.config({

  cloud\_name: process.env.CLOUD\_NAME,

  api\_key: process.env.CLOUD\_API\_KEY,

  api\_secret: process.env.CLOUD\_API\_SECRET

});

…

…

app.post("/api/users/uploadimage", auth, admin, formidable(), (req, res)=>{

  // cloudinary.uploader.upload(req.file.file.path, ()=>{}, {})

  cloudinary.uploader.upload(req.files.file.path, (result)=>{     // file is the image name set in add-product.js

    console.log(result);

    res.status(200).send({

      public\_id: result.public\_id,

      url: result.url

    })

  }, {

    public\_id: `${Date.now()}`,     // any name to assign or hashed name

    resource\_type: "auto"           // accept any format image

  })

});

Fileupload.js – show image code

onRemove = id=>{}

  showUploadedImages = () => (

    // Cloudinary response from server, we get public\_id, url (i.e. image url) back. Check /api/users/uploadimage from server.js

    this.state.uploadedFiles.map(item => (

      <div className="dropzone\_box"

           key={item.public\_id}

           onClick={() => this.onRemove(item.public\_id)}

      >

          <div

            className="wrap"

            style={{background: `url(${item.url}) no-repeat`}}

          >

          </div>

      </div>

    ))

  )

add\_product.js – parent function for saving in db

path – component/User/Admin/add\_product.js

// Image url is passed from fileupload.js for saving in MongoDB

  imagesHandler = images => {

    const newFormData = {

      ...this.state.formdata

    }

    newFormData["images"].value = images;

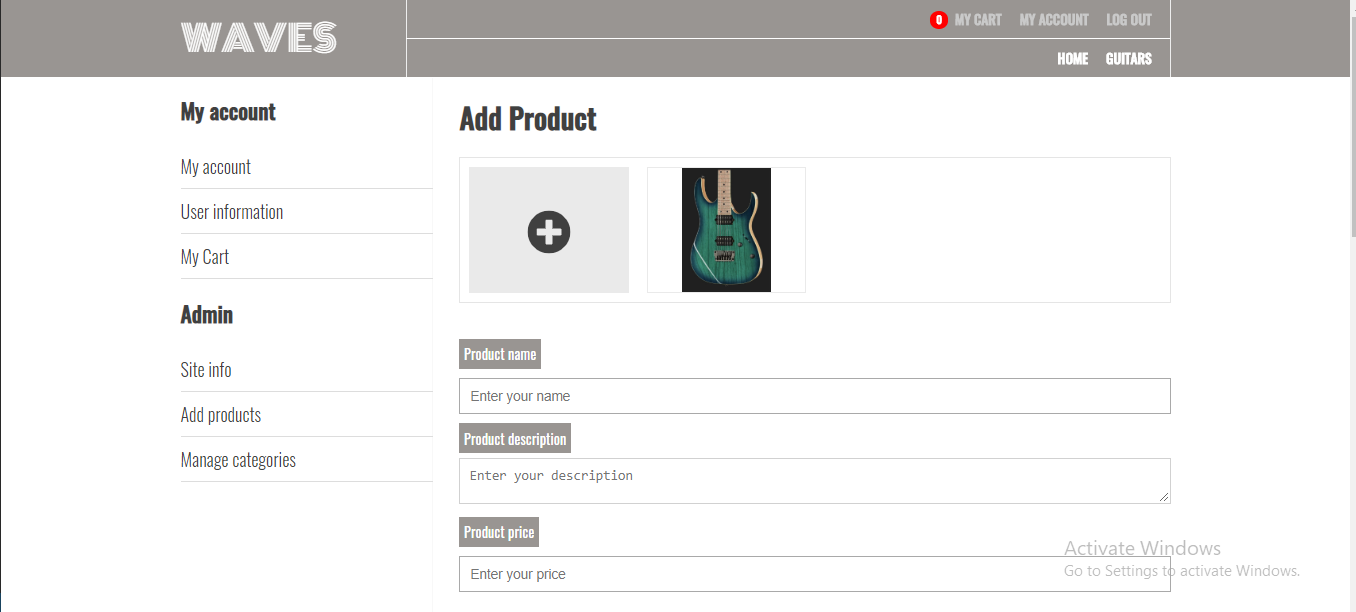
    newFormData["images"].valid = true;

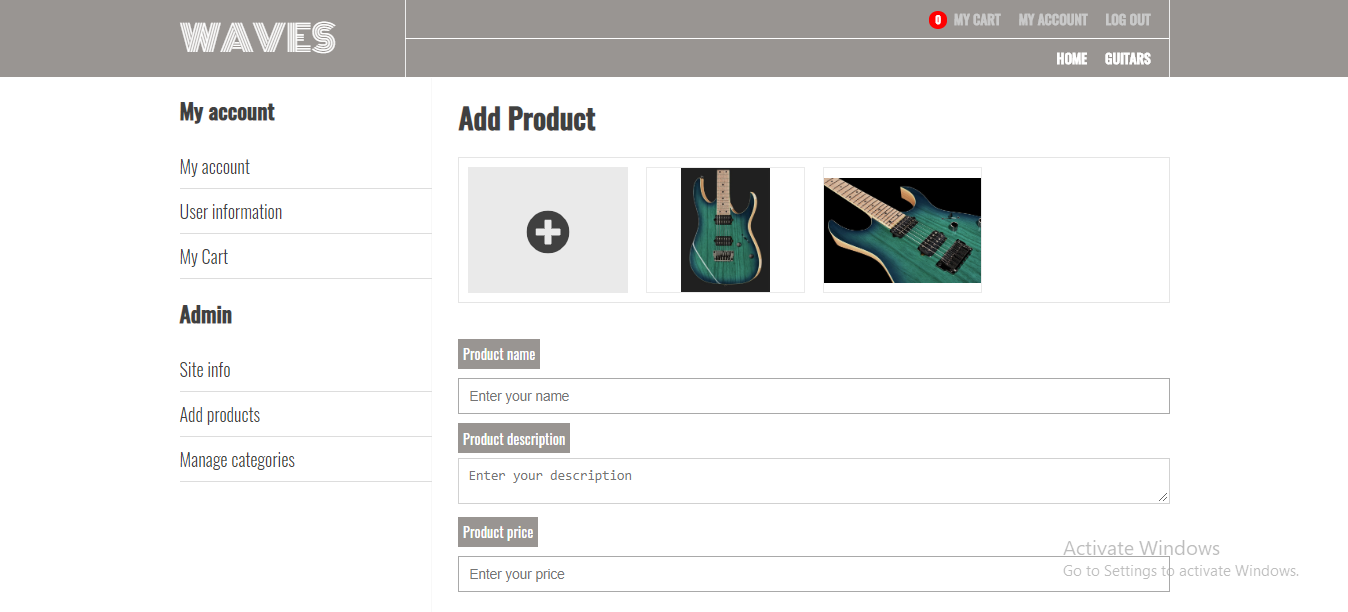
    this.setState({

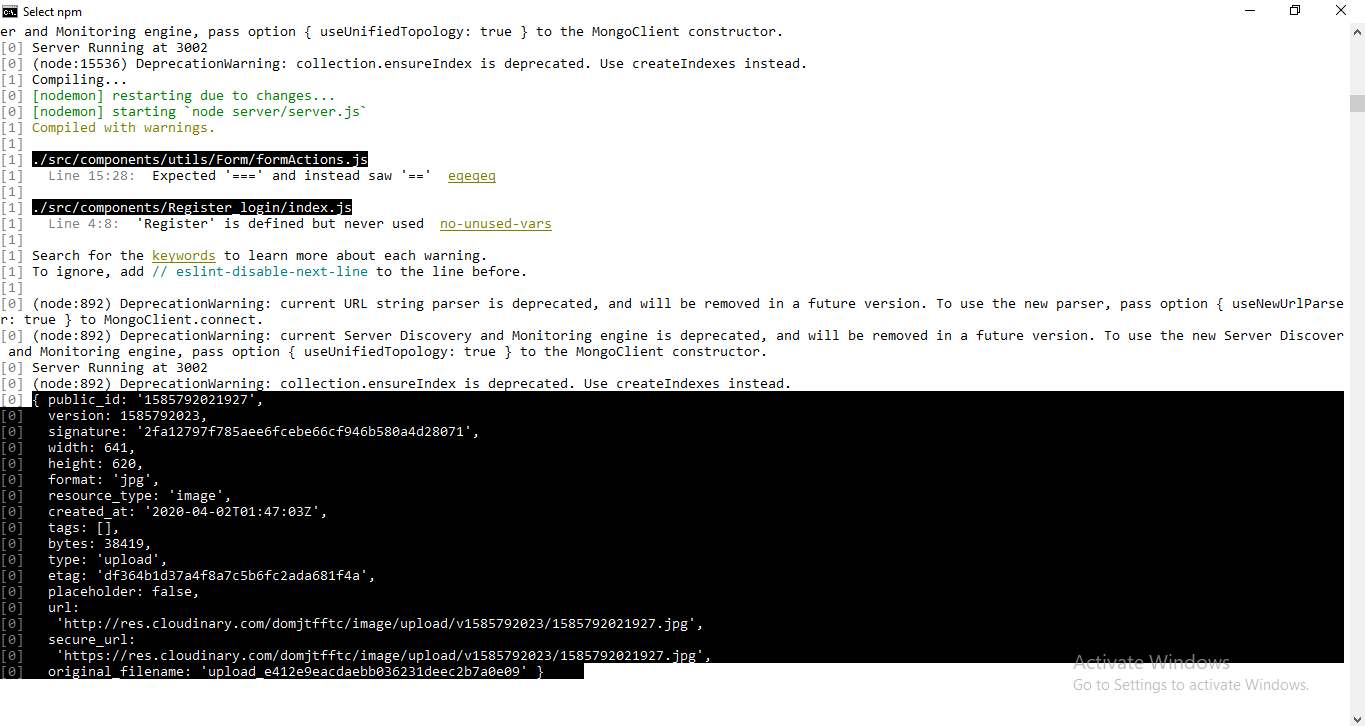
      formdata: newFormData

    });

  };







Remove dropzone image on image click (remove state)+ delete from cloudinary (done from node)

Client: Fileupload.js (sending request to node server)

 // Remove image to delete from cloudinary and state

  onRemove = (id) => {

    axios.get(`/api/users/removeimage?public\_id=${id}`).then((response) => {

      let images = this.state.uploadedFiles.filter((item) => {

        return item.public\_id !== id;

      });

      this.setState(

        {

          uploadedFiles: images,

        },

        () => {

          this.props.imagesHandler(images);

        }

      );

    });

  };

Server.js (added this below before port code)

app.get("/api/users/removeimage", auth, admin, (req, res) => {

  let image\_id = req.query.public\_id;

  cloudinary.uploader.destroy(image\_id, (error, result) => {

    if (error) return res.json({ succes: false, error });

    res.status(200).send("ok");

  });

});

const port = process.env.PORT || 3002;

app.listen(port, () => {

  console.log(`Server Running at ${port}`);

});

For handeling clearing image after file upload:

Fileupload.js

Path – component/utils/form/formupload.js

//This method is called before render

static getDerivedStateFromProps(props, state) {

    if (props.reset) {

      return (state = {

        uploadFiles: [],

      });

    }

    return null;

  }

  render() {

    return (

      <div>

…

…

View products from guitar link

* When we click guitar link from menu we see list of products
* Clicking on view products shows products details and images with light box
* Product id is in url

Src/components/Product/index.js

import React, { Component } from "react";

import PageTop from "../utils/page\_top";

import ProdNfo from "./prodNfo";

import { connect } from "react-redux";

import {

  getProductDetail,

  clearProductDetail,

} from "../../actions/products\_actions";

class ProductPage extends Component {

  componentDidMount() {

    let id = this.props.match.params.id;

    this.props.dispatch(getProductDetail(id));

  }

  componentWillUnmount() {

    this.props.dispatch(clearProductDetail());

  }

  render() {

    return (

      <div>

        <PageTop title="Product detail" />

        <div className="container">

          {this.props.products.prodDetail ? (

            <div className="product\_detail\_wrapper">

              <div className="left">images</div>

              <div className="right">

                <ProdNfo

                  addToCart={(id) => this.addToCartHandler(id)}

                  detail={this.props.products.prodDetail}

                />

              </div>

            </div>

          ) : (

            "Loading"

          )}

        </div>

      </div>

    );

  }

}

const mapStateToProps = (state) => {

  return {

    products: state.products,

  };

};

export default connect(mapStateToProps)(ProductPage);

src/component/Product/prodNfo.js

import React from "react";

import MyButton from "../utils/button";

import FontAwesomeIcon from "@fortawesome/react-fontawesome";

import faTruck from "@fortawesome/fontawesome-free-solid/faTruck";

import faCheck from "@fortawesome/fontawesome-free-solid/faCheck";

import faTimes from "@fortawesome/fontawesome-free-solid/faTimes";

const ProdNfo = (props) => {

  const showProdTags = (detail) => (

    <div className="product\_tags">

      {detail.shipping ? (

        <div className="tag">

          <div>

            <FontAwesomeIcon icon={faTruck} />

          </div>

          <div className="tag\_text">

            <div>Free shipping</div>

            <div>And return</div>

          </div>

        </div>

      ) : null}

      <div className="tag">

        <div>

          <FontAwesomeIcon icon={detail.available ? faCheck : faTimes} />

        </div>

        <div className="tag\_text">

          <div>{detail.available ? "Available" : "Not available"}</div>

          <div>{detail.available ? "In stock" : "Preorder only"}</div>

          <div></div>

        </div>

      </div>

    </div>

  );

  const showProdAction = (detail) => (

    <div className="product\_actions">

      <div className="price">$ {detail.price}</div>

      <div className="cart">

        <MyButton

          type="add\_to\_cart\_link"

          runAction={() => {

            console.log("add to cart");

          }}

        />

      </div>

    </div>

  );

  const showProdSpecifications = (detail) => (

    <div className="product\_specifications">

      <h2>Specs:</h2>

      <div>

        <div className="item">

          <strong>Frets: </strong> {detail.frets}

        </div>

        <div className="item">

          <strong>Wood: </strong> {detail.wood.name}

        </div>

      </div>

    </div>

  );

  const detail = props.detail;

  return (

    <div>

      <h1>

        {detail.brand.name} {detail.name}

      </h1>

      <p>{detail.description}</p>

      {showProdTags(detail)}

      {showProdAction(detail)}

      {showProdSpecifications(detail)}

    </div>

  );

};

export default ProdNfo;

src/components/utils/button.js

added this case to button switch

case "add\_to\_cart\_link":

        template = (

          <div

            className="add\_to\_cart\_link"

            onClick={() => {

              props.runAction();

            }}

          >

            <FontAwesomeIcon icon={faShoppingBag} />

            Add to cart

          </div>

        );

        break;

Adding images to product

* Create prodImg.js component using in product/index.js
* Create lightbox.js reusable component using in product/index.js

Components/productsIndex.js

…

import ProdImg from "./prodImg";

…

…

render() {

    return (

      <div>

        <PageTop title="Product detail" />

        <div className="container">

          {this.props.products.prodDetail ? (

            <div className="product\_detail\_wrapper">

              <div className="left">

                <div style={{ width: "500px" }}>

                  <ProdImg detail={this.props.products.prodDetail} />

                </div>

              </div>

              <div className="right">

                <ProdNfo

                  addToCart={(id) => this.addToCartHandler(id)}

                  detail={this.props.products.prodDetail}

                />

              </div>

            </div>

          ) : (

            "Loading"

          )}

        </div>

      </div>

    );

prodImg.js – handeling layouts of image, multiple images and sending props to reusable lightbox.js component

import React, { Component } from "react";

import ImageLightBox from "../utils/lightbox";

class ProdImg extends Component {

  state = {

    lightbox: false,

    imagePos: 0,

    lightboxImages: [],

  };

  componentDidMount() {

    if (this.props.detail.images.length > 0) {

      let lightboxImages = [];

      this.props.detail.images.forEach((item) => {

        lightboxImages.push(item.url);

      });

      this.setState({

        lightboxImages,

      });

    }

  }

  handleLightBox = (pos) => {

    if (this.state.lightboxImages.length > 0) {

      this.setState({

        lightbox: true,

        imagePos: pos,

      });

    }

  };

  handleLightBoxClose = () => {

    this.setState({

      lightbox: false,

    });

  };

  // show remaining images, skip first

  showThumbs = () =>

    this.state.lightboxImages.map((item, i) =>

      i > 0 ? (

        <div

          key={i}

          onClick={() => this.handleLightBox(i)}

          className="thumb"

          style={{ background: `url(${item}) no-repeat` }}

        ></div>

      ) : null

    );

  // If image is present return image, otherwise return default image

  renderCardImage = (images) => {

    if (images.length > 0) {

      return images[0].url;

    } else {

      return `/images/image\_not\_availble.png`;

    }

  };

  render() {

    const { detail } = this.props;

    return (

      <div className="product\_image\_container">

        <div className="main\_pic">

          <div

            style={{

              background: `url(${this.renderCardImage(

                detail.images

              )}) no-repeat`,

            }}

            onClick={() => this.handleLightBox(0)}

          ></div>

        </div>

        <div className="main\_thumbs">{this.showThumbs(detail)}</div>

        {this.state.lightbox ? (

          <ImageLightBox

            id={detail.id}

            images={this.state.lightboxImages}

            open={this.state.open}

            pos={this.state.imagePos}

            onclose={() => this.handleLightBoxClose()}

          />

        ) : null}

      </div>

    );

  }

}

export default ProdImg;

components/utils/lightbox.js – Reusable component

import React, { Component } from "react";

import Lightbox from "react-images";

class ImageLightBox extends Component {

  state = {

    lightboxIsOpen: true,

    currentImage: this.props.pos,

    images: [],

  };

  // changing image sturucture in state to [{src: imag1},{src: imag2}, .. ] from [img1, img2,..]

  static getDerivedStateFromProps(props, state) {

    if (props.images) {

      const images = [];

      props.images.forEach((element) => {

        images.push({ src: `${element}` });

      });

      return (state = {

        images,

      });

    }

    return false;

  }

  gotoPrevious = () => {

    this.setState({

      currentImage: this.state.currentImage - 1,

    });

  };

  gotoNext = () => {

    this.setState({

      currentImage: this.state.currentImage + 1,

    });

  };

  closeLightbox = () => {

    this.props.onclose();

  };

  render() {

    return (

      <Lightbox

        currentImage={this.state.currentImage}

        images={this.state.images}

        isOpen={this.state.lightboxIsOpen}

        onClickPrev={() => this.gotoPrevious()}

        onClickNext={() => this.gotoNext()}

        onClose={() => this.closeLightbox()}

      />

    );

  }

}

export default ImageLightBox;

**Add to Cart functionality – dispatch to redux**

Now we will do **dispatch** to add to cart in **two places**:

1. Shop page (on clicking guitar from menu)
2. Product detail page (view product from shop page)

Clicking add to cart shall either include new product to cart or increase the quantity for duplicate product

**1. Shop page**

Node JS:

Server/server.js – added route for Add to cart

app.post("/api/users/addToCart", auth, (req, res) => {

  User.findOne({ \_id: req.user.\_id }, (err, doc) => {

    let duplicate = false;

    doc.cart.forEach((item) => {

      if (item.id == req.query.productId) {

        duplicate = true;

      }

    });

    // logic for add to cart

    if (duplicate) {

    } else {

      // new product

      User.findOneAndUpdate(

        { \_id: req.user.\_id }, // id of user

        {

          $push: {

            // data to push

            cart: {

              id: mongoose.Types.ObjectId(req.user.\_id), // pushing id in mongoose \_id way

              quantity: 1,

              date: Date.now(),

            },

          },

        },

        { new: true }, // set this true when you want the doc back as reponse

        (err, doc) => {

          // callback

          if (err) return res.json({ success: false, err });

          res.status(200).json(doc.cart);

        }

      );

    }

  });

});

Client (React JS) – card js – connect with redux and send req through axios

Components/utils/card.js

import React, { Component } from "react";

import MyButton from "./button";

import { connect } from "react-redux";

import { addToCart } from "../../actions/user\_actions";

class Card extends Component {

  renderCardImage(images) {

    if (images.length > 0) {

      return images[0].url;

    } else {

      return "/images/image\_not\_availble.png";

    }

  }

  render() {

    const props = this.props;

    return (

      <div className={`card\_item\_wrapper ${props.grid}`}>

        <div

          className="image"

          style={{

            background: `url(${this.renderCardImage(props.images)}) no-repeat`,

          }}

        ></div>

        <div className="action\_container">

          <div className="tags">

            <div className="brand">{props.brand.name}</div>

            <div className="name">{props.name}</div>

            <div className="name">${props.price}</div>

          </div>

          {props.grid ? (

            <div className="description">

              <p>{props.description}</p>

            </div>

          ) : null}

          <div className="actions">

            <div className="button\_wrapp">

              <MyButton

                type="default"

                altClass="card\_link"

                title="View product"

                linkTo={`product\_detail/${props.\_id}`}

                addStyles={{

                  margin: "10px 0 0 0",

                }}

              />

            </div>

            <div className="button\_wrapp">

              <MyButton

                type="bag\_link"

                runAction={() => {

                  props.user.userData.isAuth

                    ? this.props.dispatch(addToCart(props.\_id)) // DISPATCH

                    : alert(

                        "You need to log in first for purchasing this product."

                      );

                }}

              />

            </div>

          </div>

        </div>

      </div>

    );

  }

}

const mapStateToProps = (state) => {

  return {

    user: state.user,

  };

};

export default connect(mapStateToProps)(Card);

components/actions/user\_actions.js

import {

  LOGIN\_USER,

  REGISTER\_USER,

  AUTH\_USER,

  LOGOUT\_USER,

  ADD\_TO\_CART\_USER,

} from "./types";

export function addToCart(\_id) {

  const request = axios

    .post(`${USER\_SERVER}/addToCart?productId=${\_id}`)

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

  return {

    type: ADD\_TO\_CART\_USER,

    payload: request,

  };

}

Types.js

export const ADD\_TO\_CART\_USER = "add\_to\_cart\_user";

user\_reducers.js

case ADD\_TO\_CART\_USER:

      return {

        ...state,

        userData: {

          ...state.userData, // all userdata stays as it is

          cart: action.payload, // updating the cart with its current state

        },

      };

Add to cart from Product Detail page – 2nd dispatch

Components/product/prodNfo.js

const showProdAction = (detail) => (

    <div className="product\_actions">

      <div className="price">$ {detail.price}</div>

      <div className="cart">

        <MyButton

          type="add\_to\_cart\_link"

          runAction={() => {

            props.addToCart(detail.\_id)// Passing prod Id to parent container

          }}

        />

      </div>

    </div>

  );

Component/products/index.js

import { addToCart } from "../../actions/user\_actions";

…

…

addToCartHandler(id) {

    this.props.dispatch(addToCart(id)); // DISPATCH

  }

So dispatch will either add a product to cart or for duplicate product increase qty by 1

Cart component – get all the cart items

Src/component/Users/cart.js

import React, { Component } from 'react';

import UserLayout from "../../hoc/user";

import { connect } from "react-redux";

import { getCartItems } from "../../actions/user\_actions";

import FontAwesomeIcon from "@fortawesome/react-fontawesome";

import faFrown from "@fortawesome/fontawesome-free-solid/faFrown";

import faSmile from "@fortawesome/fontawesome-free-solid/faSmile";

class UserCart extends Component {

    state = {

        loading: true,

        total: 0,

        showTotal: false,

        showSuccess: false

    }

    componentDidMount() {

        let cartItem = [];

        let user = this.props.user;

        // Pass array of product ids to the node server

        if (user.userData.cart && user.userData.cart.length > 0) {

            user.userData.cart.forEach(item => {

                cartItem.push(item.id);

            });

// redux

            this.props.dispatch(getCartItems(cartItem, user.userData.cart));

        }

    }

    render() {

        return (

            <UserLayout>

                <div>

                    cart

                </div>

            </UserLayout>

        )

    }

}

const mapStateToProps = (state) => {

    return {

        user: state.user

    }

}

export default connect(mapStateToProps)(UserCart);

user\_actions.js

export function getCartItems(cartItems, userCart) {

  const request = axios.get(`${PRODUCT\_SERVER}/articles\_by\_id?id=${cartItems}&type=array`)

    .then(response => {

      // inject quantity of products inside response

      userCart.forEach(item => {

        response.data.forEach((k, i) => {

          if (item.id === k.\_id) {

            response.data[i].quantity = item.quantity

          }

        })

      });

      return response.data;

    })

  return {

    type: GET\_CART\_ITEMS\_USER,

    payload: request

  }

}

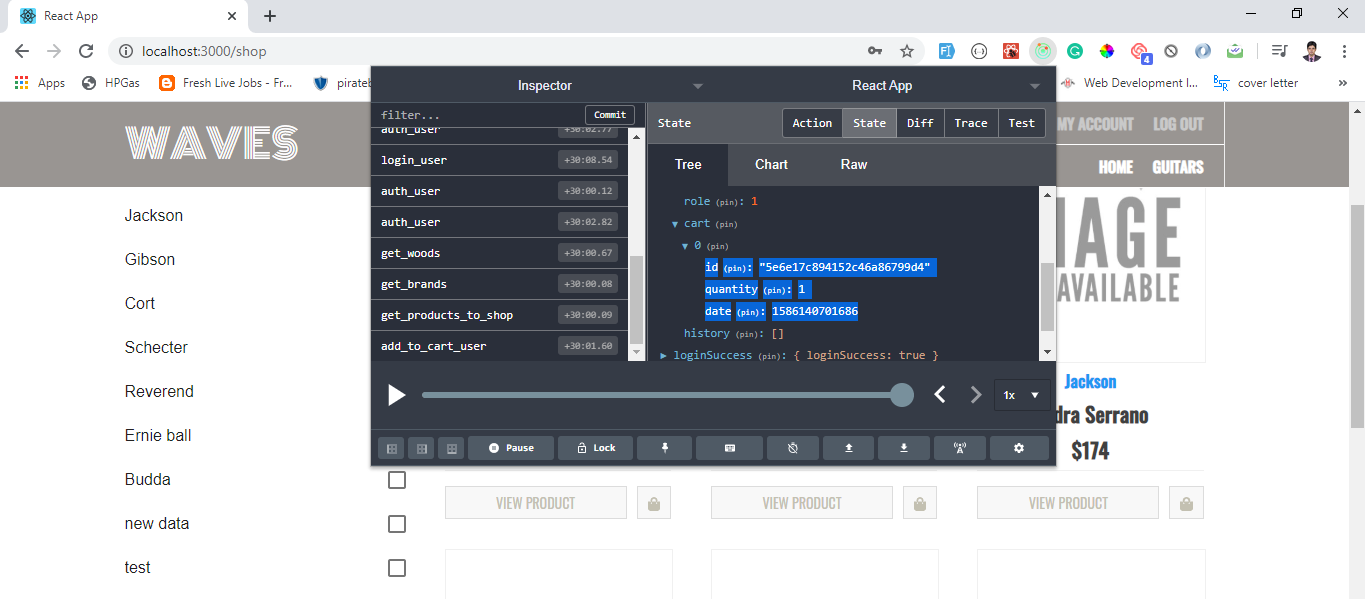
User\_reducers.js

…

…

case GET\_CART\_ITEMS\_USER:

      return { ...state, cartDetail: action.payload }



Show carts layouts:

Component/User/cart.js

import UserProductBlock from "../utils/User/product\_ block";

removeFromCart = () => { };

render() {

        return (

            <UserLayout>

                <div>

                    <h1>My cart</h1>

                    <div className="user\_cart">

                        <UserProductBlock

                            products={this.props.user}

                            type="cart"

                            removeItem={(id) => { this.removeFromCart(id) }}

                        />

                    </div>

                </div>

            </UserLayout>

        )

    }

Components/utils/product\_block.js – Reusable component

import React from 'react'

const UserProductBlock = ({ products, removeItem }) => {

    const renderCartImage = (images) => {

        if (images.length > 0) {

            return images[0].url;

        } else {

            return "/images/image\_not\_availble.png"

        }

    }

    const renderItems = () => (

        products.cartDetail ?

            products.cartDetail.map(product => (

                <div className="user\_product\_block" key={product.\_id}>

                    <div className="item">

                        <div

                            className="image"

                            style={{ background: `url(${renderCartImage(product.images)}) no-repeat` }}

                        ></div>

                    </div>

                    <div className="item">

                        <h4>Product name</h4>

                        <div>

                            {product.brand.name} {product.name}

                        </div>

                    </div>

                    <div className="item">

                        <h4>Quantity</h4>

                        <div>

                            {product.quantity}

                        </div>

                    </div>

                    <div className="item">

                        <h4>Price</h4>

                        <div>

                            $ {product.price}

                        </div>

                    </div>

                    <div className="item btn">

                        <div className="cart\_remove\_btn"

                            onClick={() => removeItem(product.\_id)}>

                            Remove

                        </div>

                    </div>

                </div>

            ))

            : null

    )

    return (

        <div>

            {renderItems()}

        </div>

    )

}

export default UserProductBlock

cart.js

import React, { Component } from 'react';

import UserLayout from "../../hoc/user";

import UserProductBlock from "../utils/User/product\_ block";

import { connect } from "react-redux";

import { getCartItems } from "../../actions/user\_actions";

import FontAwesomeIcon from "@fortawesome/react-fontawesome";

import faFrown from "@fortawesome/fontawesome-free-solid/faFrown";

import faSmile from "@fortawesome/fontawesome-free-solid/faSmile";

class UserCart extends Component {

    state = {

        loading: true,

        total: 0,

        showTotal: false,

        showSuccess: false

    }

    componentDidMount() {

        let cartItems = [];

        let user = this.props.user;

        // Pass array of product ids to the node server

        if (user.userData.cart && user.userData.cart.length > 0) {

            user.userData.cart.forEach(item => {

                cartItems.push(item.id);

            });

            this.props.dispatch(getCartItems(cartItems, user.userData.cart))

                .then(() => {

                    if (this.props.user.cartDetail.length > 0) {

                        this.calculateTotal(this.props.user.cartDetail);

                    }

                });

        }

    }

    calculateTotal = (cartDetail) => {

        let total = 0;

        cartDetail.forEach(item => {

            total += parseInt(item.price, 10) \* item.quantity

        });

        this.setState({

            total,

            showTotal: true

        });

    };

    showNoItemsMessage = () => (

        <div className="cart\_no\_items">

            <FontAwesomeIcon icon={faFrown} />

            <div>You have no items</div>

        </div>

    )

    removeFromCart = () => { };

    render() {

        return (

            <UserLayout>

                <div>

                    <h1>My cart</h1>

                    <div className="user\_cart">

                        <UserProductBlock

                            products={this.props.user}

                            type="cart"

                            removeItem={(id) => { this.removeFromCart(id) }}

                        />

                        {

                            this.state.showTotal ?

                                <div>

                                    <div className="user\_cart\_sum">

                                        <div>

                                            Total: $ {this.state.total}

                                        </div>

                                    </div>

                                </div>

                                :

                                this.state.showSuccess ?

                                    <div className="cart\_success">

                                        <FontAwesomeIcon icon={faSmile} />

                                        <div>THANK YOU</div>

                                        <div>YOUR ORDER IS NOW COMPLETE</div>

                                    </div>

                                    :

                                    this.showNoItemsMessage()

                        }

                    </div>

                    {

                        this.state.showTotal ?

                            <div className="paypal\_button\_container">

                                Paypal

                            </div>

                            : null

                    }

                </div>

            </UserLayout>

        )

    }

}

const mapStateToProps = (state) => {

    return {

        user: state.user

    }

}

export default connect(mapStateToProps)(UserCart);

product\_block.js – reusable component

import React from 'react'

const UserProductBlock = ({ products, removeItem }) => {

    const renderCartImage = (images) => {

        if (images.length > 0) {

            return images[0].url;

        } else {

            return "/images/image\_not\_availble.png"

        }

    }

    const renderItems = () => (

        products.cartDetail ?

            products.cartDetail.map(product => (

                <div className="user\_product\_block" key={product.\_id}>

                    <div className="item">

                        <div

                            className="image"

                            style={{ background: `url(${renderCartImage(product.images)}) no-repeat` }}

                        ></div>

                    </div>

                    <div className="item">

                        <h4>Product name</h4>

                        <div>

                            {product.brand.name} {product.name}

                        </div>

                    </div>

                    <div className="item">

                        <h4>Quantity</h4>

                        <div>

                            {product.quantity}

                        </div>

                    </div>

                    <div className="item">

                        <h4>Price</h4>

                        <div>

                            $ {product.price}

                        </div>

                    </div>

                    <div className="item btn">

                        <div className="cart\_remove\_btn"

                            onClick={() => removeItem(product.\_id)}>

                            Remove

                        </div>

                    </div>

                </div>

            ))

            : null

    )

    return (

        <div>

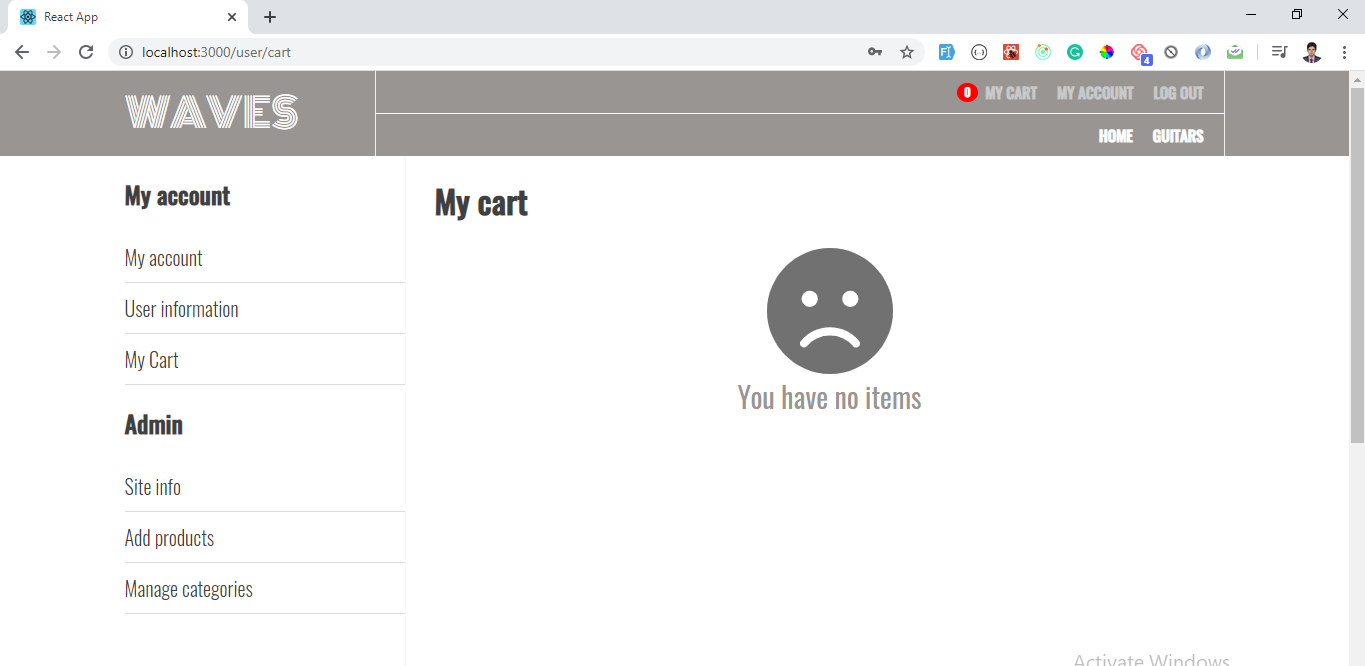
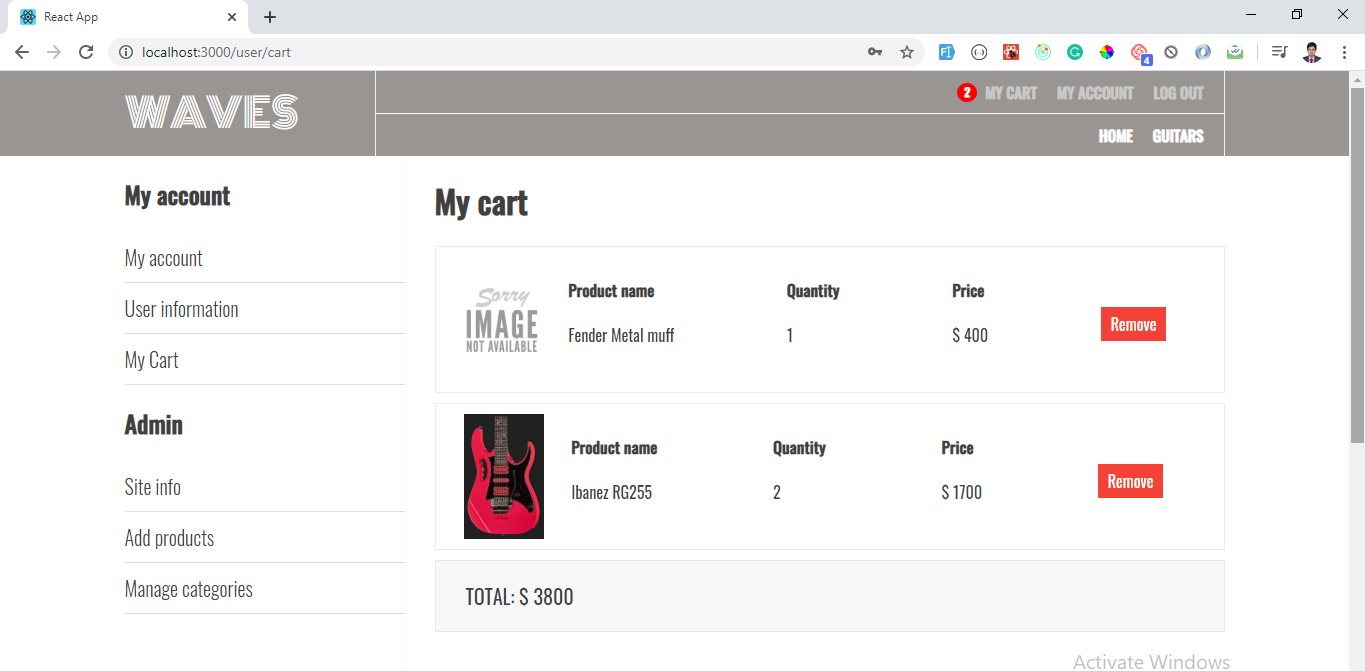
            {renderItems()}

        </div>

    )

}

export default UserProductBlock



PAYPAL Video - <https://www.loom.com/share/a338696d438c4ed5b86c73dcd2158060?focus_title=1&muted=1>

<https://www.loom.com/share/f53c9b983351488c9b7f2f9bd73a186d>

accept usd from paypal

add money to personal account

[amirengg15@outlook.com](mailto:amirengg15@outlook.com)

Admin@123

**PayPal - Way to add money in Sandbox account**

1. Login using your developer PayPal account <https://developer.paypal.com/>
2. Go to 'My Account' on the top right section of the page.
3. Go to Accounts under Sandbox.
4. See your sandbox account and click on the Clone link.
5. Fill the details and mention the amount you want to have and click on Create.

This is helpful in case if the amount in the existing sandbox account is exhausted.

Check all personal and business a/c here:

<https://developer.paypal.com/developer/accounts/>

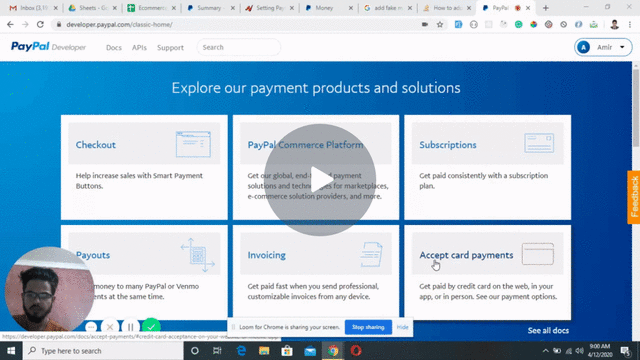
Test Ids:

Business: [amirengg15@business.example.com](mailto:amirengg15@business.example.com), Admin@123

Personal: [amirengg15@personal.example.com](mailto:amirengg15@personal.example.com), Admin@123

**Complete Video -**

[PayPal Developer Documentation - PayPal Developer -](https://www.loom.com/share/c98c625aeaba4ceea9ec70f77eebc36b) [Watch Video](https://www.loom.com/share/c98c625aeaba4ceea9ec70f77eebc36b)

[](https://www.loom.com/share/c98c625aeaba4ceea9ec70f77eebc36b?utm_medium=gif)

Payment Gateway –

Cart.js

Path = client/src/components/User/cart.js

import React, { Component } from "react";

…

…

import Paypal from "../utils/paypal";

class UserCart extends Component {

  state = {

    loading: true,

    total: 0,

    showTotal: false,

    showSuccess: false

  };

…

  …

transactionError = data => {

    console.log("Transaction error", data);

  };

  transactionCancelled = data => {

    console.log("Transaction cancelled", data);

  };

  transactionSuccess = data => {

    this.setState({

      showTotal: false,

      showSuccess: true

    });

  };

  render() {

    return (

      <UserLayout>

{this.state.showTotal ? (

            <div className="paypal\_button\_container">

              <Paypal

                toPay={this.state.total}

                transactionError={data => this.transactionError(data)}

                transactionCanceled={data => this.transactionCancelled(data)}

                onSuccess={data => this.transactionSuccess(data)}

              />

            </div>

          ) : null}

        </div>

      </UserLayout>

    );

  }

}

const mapStateToProps = state => {

  return {

    user: state.user

  };

};

export default connect(mapStateToProps)(UserCart);

paypal.js – **reusable component**

path = client/src/components/utils/paypal.js

import React, { Component } from "react";

import PaypalExpressBtn from "react-paypal-express-checkout";

class Paypal extends Component {

  render() {

    const onSuccess = payment => {

      this.props.onSuccess(payment);

      //   let payment = {

      //     paid: true,

      //     cancelled: false,

      //     payerID: "5TE27DDEFHFJG",

      //     paymentID: "PAYID-L2JLDDQ1HH244895K6025509",

      //     paymentToken: "EC-6UA68144215181137",

      //     returnUrl:

      //       "https://www.paypal.com/checkoutnow/error?paymentId=PAYID-L2JLDDQ1HH244895K6025509&token=EC-6UA68144215181137&PayerID=5TE27DDEFHFJG",

      //     address: {

      //       recipient\_name: "Amir Mustafa",

      //       line1: "1 Main St",

      //       city: "San Jose",

      //       state: "CA",

      //       postal\_code: "95131",

      //       country\_code: "US"

      //     },

      //     email: "amirengg15@personal.example.com"

      //   };

    };

    const onCancel = data => {

      // console.log(JSON.stringify(data));

      this.props.transactionCancelled(data);

    };

    const onError = err => {

      // console.log(JSON.stringify(err));

      this.props.transactionError(err);

    };

    let env = "sandbox";

    let currency = "USD";

    let total = this.props.toPay;

    const client = {

      sandbox:

        "AXZTnEjiZ6tr6YA3HnkSeaYwBiZdlIKPVUB\_xapgqOB6kZht-N0N-JOfNxoLk\_BKmFgEP0QxKpyPv-lt",

      production: ""

    };

    return (

      <div>

        <PaypalExpressBtn

          env={env}

          client={client}

          currency={currency}

          total={total}

          onError={onError}

          onSuccess={onSuccess}

          onCancel={onCancel}

          style={{

            size: "large",

            color: "blue",

            shape: "rect",

            label: "checkout"

          }}

        />

      </div>

    );

  }

}

export default Paypal;

Success Buy Route

Server/server.js

app.post("/api/users/successBuy", auth, (req, res) => {

  let history = [];

  let transactionData = {};

  // USER HISTORY

  req.body.cartDetail.forEach((item) => {

    history.push({

      dateOfPurchase: Date.now(),

      name: item.name,

      brand: item.brand.name,

      id: item.\_id,

      price: item.price,

      quantity: item.quantity,

      paymentId: req.body.paymentData.paymentID,

    });

  });

  // PAYMENT DASHBOARD

  transactionData.user = {

    id: req.user.\_id, // Data comes from auth middleware

    name: reqmuser.name,

    lastname: reqmuser.lastname,

    email: reqmuser.email,

  };

  transactionData.data = req.body.paymentData;

  transactionData.product = history;

  // Push data in user collection - save in user's history and empty the cart in db

  User.findByIdAndUpdate(

    { \_id: req.user.\_id }, // from auth

    { $push: { history: history }, $set: { cart: [] } },

    { new: true }, // to get latest doc back

    (err, user) => {

      if (err) return res.json({ success: false, err });

      const payment = new Payment(transactionData);

      payment.save((err, doc) => {

        if (err) return res.json({ success: false, err });

        // Increment sold document of Product collection by quantity - after purchase to track qty of that product sold

        let products = [];

        doc.product.forEach((item) => {

          products.push({ id: item.id, quantity: item.quantity });

        });

        async.eachOfSeries(

          products,

          (item, callback) => {

            // update sold of Product collection

            Product.update(

              { \_id: item.id },

              {

                $inc: {

                  sold: item.quantity,

                },

              },

              { new: false }, // as we do not need new document back

              callback

            );

          },

          (err) => {

            if (err) return res.json({ success: false, err });

            res.status(200).json({

              success: true,

              cart: user.cart, // this cart is empty

              cartDetail: [],

            });

          }

        );

      });

    }

  );

});

TRICK – Async (Node Package)

* Suppose there is a function component which has callback and that runs in a loop, means multiple callback.SO application will be slow.
* So if we want to run all function in a loop and one callback in the end, we use this

Npm install async

async.eachSeries( // this is a kind of loop

          products, // array to loop

          (item, callback) => {

            // update sold of Product collection

            Product.update(

              { \_id: item.id },

              {

                $inc: {

                  sold: item.quantity, // increasing qty by 1

                },

              },

              { new: false }, // as we do not need new document back

              callback

            );

          },

          (err) => { // final callback – called once

            if (err) return res.json({ success: false, err });

            res.status(200).json({

              success: true,

              cart: user.cart, // this cart is empty

              cartDetail: [],

            });

          }

        );

Now triggering the action dispatch for success buy from react and redux

Actions/User\_actions.js

import {

  LOGIN\_USER,

  REGISTER\_USER,

  AUTH\_USER,

  LOGOUT\_USER,

  ADD\_TO\_CART\_USER,

  GET\_CART\_ITEMS\_USER,

  REMOVE\_CART\_ITEM\_USER,

  ON\_SUCCESS\_BUY\_USER

} from "./types";

export function onSuccessBuy(data) {

  const request = axios

    .post(`${USER\_SERVER}/successBuy`, data)

    .then(response => response.data);

  return {

    type: ON\_SUCCESS\_BUY\_USER,

    payload: request

  };

}

Reducers/user\_reducers.js

…

…

case ON\_SUCCESS\_BUY\_USER:

      return {

        ...state,

        successBuy: action.payload.success,

        userData: {

          ...state.userData,

          cart: action.payload.cart

        },

        cartDetail: action.payload.cartDetail

      };

…

Components/Users/cart.js

import {

  getCartItems,

  removeCartItem,

  onSuccessBuy

} from "../../actions/user\_actions";

…

…

transactionSuccess = data => {

    this.props

      .dispatch(

        onSuccessBuy({

          cartDetail: this.props.user.cartDetail,

          paymentData: data

        })

      )

      .then(() => {

        if (this.props.user.successBuy) {

          this.setState({

            showTotal: false,

            showSuccess: true

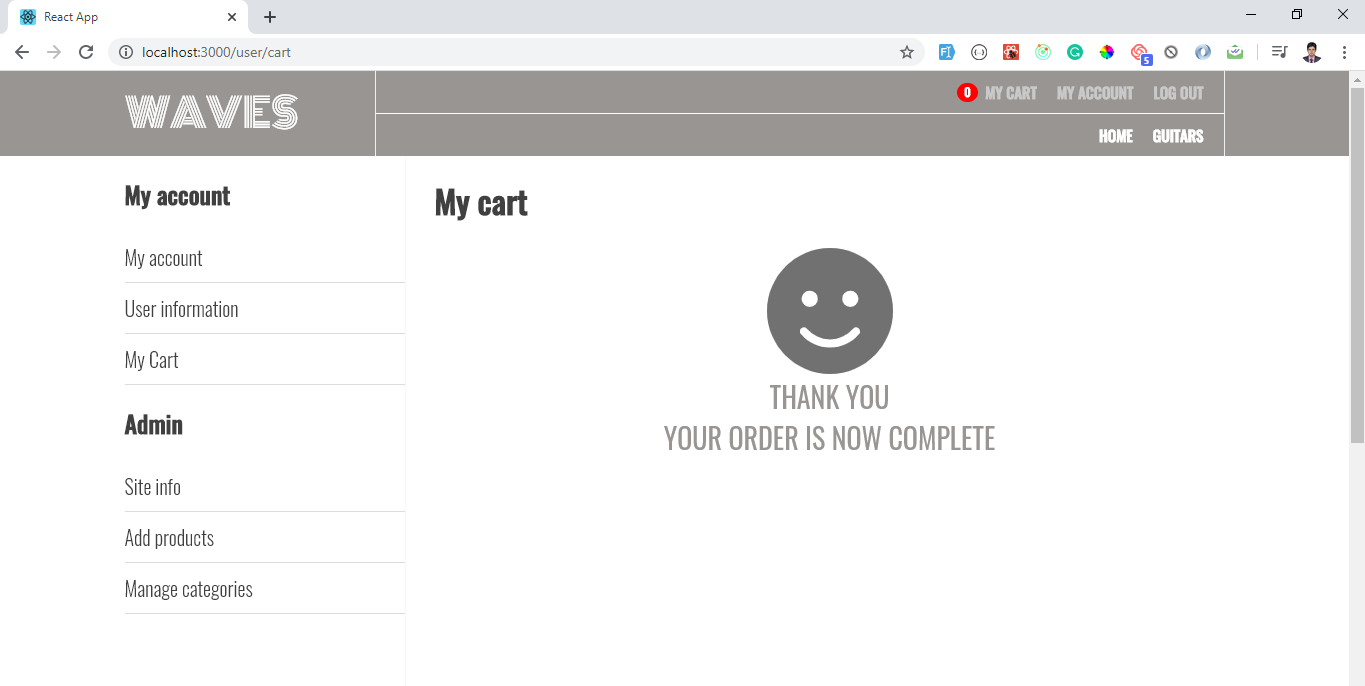
          });

        }

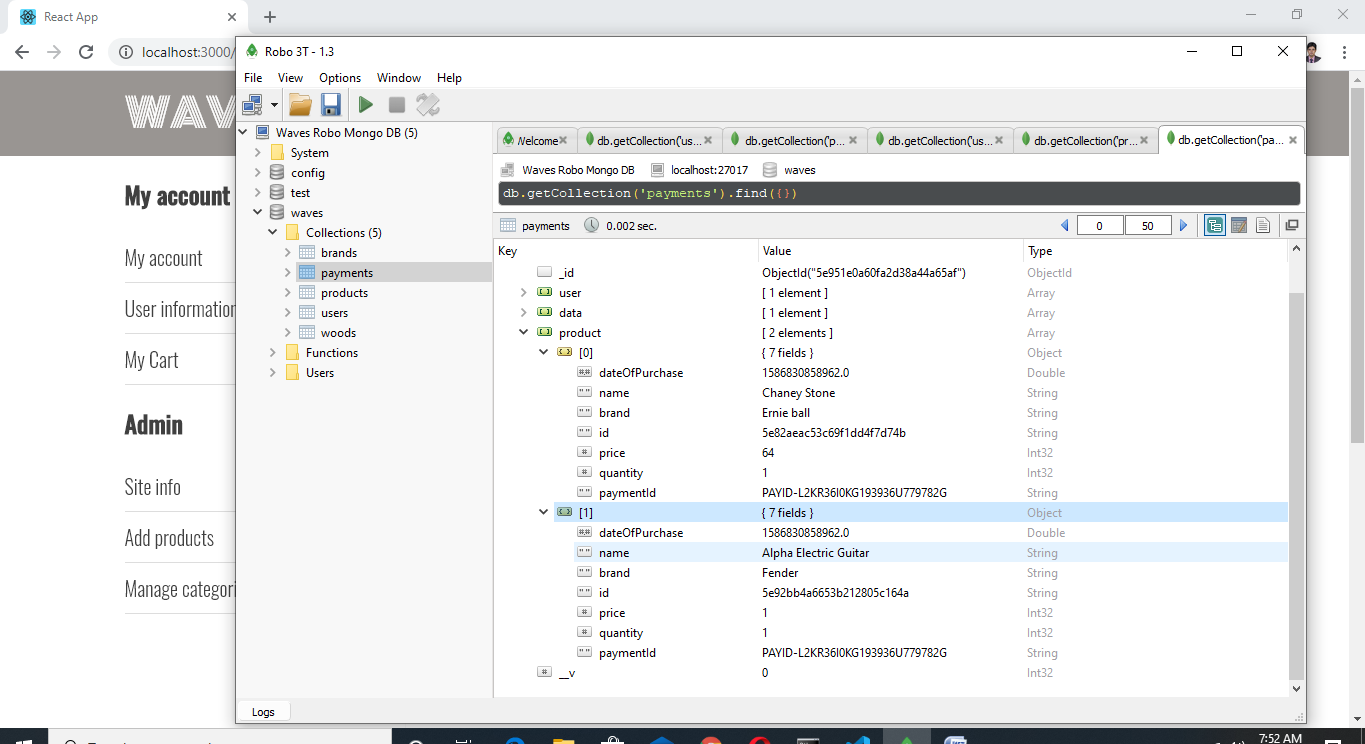
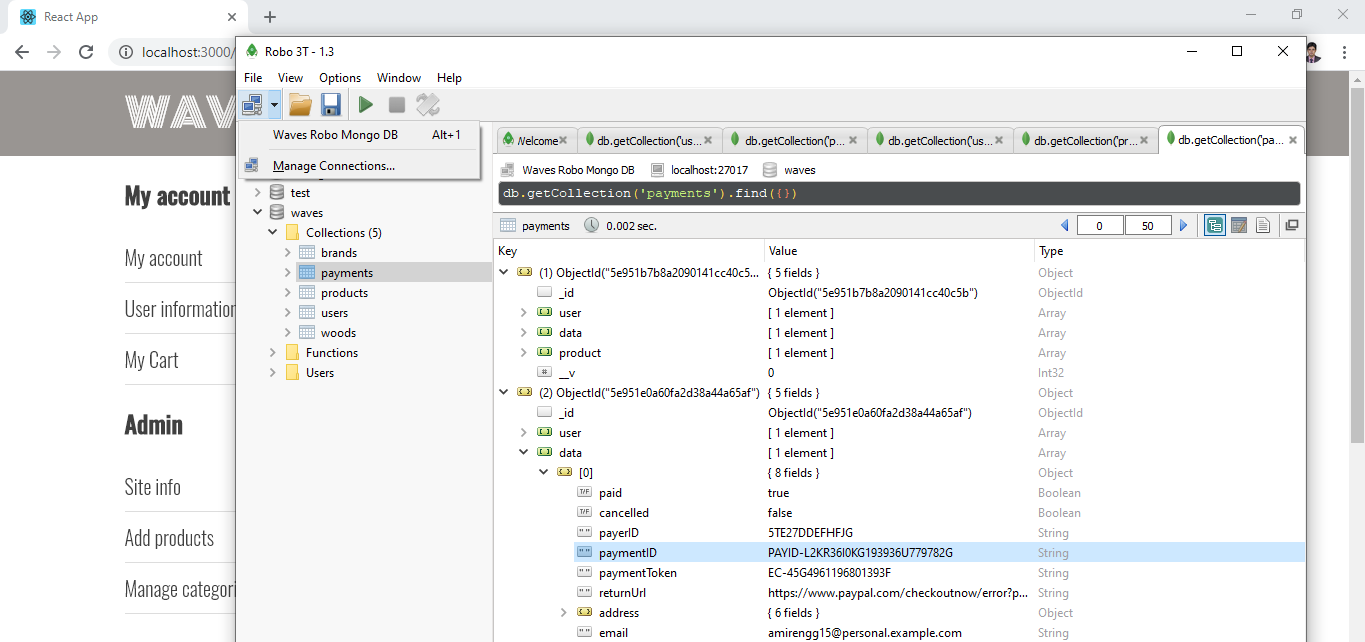
      });

  };

* On successful payment –enters payment collection, in users collection – set cart – empty, history – new data, products collection increase sold by 1



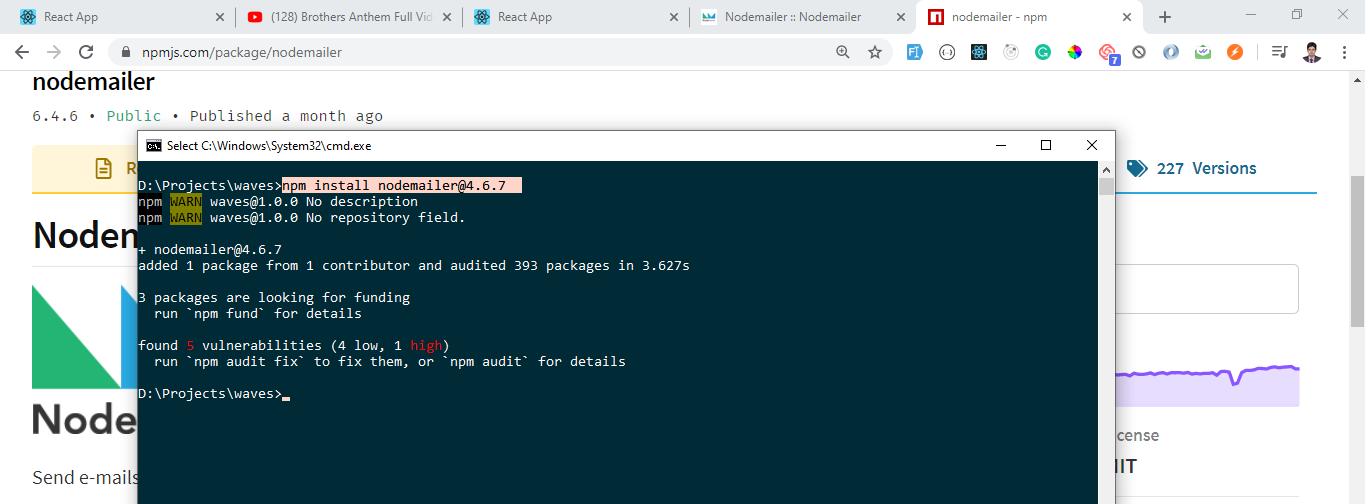
Db – Robo 3T



**NodeMailer**

* This package is used for sending emails with node
* There should be two email id i.e. sender (here waves) and receiver (user)

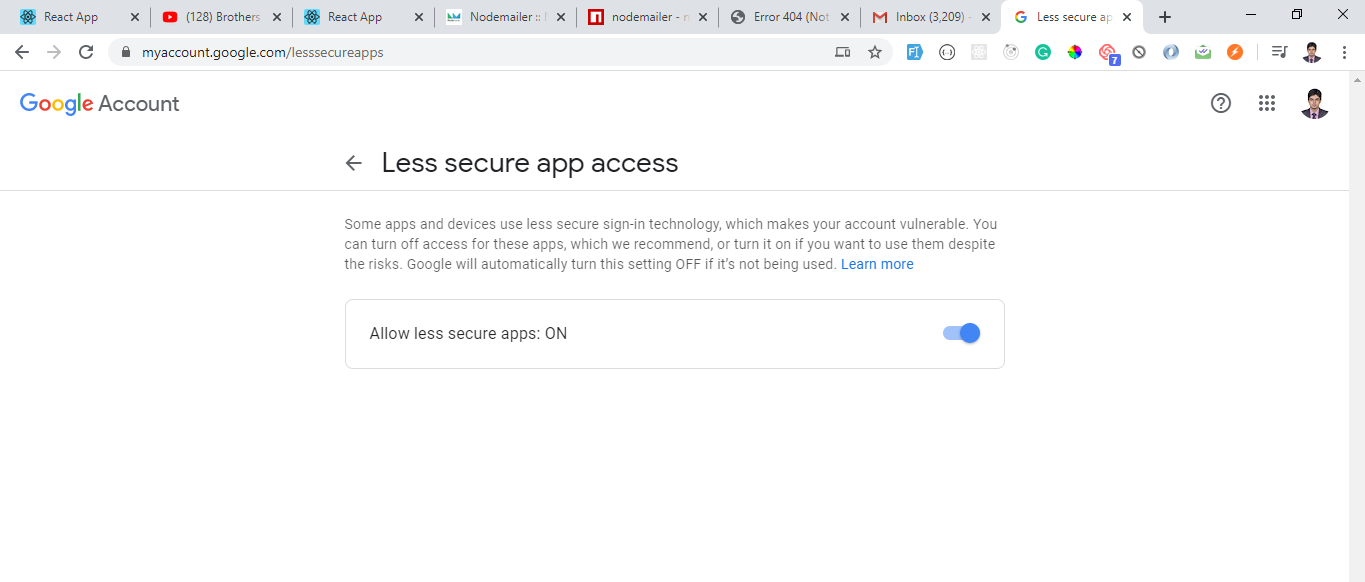
Npm install [nodemailer@4.6.7](mailto:nodemailer@4.6.7)



* Now for the sender email less secure apps should be enabled

Type in google less secure apps enter 🡪 signin with your sender email

Turn allow



* Now in your Node code:

const mailer = require("nodemailer");

const smtpTransport = mailer.createTransport({

  service: "Gmail",

  auth: {

    user: "info.waves.dev@gmail.com",

    pass: "Waves@713301",

  },

});

let mail = {

  from: "Waves <info.waves.dev@gmail.com>",

  to: "amirengg15@gmail.com",

  subject: "Send test mail",

  text: "Testing our waves mail",

  html: "<b>Welcome to Waves</b>",

};

smtpTransport.sendMail(mail, (error, response) => {

  if (error) {

    console.log(error);

  } else {

    console.log("email sent");

  }

  smtpTransport.close();

});

Welcome mail functionality – nodemailer

Server.js – node server

Path – server/server.js

// Utils

const { sendEmail } = require("./utils/mail/index");

app.post("/api/users/register", (req, res) => {

  const user = new User(req.body);

  user.save((err, doc) => {

    if (err) return res.json({ success: false, err });

    sendEmail(doc.email, doc.name, null, "welcome");

    return res.status(200).json({

      success: true,

    });

  });

});

Server/utils/mail/index.js

const mailer = require("nodemailer");

require("dotenv").config();

const { welcome } = require("./welcome\_template.js");

const getEmailData = (to, name, token, template) => {

  let data = null;

  switch (template) {

    case "welcome":

      data = {

        from: "Waves <info.waves.dev@gmail.com>",

        to,

        subject: `Welcome to waves ${name}`,

        html: welcome(),

      };

      break;

    default:

      data;

  }

  return data;

};

const sendEmail = (to, name, token, type) => {

  const smtpTransport = mailer.createTransport({

    service: "Gmail",

    auth: {

      user: "info.waves.dev@gmail.com",

      pass: process.env.EMAIL\_PASS,

    },

  });

  const mail = getEmailData(to, name, token, type);

  smtpTransport.sendMail(mail, (error, response) => {

    if (error) {

      console.log(error);

    } else {

      console.log("email sent");

    }

    smtpTransport.close();

  });

};

module.exports = { sendEmail };

welcome\_template.js

path – server/utils/mail/welcome\_template.js

const welcome = () => {

  return `

    <!DOCTYPE html>

    <html style="margin: 0; padding: 0;">

        <head>

            <title>One | Email template!</title>

        </head>

            <body style="margin: 0; padding: 0;">

                <table class="table" cellpadding="0" cellspacing="0" style="background-color: #eee; empty-cells: hide; margin: 0 auto; padding: 0; width: 600px;">

                    <tr>

                        <td style="background-color: #999592; margin: 0 auto;">

                            <h1 style="box-sizing: border-box; color: white; font-family: Helvetica, Arial, sans-serif; letter-spacing: 0.5px; line-height: 1.4; margin: 0; padding: 15px 25px; text-align: center; text-transform: uppercase;">Welcome to waves</h1></td>

                    </tr>

                    <tr>

                        <td style="margin: 0 auto;">

                            <a href="/" style="box-sizing: border-box; color: #999592 !important; font-family: Arial, Helvetica, sans-serif; line-height: 1.4; margin: 0; text-decoration: none;"><img class="full-width" src="https://media.giphy.com/media/RIm19GefKk3kY/giphy.gif" style="vertical-align: sub; width: 100%;"></a>

                        </td>

                    </tr>

                    <tr>

                        <td style="background-color: #999592; margin: 0 auto;">

                            <p style="box-sizing: border-box; color: white; font-family: Helvetica, Arial, sans-serif; letter-spacing: 0.5px; line-height: 1.4; margin: 0; padding: 15px 25px; text-align: center; text-transform: uppercase;font-size:10px">

                            Waves is a leading retailer of musical instruments, lessons, repairs

                            and rentals in the U.S. With nearly 300 stores across the U.S. and

                            one of the top direct sales websites in the industry, Waves has

                            helped people make music for more than 50 years. Waves is all about enabling musicians and

                            non-musicians alike to experience the almost indescribable joy that

                            comes from playing an instrument.

                            </p></td>

                    </tr>

                </table>

            </body>

      </html>

    `;

}

module.exports = { welcome };

Post Order

* After successful payment will save post order id in transaction collection and user’s history collection

For this we have used crypto-js package

const SHA1 = require("crypto-js/SHA1");

// Post order – code for post order

const date = new Date();

const po = `PO-${date.getSeconds()}${date.getMilliseconds()}-${SHA1(

  "2423423421"

)

  .toString()

  .substring(0, 8)}`;

console.log(po);

// Will save post order after successful payment on successBuy route

app.post("/api/users/successBuy", auth, (req, res) => {

  let history = [];

  let transactionData = {};

  // Post order

  const date = new Date();

  const po = `PO-${date.getSeconds()}${date.getMilliseconds()}-${SHA1(

    req.user.\_id

  )

    .toString()

    .substring(0, 8)}`;

  // user history

  req.body.cartDetail.forEach((item) => {

    history.push({

      porder: po,

      dateOfPurchase: Date.now(),

      name: item.name,

      brand: item.brand.name,

      id: item.\_id,

      price: item.price,

      quantity: item.quantity,

      paymentId: req.body.paymentData.paymentID,

    });

  });

  // PAYMENTS DASHBOARD

 …

…

  transactionData.data = {

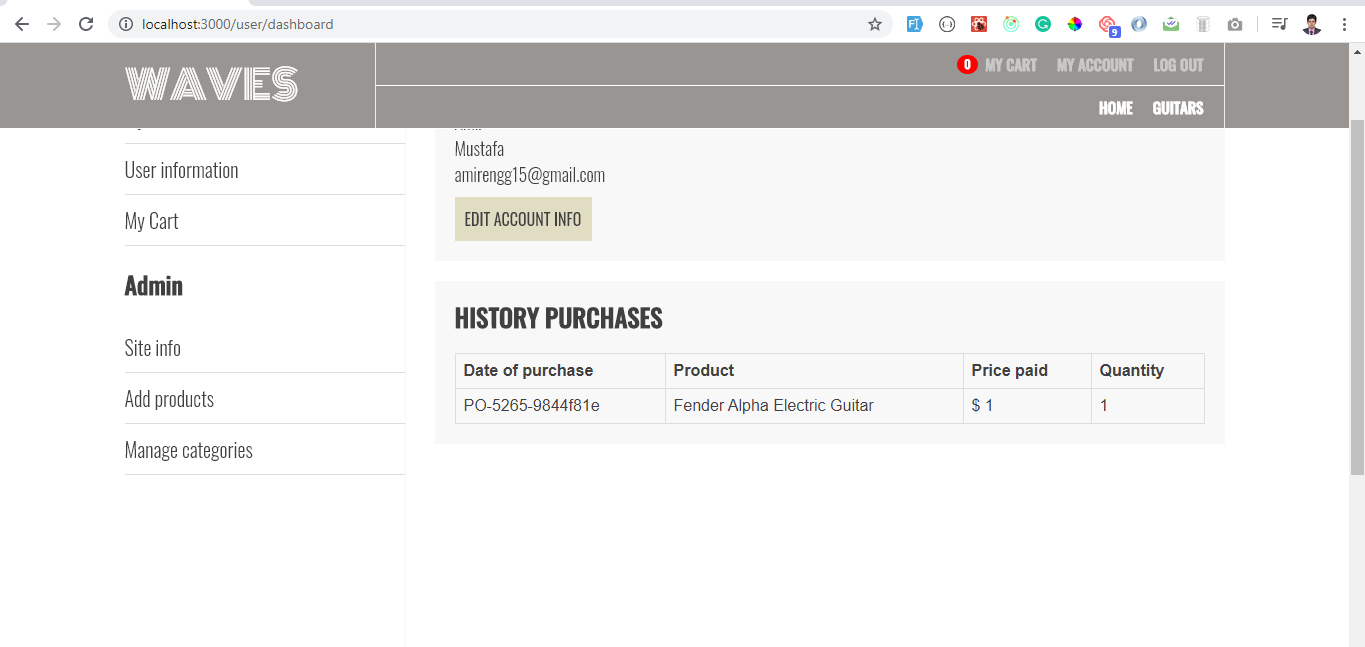
    ...req.body.paymentData,

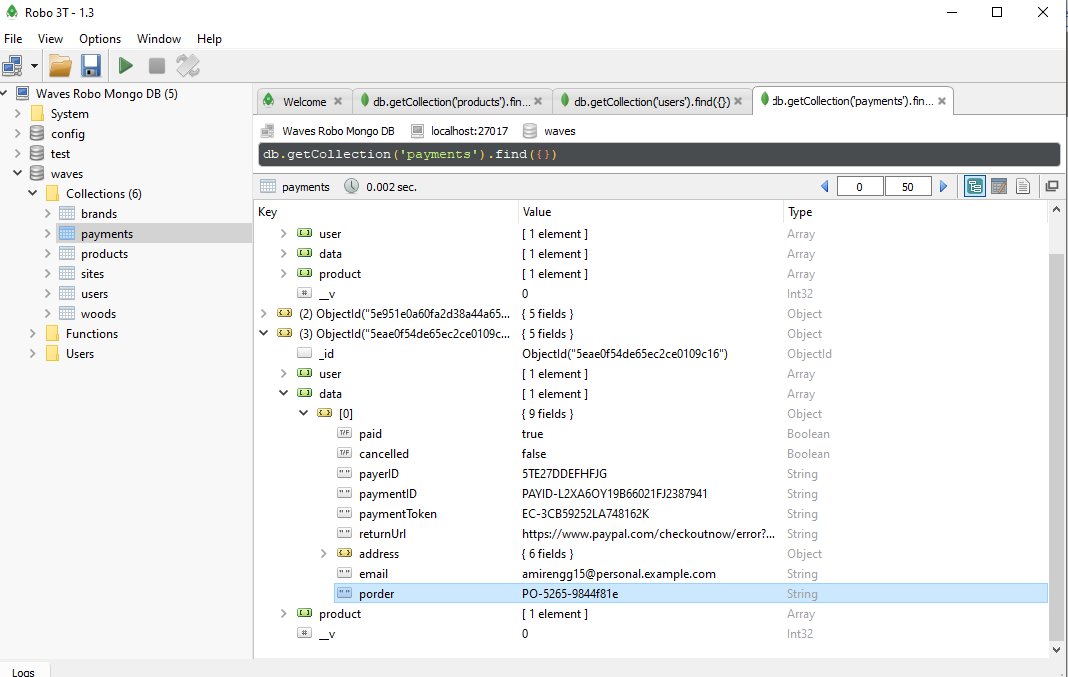
    porder: po,

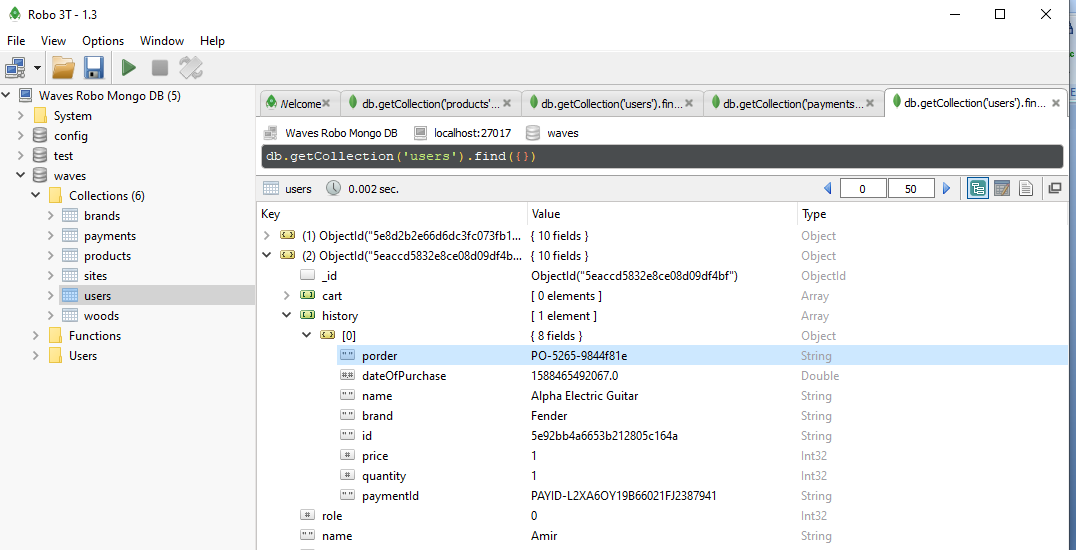
  };

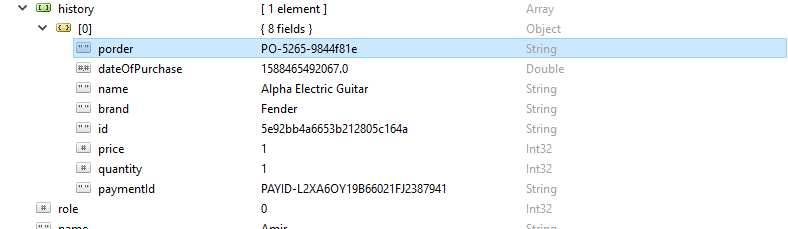
  transactionData.product = history;

});









// Send email on successful buy – same as above code just added sendmail

Server.js

app.post("/api/users/successBuy", auth, (req, res) => {

async.eachSeries(

          products,

          (item, callback) => {

            Product.update(

              { \_id: item.id },

              {

                $inc: {

                  sold: item.quantity,

                },

              },

              { new: false },

              callback

            );

          },

          (err) => {

            if (err) return res.json({ success: false, err });

            // send mail for purchase order

            sendEmail(user.email, user.name, null, "purchase", transactionData);

            res.status(200).json({

              success: true,

              cart: user.cart,

              cartDetail: [],

            });

          }

        );

Server/utils/mail/index.js

const mailer = require("nodemailer");

require("dotenv").config();

const { welcome } = require("./welcome\_template.js");

const { purchase } = require("./purchase\_template.js");

const getEmailData = (to, name, token, template, actionData) => {

  let data = null;

  switch (template) {

    case "welcome":

      data = {

        from: "Waves <info.waves.dev@gmail.com>",

        to,

        subject: `Welcome to waves ${name}`,

        html: welcome(),

      };

      break;

    case "purchase":

      data = {

        from: "Waves <info.waves.dev@gmail.com>",

        to,

        subject: `Thanks for shopping with us ${name}`,

        html: purchase(actionData),

      };

      break;

    default:

      data;

  }

  return data;

};

const sendEmail = (to, name, token, type, actionData = null) => {

  const smtpTransport = mailer.createTransport({

    service: "Gmail",

    auth: {

      user: "info.waves.dev@gmail.com",

      pass: process.env.EMAIL\_PASS,

    },

  });

  const mail = getEmailData(to, name, token, type, actionData);

  smtpTransport.sendMail(mail, (error, response) => {

    if (error) {

      console.log(error);

    } else {

      console.log("email sent");

    }

    smtpTransport.close();

  });

};

module.exports = { sendEmail };

server/mail/purchase\_template.js

const purchase = (data) => {

  const getItems = () => {

    let template = "";

    data.product.forEach((item) => {

      template += `

        <div style="font-family: Helvetica, Arial, sans-serif; letter-spacing: 0.5px; line-height: 1.4; margin: 0; padding: 15px 25px ; text-transform: uppercase;">

                <h3>

                    ${item.brand} ${item.name}

                </h3>

                <p>Price paid: $ ${item.price}</p>

                <p>Purchase order: ${item.porder}</p>

        </div>

        `;

    });

    return template;

  };

  return `

    <!DOCTYPE html>

    <html style="margin: 0; padding: 0;">

        <head>

            <title>One | Email template!</title>

        </head>

            <body style="margin: 0; padding: 0;">

                <table class="table" cellpadding="0" cellspacing="0" style="background-color: #eee; empty-cells: hide; margin: 0 auto; padding: 0; width: 600px;">

                    <tr>

                        <td style="background-color: #999592; margin: 0 auto;">

                            <h1 style="box-sizing: border-box; color: white; font-family: Helvetica, Arial, sans-serif; letter-spacing: 0.5px; line-height: 1.4; margin: 0; padding: 15px 25px; text-align: center; text-transform: uppercase;">Thank you for buying</h1></td>

                    </tr>

                    <tr>

                        <td style="margin: 0 auto;">

                             <h2 style="box-sizing: border-box; color: #000000; font-family: Helvetica, Arial, sans-serif; letter-spacing: 0.5px; line-height: 1.4; margin: 0; padding: 15px 25px; text-align: center; text-transform: uppercase;">Your purchase information</h2>

                             ${getItems()}

                        </td>

                    </tr>

                    <tr>

                         <td style="background-color: #999592; margin: 0 auto;">

                                 <p style="box-sizing: border-box; color: white; font-family: Helvetica, Arial, sans-serif; letter-spacing: 0.5px; line-height: 1.4; margin: 0; padding: 15px 25px; text-align: center; text-transform: uppercase;font-size:10px">

                                 Waves is a leading retailer of musical instruments, lessons, repairs

                                 and rentals in the U.S. With nearly 300 stores across the U.S. and

                                 one of the top direct sales websites in the industry, Waves has

                                 helped people make music for more than 50 years. Waves is all about enabling musicians and non-musicians alike to experience the almost indescribable joy that

                                 comes from playing an instrument.

                                 </p></td>

                    </tr>

                </table>

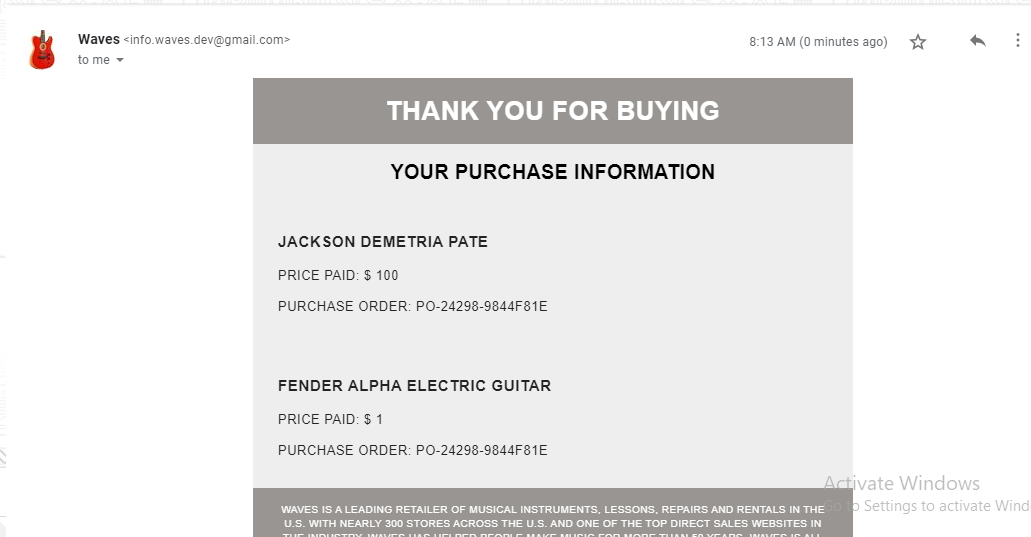
            </body>

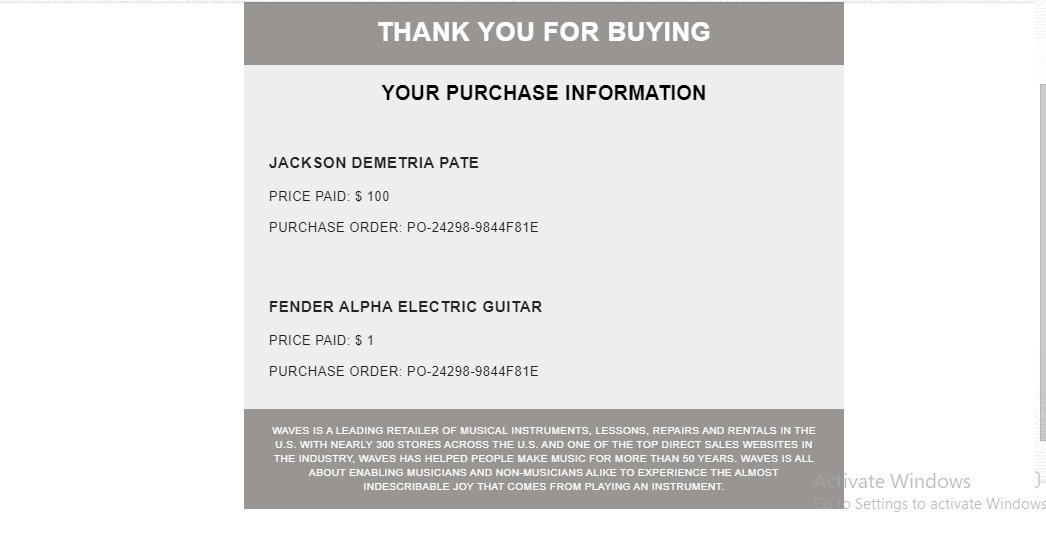
      </html>

    `;

};

module.exports = { purchase };





Multer – File upload in Node JS

Client – React JS

Here we have used dropzone to pick up data and sent to node server to upload via multer package

add\_file.js

path - client\src\components\User\Admin\add\_file.js

import React, { Component } from "react";

import UserLayout from "../../../hoc/user";

import DropZone from "react-dropzone";

import axios from "axios";

import FontAwesomeIcon from "@fortawesome/react-fontawesome";

import faPlusCircle from "@fortawesome/fontawesome-free-solid/faPlusCircle";

import CircularProgress from "@material-ui/core/CircularProgress";

import { Link } from "react-router-dom";

class AddFile extends Component {

  constructor() {

    super();

    this.state = {

      formSuccess: false,

      formError: false,

      uploading: false,

      files: [],

    };

  }

  onDrop = (files) => { // Receive here from drop zone and send to node via axios

    this.setState({ uploading: true });

    let formData = new FormData();

    const config = {

      header: { "content-type": "multipart/form-data" },

    };

    formData.append("file", files[0]); // this name is used in node server for accepting image -  (line 94 i.e. multer({ storage: storage }).single("file");)

    axios.post("/api/users/uploadfile", formData, config).then((response) => {

      if (response.data.success) {

        this.setState(

          {

            formSuccess: true,

            formError: false,

            uploading: false,

          },

          () => {

            setTimeout(() => {

              this.setState({ formSuccess: false });

            }, 2000);

          }

        );

      }

    });

  };

  showFileList = () =>

    this.state.files

      ? this.state.files.map((item, i) => (

          <li key={i}> // download all files

            <Link to={"/api/users/download/" + item} target="\_blank">

              {item}

            </Link>

          </li>

        ))

      : null;

  componentDidMount() { //

    axios.get("/api/users/admin\_files").then((response) => { // get all files

this.setState({ files: response.data });

    });

  }

  render() {

    return (

      <UserLayout>

        <h1>Upload file - Multer</h1>

        <div>

          <DropZone

            onDrop={(e) => this.onDrop(e)}

            multiple={false}

            className="dropzone\_box"

          >

            <div className="wrap">

              <FontAwesomeIcon icon={faPlusCircle} />

            </div>

          </DropZone>

          {this.state.uploading ? (

            <div

              className="dropzone\_box"

              style={{

                textAlign: "center",

                paddingTop: "60px",

              }}

            >

              <CircularProgress style={{ color: "#00bcd4" }} thickness={7} />

            </div>

          ) : null}

          <div style={{ clear: "both" }}>

            {this.state.formSuccess ? (

              <div className="form\_success">Success</div>

            ) : null}

            {this.state.formError ? (

              <div className="error\_label">Please check your data</div>

            ) : (

              ""

            )}

          </div>

          <hr />

          <div>

            <ul>{this.showFileList()}</ul>

          </div>

        </div>

      </UserLayout>

    );

  }

}

export default AddFile;

Create a directory uploads in root directory of project

Server – node js

//=================================

//         MULTER - File Upload

//=================================

const multer = require("multer");

let storage = multer.diskStorage({

  // basic config for multer

  destination: (req, file, cb) => {

    cb(null, "uploads/");

  },

  filename: (req, file, cb) => {

    cb(null, `${Date.now()}\_${file.originalname}`);

  },

  // fileFilter:(req,file,cb)=>{

  //     const ext = path.extname(file.originalname)

  //     if(ext !== '.jpg' && ext !== '.png'){

  //         return cb(res.status(400).end('only jpg, png is allowed'),false);

  //     }

  //     cb(null,true)

  // }

});

const upload = multer({ storage: storage }).single("file"); // single = single file upload - file name received from drop zone - client - add\_file.js dropzone

app.post("/api/users/uploadfile", auth, admin, (req, res) => {

  upload(req, res, (err) => {

    if (err) {

      return res.json({ success: false, err });

    }

    return res.json({ success: true });

  });

});

// Reading files uploaded in directory

const fs = require("fs");

const path = require("path");

app.get("/api/users/admin\_files", auth, admin, (req, res) => {

  const dir = path.resolve(".") + "/uploads/";

  fs.readdir(dir, (err, items) => {

    return res.status(200).send(items);

  });

});

// Download file

app.get("/api/users/download/:id", auth, admin, (req, res) => {

  const file = path.resolve(".") + `/uploads/${req.params.id}`;

  res.download(file);

});

RESET YOUR PASSWORD

**Client – React**

Routes.js

import ResetUser from "./components/Reset\_user";

<Route path="/reset\_user" exact component={Auth(ResetUser, false)} />

Reset\_user.js

import React, { Component } from "react";

import axios from "axios";

import FormField from "../utils/Form/formfield";

import { update, generateData, isFormValid } from "../utils/Form/formActions";

class ResetUser extends Component {

  state = {

    formError: false,

    formSuccess: false,

    formdata: {

      email: {

        element: "input",

        value: "",

        config: {

          name: "email\_input",

          type: "email",

          placeholder: "Enter your email",

        },

        validation: {

          required: true,

          email: true,

        },

        valid: false,

        touched: false,

        validationMessage: "",

      },

    },

  };

  updateForm = (element) => {

    const newFormdata = update(element, this.state.formdata, "reset\_email");

    this.setState({

      formError: false,

      formdata: newFormdata,

    });

  };

  submitForm = (event) => {

    event.preventDefault();

    let dataToSubmit = generateData(this.state.formdata, "reset\_email");

    let formIsValid = isFormValid(this.state.formdata, "reset\_email");

    if (formIsValid) {

      axios.post("/api/users/reset\_user", dataToSubmit).then((response) => {

        if (response.data.success) {

          this.setState({

            formSuccess: true,

          });

        }

      });

    } else {

      this.setState({

        formError: true,

      });

    }

  };

  render() {

    return (

      <div className="container">

        <h1>Reset passwords</h1>

        <form onSubmit={(event) => this.submitForm(event)}>

          <FormField

            id={"email"}

            formdata={this.state.formdata.email}

            change={(element) => this.updateForm(element)}

          />

          {this.state.formSuccess ? (

            <div className="form\_success">

              Password reset successful, check your email

            </div>

          ) : null}

          {this.state.formError ? (

            <div className="error\_label">Please check your data</div>

          ) : null}

          <button onClick={(event) => this.submitForm(event)}>

            Send email to reset password

          </button>

        </form>

      </div>

    );

  }

}

export default ResetUser;

Server – Node - server.js

app.post("/api/users/reset\_user", (req, res) => {

  User.findOne({ email: req.body.email }, (err, user) => {

    user.generateResetToken((err, user) => {

      if (err) return res.json({ success: false, err });

      sendEmail(user.email, user.name, null, "reset\_password", user);

      return res.json({ success: true });

    });

  });

});

Server – Node – server/models/user.js

const mongoose = require("mongoose");

const bcrypt = require("bcrypt");

const jwt = require("jsonwebtoken");

const crypto = require("crypto");

const moment = require("moment");

const SALT\_I = 10;

require("dotenv").config();

const userSchema = mongoose.Schema({

  email: {

    type: String,

    required: true,

    trim: true,

    unique: 1,

  },

  password: {

    type: String,

    required: true,

    minlength: 5,

  },

  name: {

    type: String,

    required: true,

    maxlength: 100,

  },

  lastname: {

    type: String,

    required: true,

    maxlength: 100,

  },

  cart: {

    type: Array,

    default: [],

  },

  history: {

    type: Array,

    default: [],

  },

  role: {

    type: Number,

    default: 0,

  },

  token: {

    type: String,

  },

  resetToken: {

    type: String,

  },

  // token expiry

  resetTokenExp: {

    type: Number,

  },

});

userSchema.pre("save", function (next) {

  var user = this;

  if (user.isModified("password")) {

    bcrypt.genSalt(SALT\_I, function (err, salt) {

      if (err) return next(err);

      bcrypt.hash(user.password, salt, function (err, hash) {

        if (err) return next(err);

        user.password = hash;

        next();

      });

    });

  } else {

    next();

  }

});

// Method for comparing password

userSchema.methods.comparePassword = function (candidatePassword, cb) {

  bcrypt.compare(candidatePassword, this.password, function (err, isMatch) {

    if (err) return cb(err);

    cb(null, isMatch);

  });

};

userSchema.methods.generateResetToken = function (cb) {

  var user = this;

  crypto.randomBytes(20, function (err, buffer) {

    var token = buffer.toString("hex");

    var today = moment().startOf("day").valueOf();

    var tomorrow = moment(today).endOf("day").valueOf();

    user.resetToken = token;

    user.resetTokenExp = tomorrow;

    user.save(function (err, user) {

      if (err) return cb(err);

      cb(null, user);

    });

  });

};

// generate token

userSchema.methods.generateToken = function (cb) {

  var user = this;

  // strong token = userid + env secret password

  var token = jwt.sign(user.\_id.toHexString(), process.env.SECRET);

  user.token = token;

  user.save(function (err, user) {

    if (err) return cb(err);

    cb(null, user);

  });

};

userSchema.statics.findByToken = function (token, cb) {

  var user = this;

  // decode has \_id of mongodb

  jwt.verify(token, process.env.SECRET, function (err, decode) {

    user.findOne({ \_id: decode, token: token }, function (err, user) {

      if (err) return cb(err);

      cb(null, user);

    });

  });

};

const User = mongoose.model("User", userSchema);

module.exports = { User };

Create mail for sending

Server/utils/mail/index.js

const mailer = require("nodemailer");

require("dotenv").config();

const { welcome } = require("./welcome\_template.js");

const { purchase } = require("./purchase\_template.js");

const { resetPass } = require("./reset\_password.js");

…

…

case "reset\_password":

      data = {

        from: "Waves <info.waves.dev@gmail.com>",

        to,

        subject: `Hey ${name}, reset your password here`,

        html: resetPass(actionData),

};

Server/utils/mail/reset\_password.js

require("dotenv").config();

const resetPass = (data) => {

  const URL =

    process.env.NODE\_ENV === "production"

      ? process.env.ROOT\_URL

      : "http://localhost:3000";

  return `<!DOCTYPE html>

    <html style="margin: 0; padding: 0;">

      <head>

        <title>One | Email template!</title>

      </head>

      <body style="margin: 0; padding: 0;">

        <table

          class="table"

          cellpadding="0"

          cellspacing="0"

          style="

            background-color: #eee;

            empty-cells: hide;

            margin: 0 auto;

            padding: 0;

            width: 600px;

          "

        >

          <tr>

            <td style="background-color: #999592; margin: 0 auto;">

              <h1

                style="

                  box-sizing: border-box;

                  color: white;

                  font-family: Helvetica, Arial, sans-serif;

                  letter-spacing: 0.5px;

                  line-height: 1.4;

                  margin: 0;

                  padding: 15px 25px;

                  text-align: center;

                  text-transform: uppercase;

                "

              >

                Waves

              </h1>

            </td>

          </tr>

          <tr>

            <td style="margin: 0 auto; padding: 15px 25px; box-sizing: border-box;">

              <p>Click on this link to reset your password:</p>

              <a href="${URL}/reset\_password/${data.resetToken}">Reset your password</a>

            </td>

          </tr>

          <tr>

            <td style="background-color: #999592; margin: 0 auto;">

              <p

                style="

                  box-sizing: border-box;

                  color: white;

                  font-family: Helvetica, Arial, sans-serif;

                  letter-spacing: 0.5px;

                  line-height: 1.4;

                  margin: 0;

                  padding: 15px 25px;

                  text-align: center;

                  text-transform: uppercase;

                  font-size: 10px;

                "

              >

                Waves is a leading retailer of musical instruments, lessons, repairs

                and rentals in the U.S. With nearly 300 stores across the U.S. and

                one of the top direct sales websites in the industry, Waves has

                helped people make music for more than 50 years. Waves is all about

                enabling musicians and non-musicians alike to experience the almost

                indescribable joy that comes from playing an instrument.

              </p>

            </td>

          </tr>

        </table>

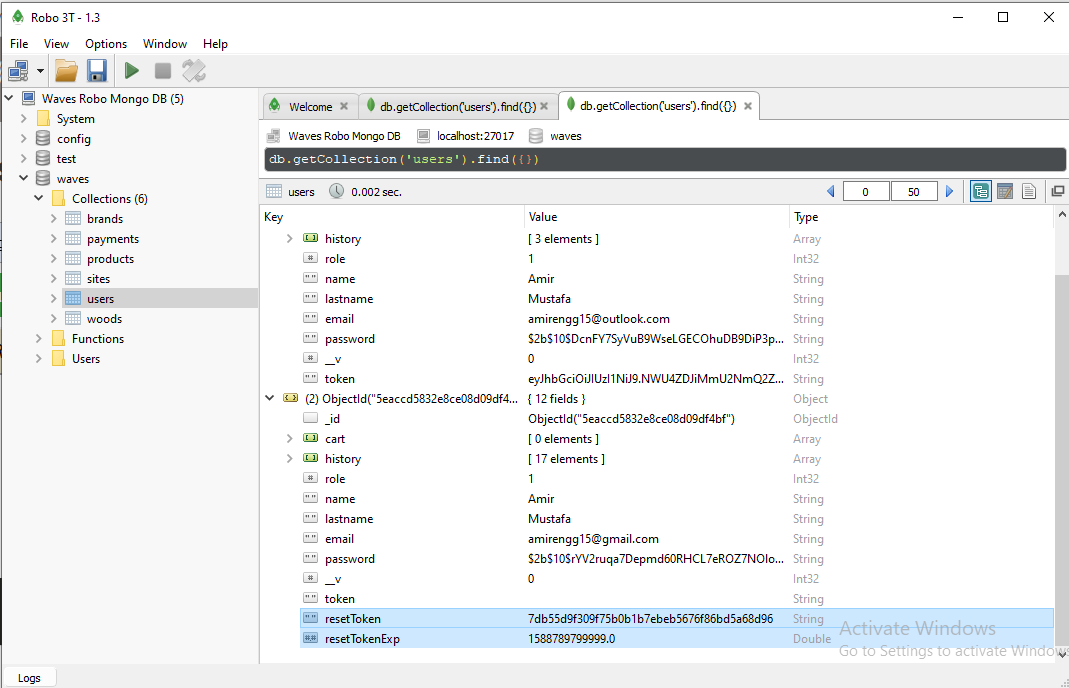
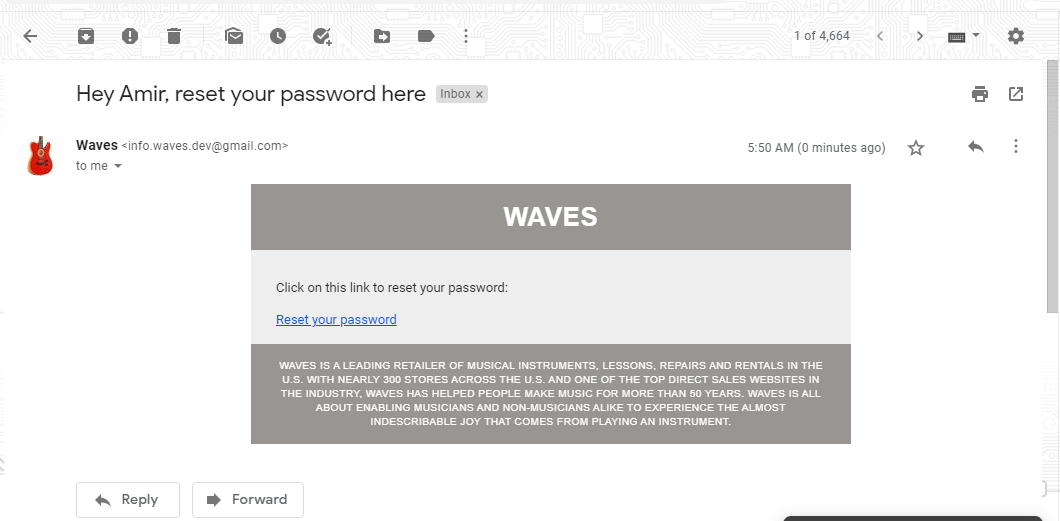
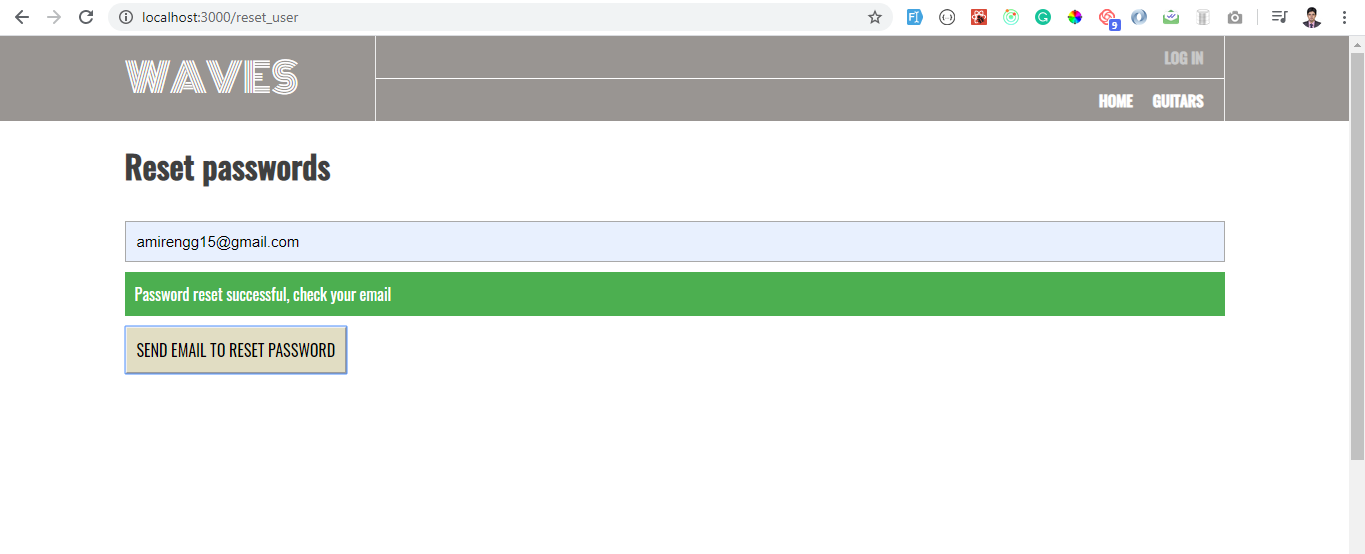
      </body>

    </html>

    `;

};

module.exports = { resetPass };



Server.js

app.post("/api/users/reset\_password", (req, res) => {

  var today = moment().startOf("day").valueOf();

  User.findOne(

    {

      resetToken: req.body.resetToken,

      resetTokenExp: {

        $gte: today,

      },

    },

    (err, user) => {

      if (!user)

        return res.json({

          success: false,

          message: "Sorry, token bad, generate a new one.",

        });

 // goes to the userSchema.pre("save", function (next) {} of user model and update in db

      user.password = req.body.password;

      user.resetToken = "";

      user.resetTokenExp = "";

      user.save((err, doc) => {

        if (err) return res.json({ success: false, err });

        return res.status(200).json({

          success: true,

        });

      });

    }

  );

});

Client – added route

import ResetPass from "./components/Reset\_user/reset\_pass";

<Route

          path="/reset\_password/:token"

          exact

          component={Auth(ResetPass, false)}

        />

Client – reset\_password.js – client\src\components\Reset\_user\reset\_pass.js

import React, { Component } from "react";

import axios from "axios";

import FormField from "../utils/Form/formfield";

import { update, generateData, isFormValid } from "../utils/Form/formActions";

import Dialog from "@material-ui/core/Dialog";

class ResetPass extends Component {

  state = {

    resetToken: "",

    formError: false,

    formErrorMessage: "",

    formSuccess: "",

    formdata: {

      password: {

        element: "input",

        value: "",

        config: {

          name: "password\_input",

          type: "password",

          placeholder: "Enter your password",

        },

        validation: {

          required: true,

        },

        valid: false,

        touched: false,

        validationMessage: "",

      },

      confirmPassword: {

        element: "input",

        value: "",

        config: {

          name: "confirm\_password\_input",

          type: "password",

          placeholder: "Confirm your password",

        },

        validation: {

          required: true,

          confirm: "password",

        },

        valid: false,

        touched: false,

        validationMessage: "",

      },

    },

  };

  updateForm = (element) => {

    const newFormdata = update(element, this.state.formdata, "reset\_pass");

    this.setState({

      formError: false,

      formdata: newFormdata,

    });

  };

  submitForm = (event) => {

    event.preventDefault();

    let dataToSubmit = generateData(this.state.formdata, "reset\_pass");

    let formIsValid = isFormValid(this.state.formdata, "reset\_pass");

    if (formIsValid) {

      axios

        .post("/api/users/reset\_password", {

          ...dataToSubmit,

          resetToken: this.state.resetToken,

        })

        .then((response) => {

          if (!response.data.success) {

            this.setState({

              formError: true,

              formErrorMessage: response.data.message,

            });

          } else {

            this.setState({ formError: false, formSuccess: true });

            setTimeout(() => {

              this.props.history.push("/register\_login");

            }, 3000);

          }

        });

    } else {

      this.setState({

        formError: true,

      });

    }

  };

  componentDidMount() {

    const resetToken = this.props.match.params.token;

    this.setState({ resetToken });

  }

  render() {

    return (

      <div className="container">

        <form onSubmit={(event) => this.submitForm(event)}>

          <h2>Reset password</h2>

          <div>

            <FormField

              id={"password"}

              formdata={this.state.formdata.password}

              change={(element) => this.updateForm(element)}

            />

            <FormField

              id={"confirmPassword"}

              formdata={this.state.formdata.confirmPassword}

              change={(element) => this.updateForm(element)}

            />

          </div>

          <div>

            {this.state.formError ? (

              <div className="error\_label">{this.state.formErrorMessage}</div>

            ) : (

              ""

            )}

            <button onClick={(event) => this.submitForm(event)}>Update</button>

          </div>

        </form>

        <Dialog open={this.state.formSuccess}>

          <div className="dialog\_alert">

            <div>Alright !!</div>

            <div>Your password was reseted... you will be redirected</div>

          </div>

        </Dialog>

      </div>

    );

  }

}

export default ResetPass;

