Planning the models

|  |  |  |
| --- | --- | --- |
| Model No | Model Name | Description |
| 01 | USER | User signup/signin. |
| 02 | Profile | Life Member (verified sign in the name) & General Member, admin |
| 03 | Home page | All show in home page |
|  |  |  |
|  |  |  |

Project Setup

* Create a folder named backend.
* Open cmd and write: npm init –y
* npm i express bcrypt cors dotenv joi jsonwebtoken lodash morgan

now create .env file:

|  |
| --- |
| MONGODB\_URL\_LOCAL=mongodb://localhost:27017/city-dental-college-alumni-association  NODE\_ENV=development  PORT=3001  JWT\_SECRET\_KEY=JWT\_SECRET\_KEY |

Now create server.js:

require('dotenv/config')

const app = require('./app')

const mongoose = require('mongoose');

mongoose.connect(process.env.MONGODB\_URL\_LOCAL,{

    useNewUrlParser: true,

    useUnifiedTopology: true,

    useCreateIndex: true

})

.then(()=> console.log("Connected to MongoDB"))

.catch(err=> console.log("MongoDB Connection Failed"));

const port = process.env.PORT || 3001;

app.listen(port, ()=>{

    console.log(`App running on port ${port}`);

})

**Create app.js:**

const express = require('express');

const app = express();

const cors = require('cors');

const morgan = require('morgan');

app.use(express.json());

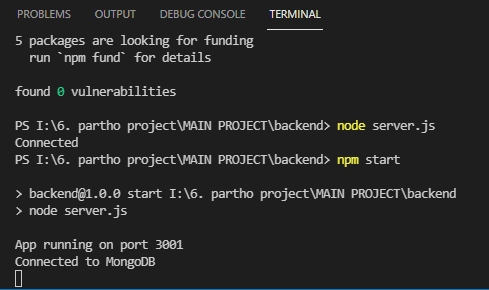
app.use(cors());

if(process.env.NODE\_ENV === 'development'){

    app.use(morgan);

}

module.exports = app;



Node – authentication and authorization

User Model

Create model/user.js:

const { Schema, model} = require('mongoose');

const joi = require('joi');

const jwt = require ('jsonwebtoken');

const Joi = require('joi');

const userSchema = Schema({

    name: {

        type: String,

        required: true,

        minlength: 3,

        maxlength: 50

    },

    email:{

        type: String,

        require: true,

        minlength: 5,

        maxlength: 255,

        unique: true,

    },

    batchNo: {

        type: String,

        required: true,

        minlength: 3,

        maxlength: 50

    },

    BMDCregNo: {

        type: String,

        required: true,

        minlength: 3,

        maxlength: 50

    },

    phoneNo: {

        type: String,

        required: true,

    },

    bloodGroup: {

        type: String,

        enum: ['Select', 'A+ ve', 'B+ ve', 'O+ ve', 'AB+ ve', 'AB- ve', 'A- ve', 'B- ve', 'O- ve'],

        default: 'Select'

    },

    ChamberName: {

        type: String,

        required: true,

        minlength: 3,

        maxlength: 50

    },

    Designation: {

        type: String,

        required: true,

        minlength: 3,

        maxlength: 50

    },

    Address: {

        type: String,

        required: true,

        minlength: 3,

        maxlength: 50

    },

    password: {

        type: String,

        require: true,

        minlength: 6,

        maxlength: 1024

    },

    role:{

        type: String,

        enum: ['user', 'admin'],

        default: 'user'

    },

    UserType: {

        type: String,

        enum: ['Life Member', 'General Member'],

        default: 'General Member'

    },

    TransactionID: {

        type: String,

        required: true,

        minlength: 1,

        maxlength: 200

    },

},{timestamps: true});

userSchema.methods.generateJWT = function(){

    const token = jwt.sign({

        \_id: this.\_id,

        name: this.name,

        email: this.email,

        batchNo: this.batchNo,

        BMDCregNo: this.BMDCregNo,

        phoneNo: this.phoneNo,

        bloodGroup: this.bloodGroup,

        ChamberName: this.ChamberName,

        Designation: this.Designation,

        Address: this.Address,

        password: this.password,

        role: this.role,

        UserType: this.UserType,

        TransactionID: this.TransactionID,

    }, process.env.JWT\_SECRET\_KEY, {expiresIn: "7d"});

    return token;

}

const validateUser = user => {

    const schema = Joi.object ({

        name: Joi.string().min(3).max(100).required(),

        email: Joi.string().min(5).max(255).required(),

        batchNo: Joi.string().min(3).max(255).required(),

        BMDCregNo: Joi.string().min(2).max(255).required(),

        phoneNo: Joi.string().min(11).max(15).required(),

        bloodGroup: Joi.string().required(),

        ChamberName: Joi.string().required(),

        Designation: Joi.string().required(),

        Address: Joi.string().required(),

        password: Joi.string().min(5).max(255).required(),

        TransactionID: Joi.string().min(3).max(255).required()

    });

    return schema.validate(user);

}

module.exports.User = model('User', userSchema);

module.exports.validate = validateUser;

**Create routers/userRouter.js:**

const router = require('express').Router();

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

module.exports = router;

**Create controllers/userControllers.js:**

module.exports.signUp = async (req, res) => {

}

module.exports.signIn = async (req, res) => {

**}**

**Now routers/userRouter.js:**

const router = require('express').Router();

const {signIn, signUp} = require('../controllers/userControllers')

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

module.exports = router;

app.js:

const express = require('express');

const app = express();

const cors = require('cors');

const morgan = require('morgan');

app.use(express.json());

app.use(cors());

const userRouter = require ('./routers/userRouter');

if(process.env.NODE\_ENV === 'development'){

    app.use(morgan);

}

app.use('/api/user', userRouter);

module.exports = app;

**in controllers/userController.js:**

const bcrypt = require ('bcrypt');

const \_ = require ('lodash');

const {User, validate} = require('../model/user');

module.exports.signUp = async (req, res) => {

    const {error} = validate (req.body);

    if(error) return res.status(400).send(error.details[0].message);

    let user = {};

    user = await User.findOne({email: req.body.email});

    if(user) return res.status(400).send("User Already Exist");

    user = new User(\_.pick(res.body, ['name', 'email', 'batchNo', 'BMDCregNo', 'phoneNo','bloodGroup', 'ChamberName', 'Designation', 'Address', 'password', 'TransactionID']));

    const salt = await bcrypt.genSalt(10);

    user.password = await bcrypt.hash(user.password, salt);

    const token = user.generateJWT();

    try{

        const result = await user.save();

        return res.status(201).send({

            message: "registration successful",

            token: token,

            user: \_.pick(resukt, ["\_id", "name", "email"])

        })

    }catch(error){

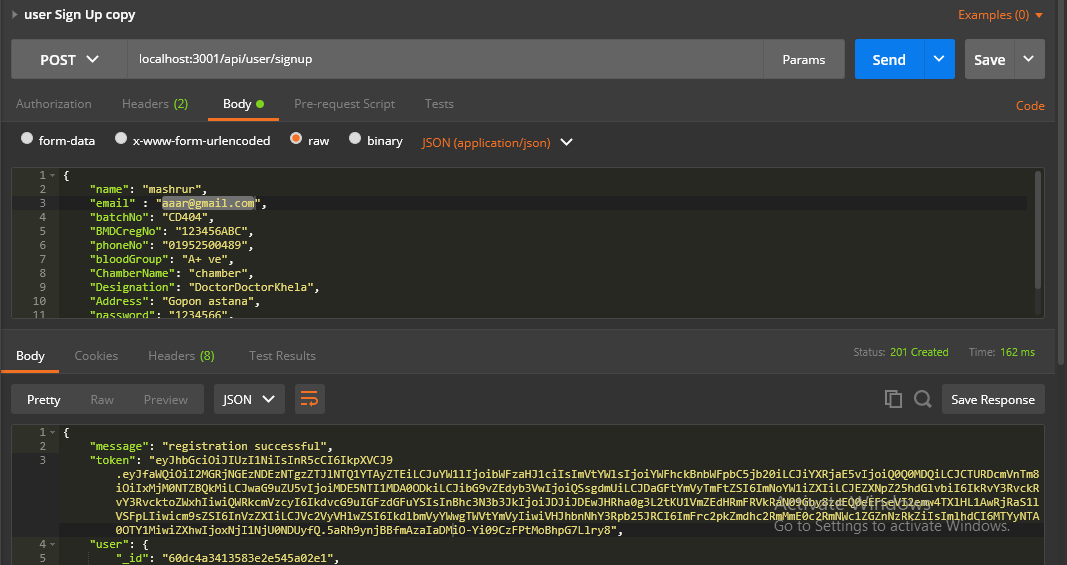
        return res.status(500).send("Something failed!");

    }

}

module.exports.signIn = async (req, res) => {

}



# Handle rejected promise Errors

* npm i express-async-errors

app.js:

require ('express-async-error')

const express = require('express');

const app = express();

const cors = require('cors');

const morgan = require('morgan');

app.use(express.json());

app.use(cors());

const userRouter = require ('./routers/userRouter');

if(process.env.NODE\_ENV === 'development'){

    app.use(morgan('dev'));

}

app.use('/api/user', userRouter);

app.use((err, req, res, next) => {

    return res.status(500).send("Something Went Wrong");

})

module.exports = app;

**Now try…catch is not needed:**

const bcrypt = require('bcrypt');

const \_ = require('lodash');

const {User, validate} = require('../model/user');

module.exports.signUp = async (req, res) => {

    const { error } = validate(req.body);

    if (error) return res.status(400).send(error.details[0].message);

    let user = {};

    user = await User.findOne({ email: req.body.email });

    if(user) return res.status(400).send("User Already Exist");

    user = new User(\_.pick(req.body, ['name', 'email', 'batchNo', 'BMDCregNo', 'phoneNo','bloodGroup', 'ChamberName', 'Designation', 'Address', 'password', 'TransactionID']));

    const salt = await bcrypt.genSalt(10);

    user.password = await bcrypt.hash(user.password, salt);

    const token = user.generateJWT();

        const result = await user.save();

        return res.status(201).send({

            message: "registration successful",

            token: token,

            user: \_.pick(result, ["\_id", "name", "email"])

        })

}

module.exports.signIn = async (req, res) => {

    let user = await User.findOne ({email: req.body.email});

    if(!user) return res.status(400).send("Invalid email or password");

    const validUser = await bcrypt.compare(req.body.password, user.password);

    if(!validUser) return res.status.send('Invalid email or password');

    const token = user.generateJWT();

    return res.status(200).send({

        message: "Login successful",

        token: token,

        user: \_.pick(user, ["\_id", "name", "email"])

    });

}

**Create middlewares/error.js:**

module.exports = (err, req, res, next) => {

    return res.status(500).send("Something went wrong!");

}

Now in app.js:

require ('express-async-error')

const express = require('express');

const app = express();

const cors = require('cors');

const morgan = require('morgan');

const error = require('./middlewares/error')

app.use(express.json());

app.use(cors());

const userRouter = require ('./routers/userRouter');

if(process.env.NODE\_ENV === 'development'){

    app.use(morgan('dev'));

}

app.use('/api/user', userRouter);

app.use(error);

module.exports = app;

Authorization middleware

**Create middlewares/authorize.js:**

const jwt = require('jsonwebtoken');

module.exports = async function (res, req, next) {

    let token = req.header('Authorization');

    if(!token) return res.status(401).send('Access denied! No token provided!');

    else token = token.split(" ")[1].trim();

    const decoded = await jwt.verify(token, process.env.JWT\_SECRET\_KEY);

    if(!decoded) return res.status(400).send('Invalid token');

    req.user = decoded;

    next();

}

**Create middlewares/admin.js:**

module.exports = function (req, res, next){

    if(req.user.role != "admin") return res.status(403).send("Forbidden!");

}

NODE – Activity Name and activity details

**Create models/activityName.js.js: (instead of category.js):**

const { Schema, model } = require('mongoose');

const Joi = require('joi');

module.exports.Activity = model('Category', Schema({

    name: {

        type: String,

        unique: true

    }

}, {timeStamps: true}));

module.exports.validate = activity => {

    const schema = Joi.object({

        name: Joi.string().min(1).max(50).required()

    });

    return schema.validate(activity);

}

**Create controllers/activityControllers.js:**

const \_ = require('lodash');

const { Activity, validate } = require('../model/activityName');

module.exports.createActivity = async (req, res) => {

}

module.exports.getCategory = async (req, res) =>{

}

**Create routers/activityRouter.js:**

const router = require ('express').Router();

const {getActivity, createActivity} = require('../controllers/activityControllers');

const authorize = require('../middlewares/authorize');

const admin = require('../middlewares/admin');

router.route('/')

    .post([authorize, admin], createActivity)

    .get(getActivity);

module.exports = router;

**in app.js:**

require ('express-async-error')

const express = require('express');

const app = express();

const cors = require('cors');

const morgan = require('morgan');

const error = require('./middlewares/error')

const activityRouter = require ('./routers/activityRouter')

app.use(express.json());

app.use(cors());

const userRouter = require ('./routers/userRouter');

if(process.env.NODE\_ENV === 'development'){

    app.use(morgan('dev'));

}

app.use('/api/user', userRouter);

app.use('/api/activity', activityRouter);

app.use(error);

module.exports = app;

**controllers/activityControllers.js:**

const \_ = require('lodash');

const { Activity, validate } = require('../model/activityName');

module.exports.createActivity = async (req, res) => {

    const { error } = validate(\_.pick(req.body, ["name"]));

    if(error) return res.status(400).send(error.details[0].message);

    const activity = new Activity(\_.pick(req.body, ["name"]));

    const result = await activity.save();

    return res.status(201).send({

        message: "Activity created successfully!",

        data: {

            name: result.name

        }

    })

}

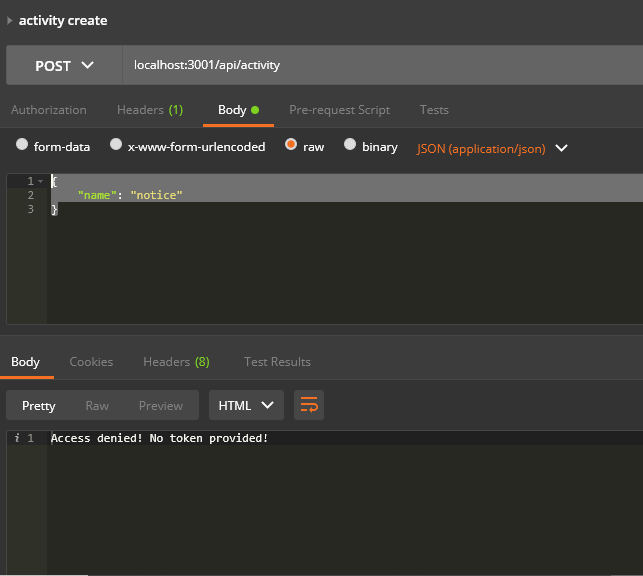
module.exports.getActivity = async (req, res) =>{

    const activities = await Activity.find()

        .sort({name: 1});

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

}



**In Middlewares/Authorize.js:**

const jwt = require('jsonwebtoken');

module.exports = async function (req, res, next) {

    let token = req.header('Authorization');

    if (!token) return res.status(401).send('Access denied! No token provided!');

    // Bearer 1234abcd

    else token = token.split(" ")[1].trim();

    try{

        const decoded = await jwt.verify(token, process.env.JWT\_SECRET\_KEY);

        // decoded => undefined

        if (!decoded) return res.status(400).send('Invalid token');

        req.user = decoded;

        next();

    }catch(err){

        return res.status(400).send('Invalid Token!');

    }

}

**Admin.js:**

module.exports = function (req, res, next){

    if(req.user.role != "admin") return res.status(403).send("Forbidden!");

    next();

}

**ActivityControllers.js:**

const \_ = require('lodash');

const { Activity, validate } = require('../model/activityName');

module.exports.createActivity = async (req, res) => {

    const { error } = validate(\_.pick(req.body, ["name"]));

    if(error) return res.status(400).send(error.details[0].message);

    const activity = new Activity(\_.pick(req.body, ["name"]));

    const result = await activity.save();

    return res.status(201).send({

        message: "Activity Created Successfully!",

        data: {

            name: result.name

        }

    })

}

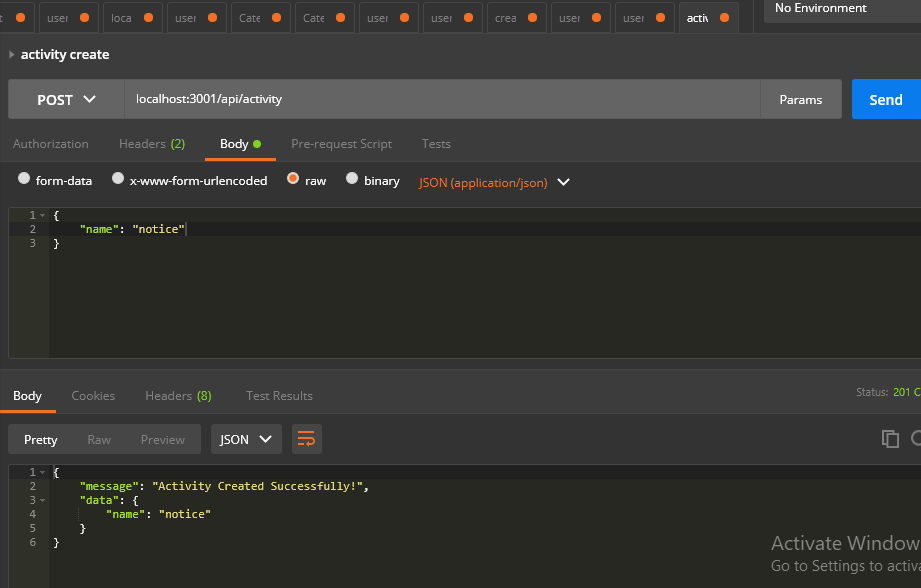
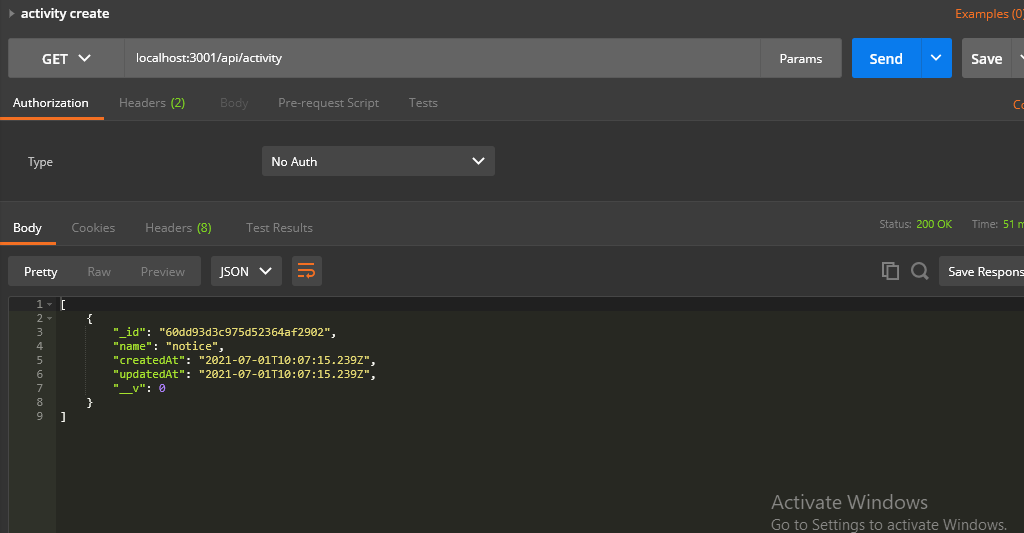
module.exports.getActivity = async (req, res) =>{

    const activities = await Activity.find()

        .sort({name: 1});

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

}

**Rewrite middlewares/authorize.js:**

const jwt = require('jsonwebtoken');

module.exports = async function (req, res, next) {

    let token = req.header('Authorization');

    if (!token) return res.status(401).send('Access denied! No token provided!');

    // Bearer 1234abcd

    else token = token.split(" ")[1].trim();

    try{

        const decoded = await jwt.verify(token, process.env.JWT\_SECRET\_KEY);

        // decoded => undefined

        if (!decoded) return res.status(400).send('Invalid token');

        req.user = decoded;

        next();

    }catch(err){

        return res.status(400).send('Invalid Token!');

    }

}

.gitignore:

/node\_modules

\*.env

**Create model/ detail.js:**

const { Schema, model } = require('mongoose');

const Joi = require('joi');

module.exports.Detail = model('Detail', Schema({

    name: String,

    description: String,

    activity: {

        type: Schema.Types.ObjectId,

        ref: 'Activity',

        require: true

    },

    photo: {

        data: Buffer,

        contentType: String,

    }

}, {timestamps: true}));

module.exports.validate = detail => {

    const schema = Joi.object({

        name: Joi.string().min(3).max(255).required(),

        description: Joi.string().max(2000).required(),

        activity: Joi.string().required(),

    });

    return schema.validate(detail);

}

**Create controllers/detailController.js:**

const \_ = require('lodash');

const formidable = require ('formidable');

const fs = require ('fs');

const { Detail, validate } = require('../model/detail');

const { valid } = require ('joi');

const { parse } = require('path');

module.exports.createDetail = async (req, res) => {

    let form = new formidable.IncomingForm();

    form.keepExtensions = true;

    form.parse(req, (err, fields, files) => {

        if(err) return res.status(400).send("Something Went Wrong!");

        const { error } = validate(\_.pick(fields, ["name", "description", "activity"]));

        if(error) return res.status(400).send(error.details[0].message);

        const detail = new Detail(fields);

        if(files.photo) {

            fs.readFile(files.photo.path, (err, data) => {

                if(err) return res.status(400).send("Problem in the file data!");

                detail.photo.data = data;

                detail.photo.contentType = files.photo.type;

                detail.save((err, result) =>{

                    if(err) res.status(500).send("Internal Server Error!");

                    else return res.status(201).send({

                        message: "Created Successfully",

                        data: \_.pick(result, ["name", "description", "activity"])

                    })

                })

            })

        }else{

            return res.status(400).send("No image provided");

        }

    })

}

//Query parameter

//api/product?order=desc$sortBy=name&limit=10

module.exports.getDetail = async (req, res) => {

    let order = req.query.order === 'desc' ?-1 : 1;

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

    let limit = req.query.limit ? parseInt(req.query.limit) : 10;

    const details = await Detail.find()

    .select({ photo: 0})

    .sort({ [sortBy]: order})

    .limit(limit)

    .populate('activity', 'name');

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

}

module.exports.getDetailById = async (req, res) => {

    const detailId = req.params.id;

    const detail = await Detail.findById(detailId)

        .select({ photo: 0 })

        .populate('activity', 'name');

    if(!detail) res.status(404).send("Not Found!");

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

}

module.exports.getPhoto = async (req, res) => {

    const detailId = req.params.id;

    const detail = await Detail.findById(detailId)

        .select({ photo: 1, \_id: 0})

    res.set ('Content-Type', detail.photo.contentType);

    return res.status(200).send(detail.photo.data);

}

//get detail by Id

//collect form data

//update provided from fields

//update photos (if provided)

module.exports.updateDetailById = async (req, res) => {

    const detailId = req.params.id;

    const detail = await Detail.findById(detailId);

    let form = new formidable.IncomingForm();

    form.keepExtensions = true;

    form.parse(req, (err, fields, files) => {

        if(err) return res.status(400).send("Something wrong");

        const updatedFields = \_.pick(fields, ["name", "description", "activity"]);

        \_.assignIn(detail, updatedFields);

        if(files.photo) {

            fs.readFile(files.photo.path, (err, data) => {

                if(err) return res.status(400).send("Something wrong!");

                detail.photo.data = data;

                detail.photo.contentType = files.photo.type;

                detail.save((err, result) => {

                    if(err) return res.status(500).send ("Something failed!");

                    else return res.status(200).send({

                        message: "updated successfully!!"

                    })

                })

            })

        }else{

            detail.save((err, result) => {

                if(err) return res.status(500).send("Something failed!");

                else return res.status(200).send({

                    message: "updated successfully!!"

                })

            })

        }

    })

}

**Create routers/detailRouter.js:**

const router = require('express').Router();

const {

    getDetail,

    createDetail,

    getDetailById,

    updateDetailById,

    getPhoto

} = require('../controllers/detailController');

const admin = require('../middlewares/admin');

const authorize = require('../middlewares/authorize');

router.route ('/')

    .post([authorize, admin], createDetail)

    .get(getDetail);

router.route('/:id')

    .get(getDetailById)

    .put([authorize, admin], updateDetailById);

router.route('/photo/:id')

    .get(getPhoto)

module.exports = router;

**in app.js:**

require ('express-async-error')

const express = require('express');

const app = express();

const cors = require('cors');

const morgan = require('morgan');

const error = require('./middlewares/error')

const activityRouter = require ('./routers/activityRouter')

const detailRouter = require ('./routers/detailRouter')

app.use(express.json());

app.use(cors());

const userRouter = require ('./routers/userRouter');

if(process.env.NODE\_ENV === 'development'){

    app.use(morgan('dev'));

}

app.use('/api/user', userRouter);

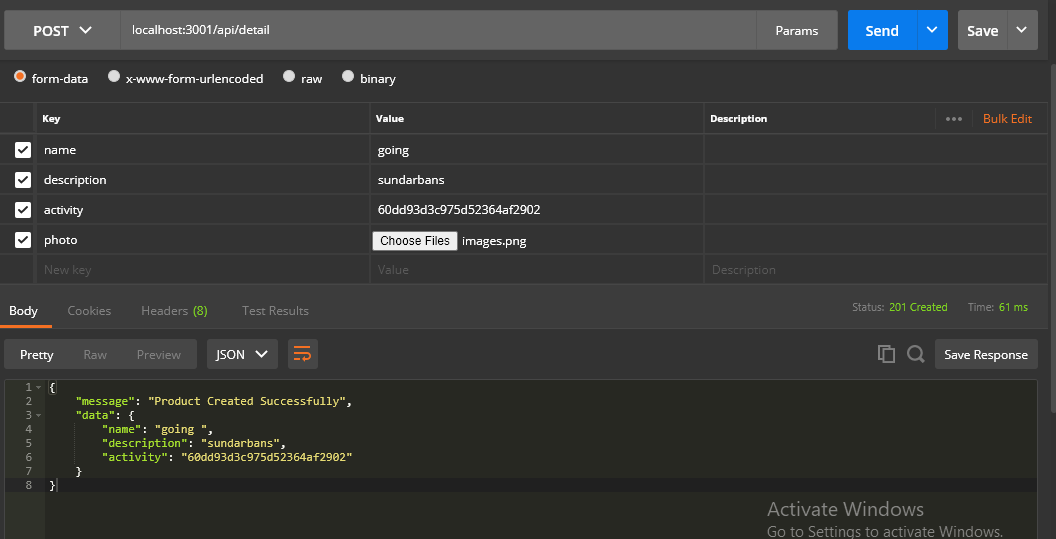
app.use('/api/activity', activityRouter);

app.use('/api/detail', detailRouter);

app.use(error);

module.exports = app;

* npm i formidable



CLEAN CODE:

Middleware/index.js:

const express = require('express');

const cors = require('cors');

const morgan = require('morgan');

const compression = require('compression');

module.exports = (app) => {

    app.use(express.json());

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

    app.use(express.static('public'));

    app.use(cors());

    app.use(compression());

    if (process.env.NODE\_ENV === 'development') {

        app.use(morgan('dev'));

    }

}

Middleware/route.js:

const activityRouter = require ('../routers/activityRouter')

const detailRouter = require ('../routers/detailRouter')

const userRouter = require ('../routers/userRouter');

module.exports = (app) => {

    app.use('/api/user', userRouter);

    app.use('/api/activity', activityRouter);

    app.use('/api/detail', detailRouter);

}

App.js:

require('express-async-errors');

const express = require('express');

const compression = require ('compression')

const app = express();

const error = require('./middlewares/error');

require('./middlewares')(app);

require('./middlewares/routes')(app);

app.use(error);

app.use(compression());

module.exports = app;

**.env:**

|  |
| --- |
| REACT\_APP\_API\_URL=http://localhost:3001/api |

**Create src/config.js:**

export const API = process.env.REACT\_APP\_API\_URL;

**in component/main.js:**

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

const Main = () => {

    return (

    <div>

        <Switch>

        </Switch>

    </div>

    )

}

export default Main;

**create component/Layout.js:**

import {useEffect} from 'react';

const Layout = ({title = 'Title', className, children}) => {

    useEffect(()=>{

        document.title = title;

    },[])

    return (

        <div>

            <div className="mb-3">

                <h3>Menu</h3>

            </div>

            <div className={className}>{children}</div>

        </div>

    )

}

export default Layout;

**create components/home/home.js:**

import Layout from '../Layout';

const Home = () => {

    return (

        <Layout title="Home page" className="container">

        </Layout>

    )

}

export default Home;

**in main.js:**

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

const Main = () => {

    return (

    <div>

        <Switch>

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

        </Switch>

    </div>

    )

}

export default Main;

**in app.js:**

import Main from './components/Main';

import {BrowserRouter} from 'react-router-dom'

function App() {

  return (

    <BrowserRouter>

    <Main />

</BrowserRouter>

  );

}

export default App;

page 64

\*\* add