Kunal Makwana

Team-20

Atmanirbhar Farmer

```
import React from 'react';

import { createAppContainer, createSwitchNavigator } from 'react-navigation';
import { createStackNavigator } from 'react-navigation-stack';
import SplashScreen from './Screen/SplashScreen';
import LoginScreen from './Screen/LoginScreen';
import RegisterScreen from './Screen/RegisterScreen';
import DrawerNavigationRoutes from './Screen/DrawerNavigationRoutes';

const Auth = createStackNavigator({
  LoginScreen: {
    screen: LoginScreen,
    navigationOptions: {
      headerShown: false,
    },
  },
  RegisterScreen: {
    screen: RegisterScreen,
    navigationOptions: {
      title: 'Register',
      headerStyle: {
        backgroundColor: '#384245',
      },
      headerTintColor: '#fff',
    },
  },
});
const App = createSwitchNavigator({
  SplashScreen: {
    screen: SplashScreen,
    navigationOptions: {
      headerShown: false,
    },
  },
  Auth: {
    screen: Auth,
  },
  DrawerNavigationRoutes: {
    screen: DrawerNavigationRoutes,
    navigationOptions: {
      headerShown: false,
    },
  },
});

export default createAppContainer(App);
```

```
const express = require("express");
const router = express.Router();

// Item model
const item = require("../../models/Item");

// @routes  GET api/items
// @desc    GET All Items
// @access  Public
router.get("/", (req, res) => {
  item
    .find()
    .sort({ date: -1 })
    .then(items => res.json(items));
});

// @routes  POST api/items
// @desc    CREATE An Item
// @access  Public
router.post("/", (req, res) => {
  const newItem = new Item({
    name: req.body.name
  });

  newItem.save().then(item => res.json(item));
});

// @routes  POST api/items/:id
// @desc    DELETE An Item
// @access  Public
router.delete("/:id", (req, res) => {
  Item.findById(req.params.id)
    .then(item => item.remove().then(() => res.json({ success: true })))
    .catch(err => res.status(404).json({ success: false, error: err }));
});

module.exports = router;
```

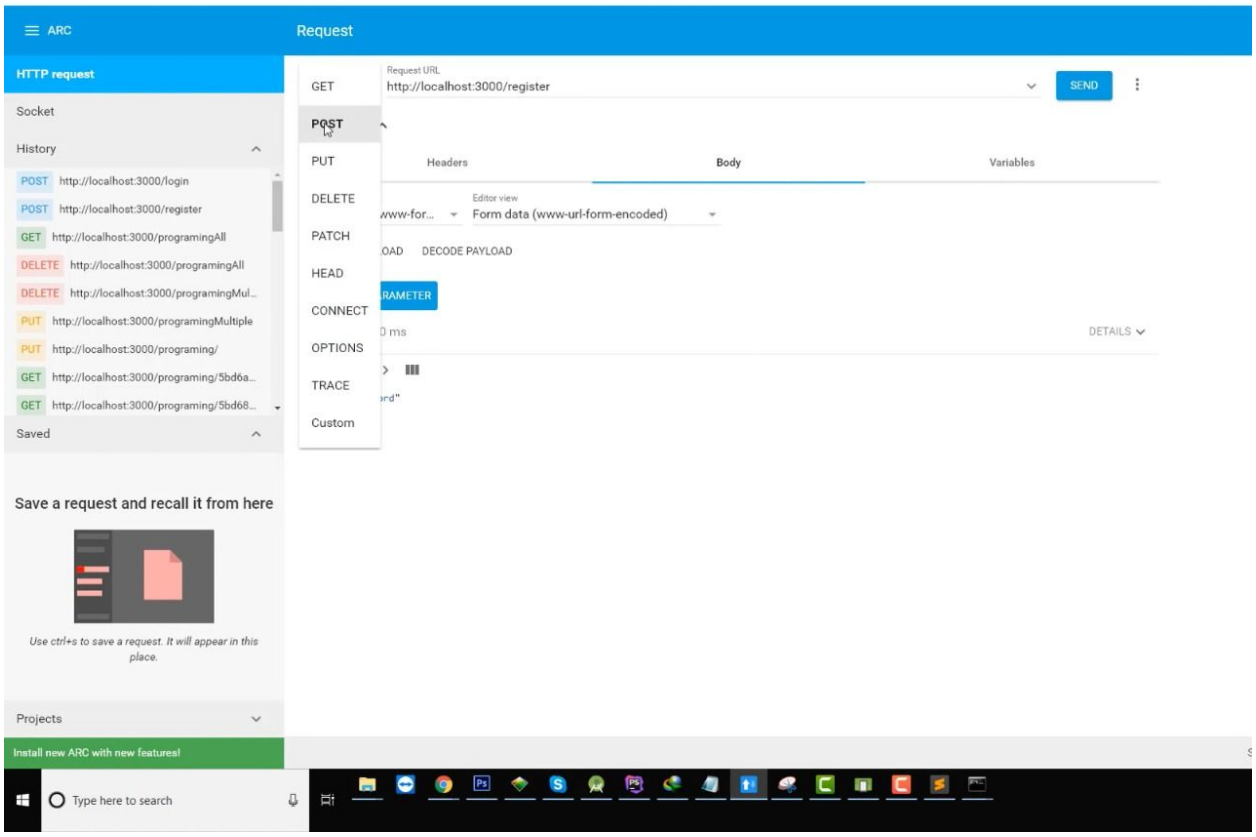
```
//let is a block level variable declaration ;
//const is similar to let but we can modify the object(json) crea
//var is function level variable declaration


require('dotenv').config();
const express = require('express');
const cors = require('cors');
const bodyParser  = require('body-parser');
const db = require('./models')
const handle = require('./handlers')
const routes = require('./routes');
const app = express();
const port = process.env.PORT;
app.use(cors());
app.use(express.json());
app.get('/',(req,res)=>res.json({hello:"world"}));
var tppp = '/api/auth';
app.use(tppp,routes.auth);
app.use('/api/polls',routes.poll);
app.use(handle.notFound);
app.use(handle.errors);
console.log(port);
app.listen(port,console.log(`server on ${port}`));
```

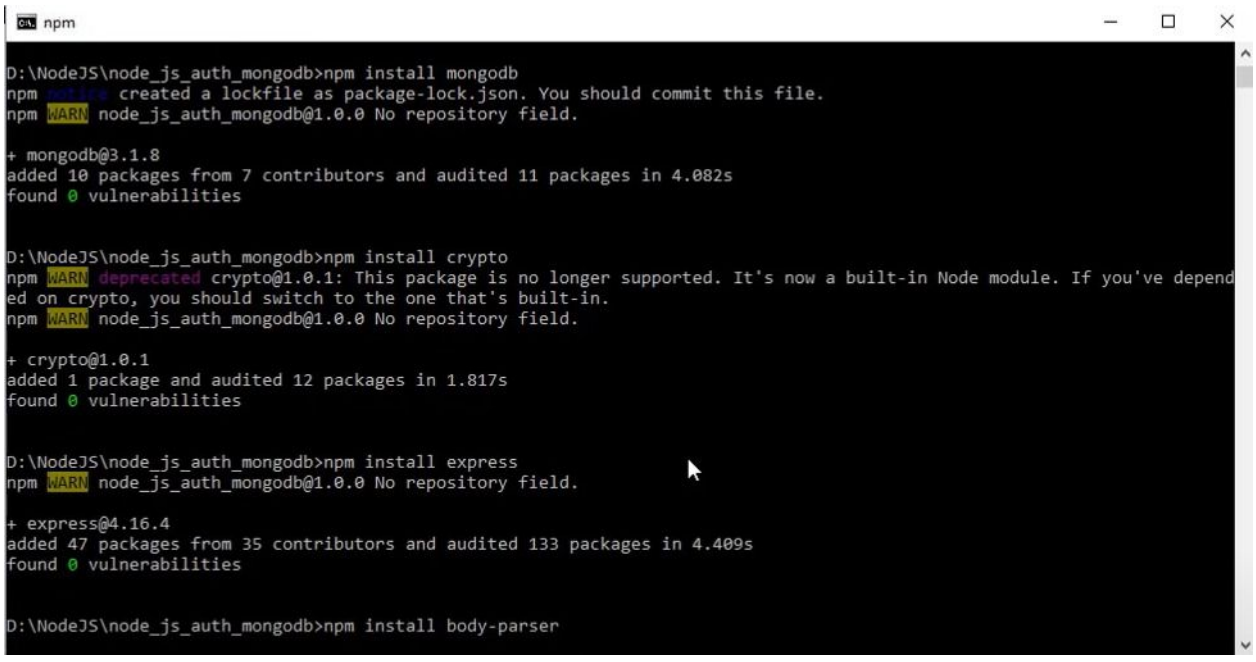
```
const jwt = require("jsonwebtoken");

const db = require("../models");
exports.register = async (req, res, next) => {
  try {
    const user = await db.User.create(req.body);
    const { id, username } = user;
    const token = jwt.sign({ id, username }, process.env.SECRET);
    res.status(201).json({ id, username, token });
  } catch (err) {
    if (err.code === 11000) {
      err.message = "Sorry,that username is already taken";
    }
    next(err);
  }
};
exports.login = async (req, res, next) => {
  try {
    const user = await db.User.findOne({ username: req.body.username });
    const { id, username } = user;

    const valid = await user.comparePassword(req.body.password);
    if (valid) {
      const token = jwt.sign({ id, username }, process.env.SECRET);
      res.json({
        id,
        username,
        token
      });
    } else {
      throw new Error();
    }
  } catch (err) {
    err.message = "Invalid Username/password";
    next(err);
  }
};
```

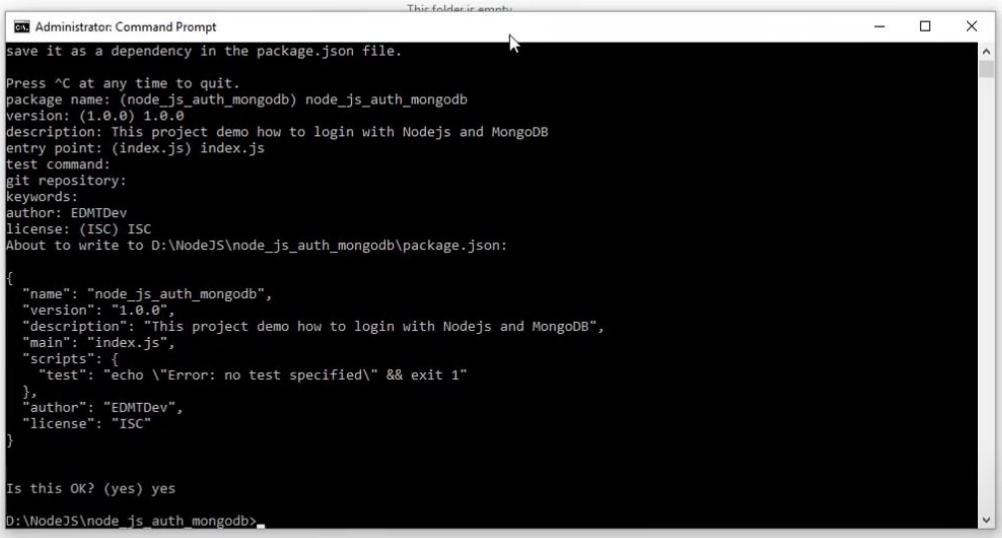
```
module.exports = {
    ...require('./auth'),
    ...require('./poll')
};


module.exports.notFound = (req,res,next)=>{
    const err = new Error('Not found');
    err.status = 404;
    next(err);
};

module.exports.errors = (err,req,res,next)=>{
    res.status(err.status || 400).json({
        message:err.message || 'Something went wrong'
    });
};
```

```
//take the data from models index.js and check the avaibility

const db = require("../models");
exports.showPolls = async (req, res, next) => {
    try {
        const polls = await db.Poll.find().populate("user", ["username", "id"]);
        res.status(200).json(polls);
    } catch (err) {
        err.status = 400;
        next(err);
    }
};
exports.usersPolls = async (req, res, next) => {
    try {
        const { id } = req.decoded;
        const user = await db.User.findById(id).populate("polls");

        res.status(200).json(user.polls);
    } catch (err) {
        err.status = 400;
        //res.json({this:"is bullshit"});
        next(err);
    }
};

exports.createPoll = async (req, res, next) => {
    try {
        console.log(req.decoded);
        const { id } = req.decoded;
        const user = await db.User.findById(id);
        const { question, options } = req.body;
        const poll = await db.Poll.create({
            user,
            question,
            options: options.map(option => ({ option, votes: 0 }))
        });
        user.polls.push(poll._id);
        await user.save();
        res.status(201).json({ ...poll._doc, user: user._id });
    } catch (err) {
        err.status = 400;
        next(err);
    }
};
```

```
exports.getPoll = async (req, res, next) => {
    try {
        const { id } = req.params;
        const poll = await db.Poll.findById(id).populate("user", [
            "username",
            "id"
        ]);
        if (!poll) throw new Error("no poll found");
        res.status(200).json(poll);
    } catch (err) {
        err.status = 400;
        next(err);
    }
};

exports.deletePoll = async (req, res, next) => {
    try {
        const { id: pollId } = req.params;
        const { id: userId } = req.decoded;
        const poll = await db.Poll.findById(pollId);
        if (!poll) throw new Error("no poll found");
        if (poll.user.toString() !== userId) {
            throw new Error("Unauthorised access");
        }
        await poll.remove();
        res.status(202).json(poll);
    } catch (err) {
        err.status = 400;
        next(err);
    }
};
```


ARC
Request
HTTP request
Socket
History
POST http://localhost:3000/login
POST http://localhost:3000/register
GET http://localhost:3000/programingAll
DELETE http://localhost:3000/programingAll
DELETE http://localhost:3000/programingMul...
PUT http://localhost:3000/programingMultiple
PUT http://localhost:3000/programing/
GET http://localhost:3000/programing/5bd6a...
GET http://localhost:3000/programing/5bd68...
Saved
Save a request and recall it from here
Use ctrl+s to save a request. It will appear in this place.
Projects
Install new ARC with new features!
GET
POST
PUT
DELETE
PATCH
HEAD
CONNECT
OPTIONS
TRACE
Custom
Request URL
http://localhost:3000/register
SEND
Headers
Body
Variables
Editor view
Form data (www-url-form-encoded)
DECODE PAYLOAD
DETAILS
Type here to search


npm
D:\NodeJS\node_js_auth_mongodb>npm install mongodb
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN node_js_auth_mongodb@1.0.0 No repository field.
+ mongodb@3.1.8
added 10 packages from 7 contributors and audited 11 packages in 4.082s
found 0 vulnerabilities
D:\NodeJS\node_js_auth_mongodb>npm install crypto
npm WARN deprecated crypto@1.0.1: This package is no longer supported. It's now a built-in Node module. If you've depend
ed on crypto, you should switch to the one that's built-in.
npm WARN node_js_auth_mongodb@1.0.0 No repository field.
+ crypto@1.0.1
added 1 package and audited 12 packages in 1.817s
found 0 vulnerabilities
D:\NodeJS\node_js_auth_mongodb>npm install express
npm WARN node_js_auth_mongodb@1.0.0 No repository field.
+ express@4.16.4
added 47 packages from 35 contributors and audited 133 packages in 4.409s
found 0 vulnerabilities
D:\NodeJS\node_js_auth_mongodb>npm install body-parser


This folder is empty.
Administrator: Command Prompt
save it as a dependency in the package.json file.
Press ^C at any time to quit.
package name: (node_js_auth_mongodb) node_js_auth_mongodb
version: (1.0.0) 1.0.0
description: This project demo how to login with Nodejs and MongoDB
entry point: (index.js) index.js
test command:
git repository:
keywords:
author: EDMTDev
license: (ISC) ISC
About to write to D:\NodeJS\node_js_auth_mongodb\package.json:
{
"name": "node_js_auth_mongodb",
"version": "1.0.0",
"description": "This project demo how to login with Nodejs and MongoDB",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"author": "EDMTDev",
"license": "ISC"
}
Is this OK? (yes) yes
D:\NodeJS\node_js_auth_mongodb>