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());

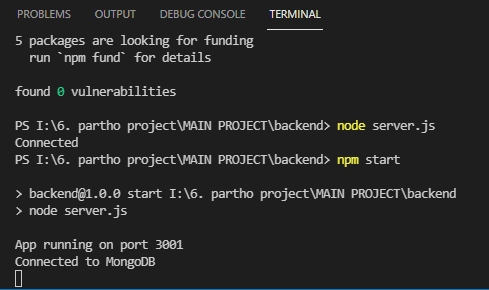
app.use(cors());

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){

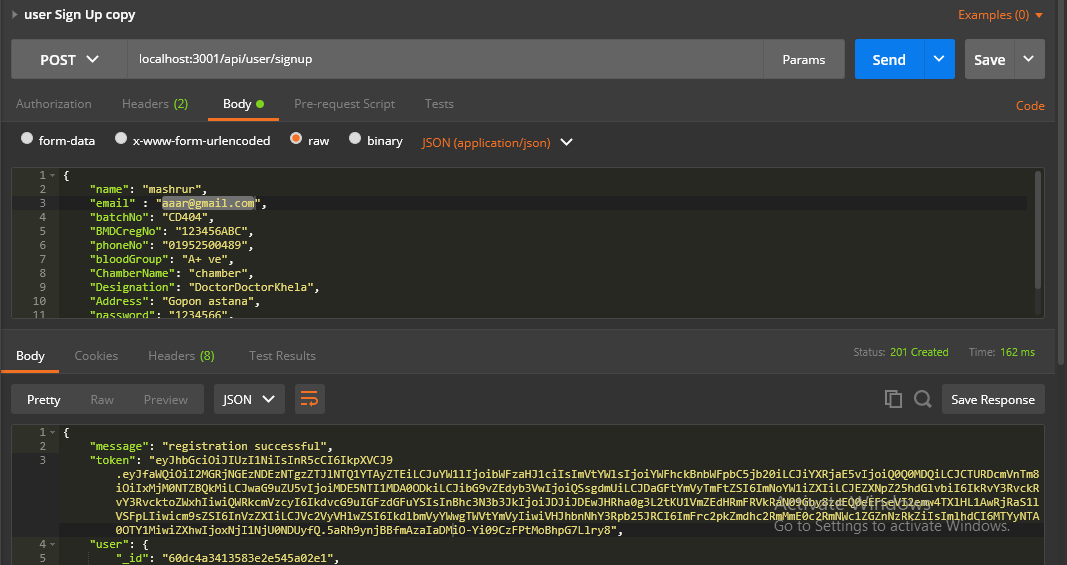
        return res.status(500).send("Something failed!");

    }

}

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

        }

    })

}

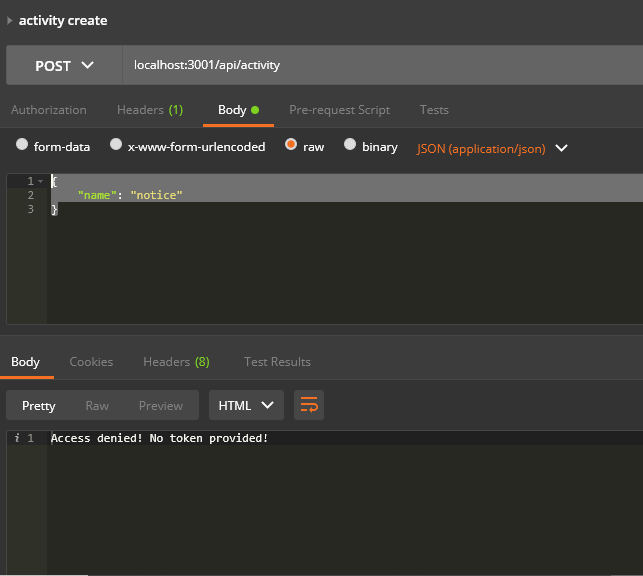
module.exports.getActivity = async (req, res) =>{

    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

        }

    })

}

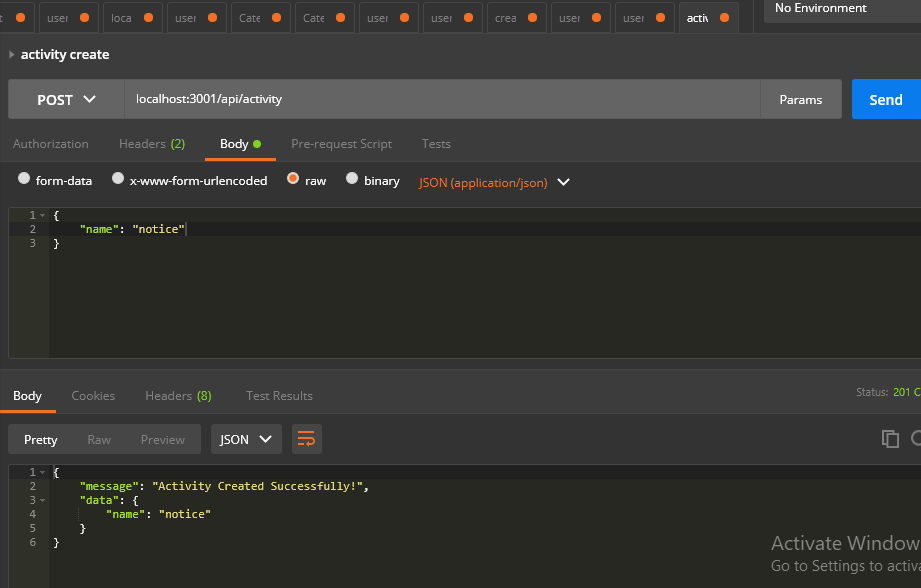
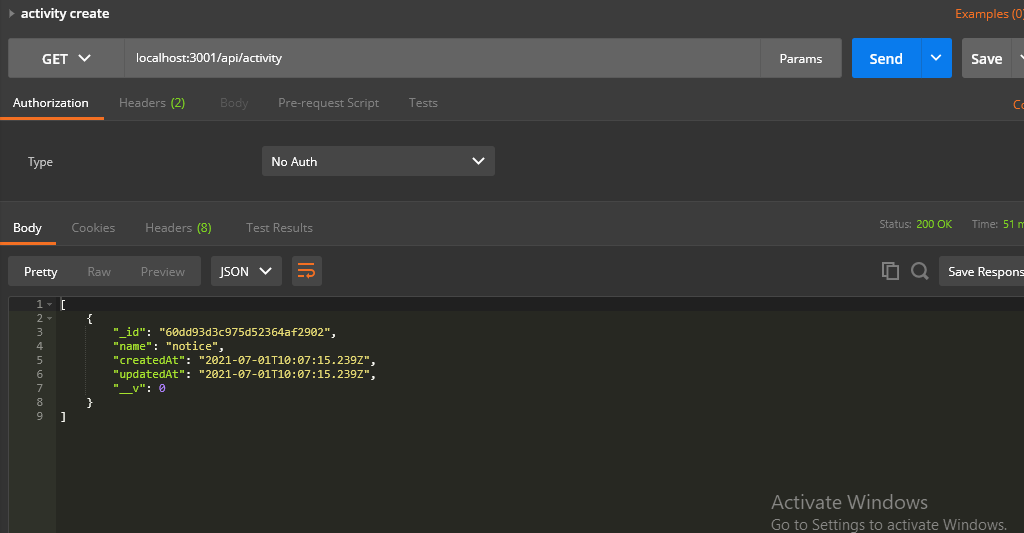
module.exports.getActivity = async (req, res) =>{

    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);

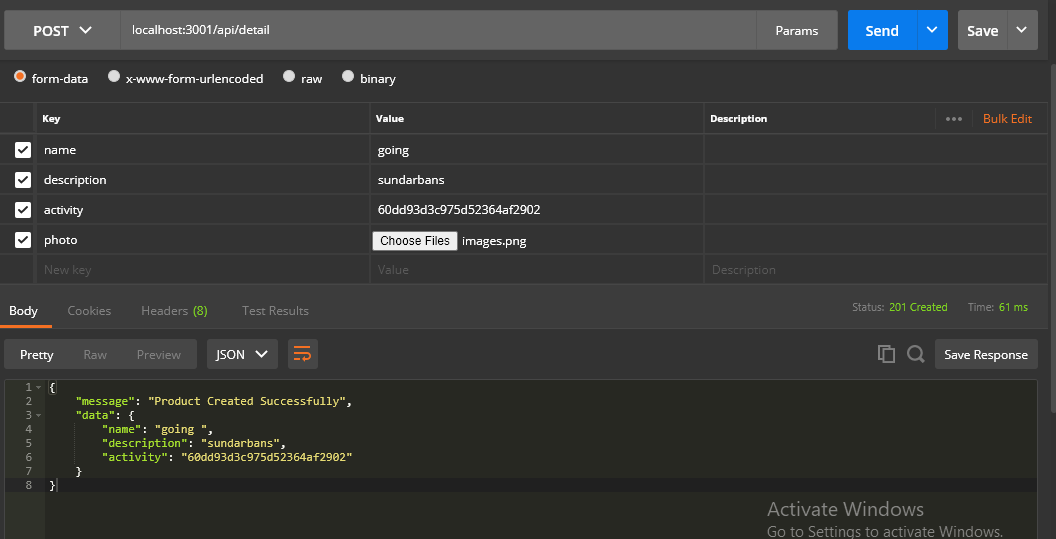
app.use('/api/activity', activityRouter);

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;

Add 2 data in model/user.js

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

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'

    },

    UserStatus: {

        type: String,

        enum: ['In review', 'Active', 'Renew'],

        default: 'In review'

    },

    timeRemain:{

        type: String,

        require: true,

        default: new Date().toJSON().slice(0,10)

    },

    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,

        UserStatus: this.UserStatus,

        timeRemain: this.timeRemain,

        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;

now, in frontend:

**login.js:**

import { useState } from "react";

import Layout from "../Layout";

import '../style/style.css'

const Login = () => {

    const [values, setValues] = useState({

        email : '',

        password: '',

        error: false,

        loading: false,

        disabled: false,

        redirect: false

    });

    const {email, password, loading, error, redirect, disabled} = values;

    const signInForm = () => (

      <div className="hero">

          <div className="form-box">

              <div className="button-box">

                  <div id="btn"></div>

                  <button type="button" className="toggle-btn">Log In</button>

              </div>

              <form id = "login" className="input-group">

                    <input type="email" name="email" className="input-field" placeholder="Email" required />

                    <input type="password" name="password" className="input-field" placeholder="Password" required />

                    <button type="submit" class="submit-btn" disable = {disabled}>Login</button>

              </form>

          </div>

      </div>

    );

    return (

        <Layout title = "Login" className = "container col-md-8 offset-md-2">

            <h3>Login here</h3>

            <hr />

            {signInForm()}

            <hr/>

        </Layout>

    );

}

export default Login;

**..style/style.css:**

.hero

{

    height: 100%;

    width: 100%;

    background-image: linear-gradient(rgba(0,0,0,0.4),rgba(0,0,0,0.4));

    background-position: center;

    position: static;

    background-size: cover;

}

.form-box

{

    width: 380px;

    height: 480px;

    position: relative;

    margin: 6% auto;

    background: #fff;

    padding: 5px;

    overflow: hidden;

}

.button-box

{

    width: 220px;

    margin: 35px auto;

    position: relative;

    box-shadow: 0 0 20px 9px #ff61241f;

    border-radius: 30px;

}

.toggle-btn

{

    padding: 10px 30px;

    cursor: pointer;

    background: transparent;

    border: 0;

    outline: none;

    position: relative;

    position: relative;

}

#btn

{

    top: 0;

    left: 0;

    position: absolute;

    width: 110px;

    height: 100%;

    background: linear-gradient( to right, #ff105f, #ffad06);

    border-radius: 30px;

    transition: .5s;

}

.input-group

{

    top: 180px;

    position: absolute;

    width: 280px;

    transition: .5s;

}

.input-field

{

    width: 100%;

    padding: 10px 0;

    margin: 5px 0;

    border-left: 0;

    border-top: 0;

    border-right: 0;

    border-bottom: 1px solid #999;

    outline: none;

    background: transparent;

}

.submit-btn

{

    width: 85%;

    padding: 10px 30px;

    cursor: pointer;

    display: block;

    margin: auto;

    background: linear-gradient(to right, #ff105f, #ffad06);

    border: 0;

    outline: none;

    border-radius: 30px;

}

.check-box

{

    margin: 30px 10px 30px 0;

}

span

{

    color: #777;

    font-size: 12px;

    bottom: 68px;

    position: absolute;

}

#login

{

    left: 50px;

}

#register

{

    left: 450px;

}

.form-box h1

{

    color: black;

    margin-top: 200px !important;

    text-align: center;

    text-transform: uppercase;

}

Register.js:

import { useState } from "react";

import Layout from "../Layout";

//import { showError, showLoading } from "../../utils/messages";

import '../style/style.css'

const Register = () => {

    const [values, setValues] = useState({

        name: '',

        email: '',

        batchNo: '',

        BMDCregNo: '',

        PhoneNo: '',

        bloodGroup: '',

        ChamberName: '',

        Designation: '',

        Address: '',

        password: '',

        TransactionID : '',

        error: false,

        loading: false,

        disabled: false,

        success: false

    });

    const {name, email, batchNo, BMDCregNo ,PhoneNo ,bloodGroup,ChamberName,Designation,Address,password,TransactionID,success, error, loading, disabled} = values;

    const handleChange = e => {

        setValues({

            ...values,

            error: false,

            [e.target.name]: e.target.value

        })

    }

    const handleSubmit = e => {

        e.preventDefault();

        alert(JSON.stringify(values));

    }

    const signUpForm = () => (

        <div className="hero">

        <div className="form-box">

            <div className="button-box">

                <div id="btn"></div>

                <button type="button" class="toggle-btn" onclick="register()">Register</button>

            </div>

        <form onSubmit={handleSubmit} id="login" className="input-group">

            <input type="text" placeholder="Your Name" name="name" className="input-field" value={name} required onChange={handleChange} />

            <input type="email" placeholder="Email" name="email" className="input-field" value={email} required onChange={handleChange} />

            <input type="text" placeholder="Batch No." name="batchNo" className="input-field" value={batchNo} required onChange={handleChange} />

            <input type="number" placeholder="BMDC Registration No" name="BMDCregNo" className="input-field" value={BMDCregNo} required onChange={handleChange} />

            <input type="number" placeholder="Phone No" name="phoneNo" className="input-field" value={PhoneNo} required onChange={handleChange} />

            <input type="text" placeholder="Blood Group" name="bloodGroup" className="input-field" value={bloodGroup} required onChange={handleChange} />

            <input type="text" placeholder="Chamber Name" name="ChamberName" className="input-field" value={ChamberName} required onChange={handleChange} />

            <input type="text" placeholder="Designation" name="Designation" className="input-field" value={Designation} required onChange={handleChange} />

            <input type="text" placeholder="Address" name="Address" className="input-field" value={Address} required onChange={handleChange} />

            <input type="password" placeholder="Password" name="password" className="input-field" value={password} required onChange={handleChange} />

            <input type="text" placeholder="Transaction ID" name="TransactionID" className="input-field" value={TransactionID} required onChange={handleChange} />

            <button type="submit" class="btn btn-primary" disabled={disabled}>REGISTER</button>

        </form>

        </div>

      </div>

    );

    return(

        <Layout title = "Register" className="container col-md-8 offset-md-2">

            <h3>Register Here</h3>

            {signUpForm()}

        </Layout>

    );

}

export default Register;

now create error and success message:

**src/utils/messages.js:**

export const showError = (error, msg) => {

    if (error) return <div className="alert alert-danger">{msg}</div>

}

export const showSuccess = (success, msg) => {

    if (success) return <div className="alert alert-success">{msg}</div>

}

export const showLoading = loading => {

    if (loading) return <div className="alert alert-info">Loading.....</div>

}

**In src/utils/config.js:**

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

**now create src/api/apiAuth.js:**

import axios from 'axios';

import { API } from '../utils/config';

export const register = (user) => {

    return axios.post(`${API}/user/signup`, user, {

        headers: {

            'Content-Type': 'application/json'

        }

    })

};

export const login = (user) => {

    return axios.post(`${API}/user/signin`, user, {

        headers: {

            'Content-Type': 'application/json'

        }

    })

};

**In backend:**

**Model/user.js:**

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

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,

        required: true,

    },

    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'

    },

    UserStatus: {

        type: String,

        enum: ['In review', 'Active', 'Renew'],

        default: 'In review'

    },

    timeRemain:{ //for time track

        type: String,

        require: true,

        default: new Date().toJSON().slice(0,10)

    },

    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,

        UserStatus: this.UserStatus,

        timeRemain: this.timeRemain,

        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;

**In frontend:**

**User/login.js:**

import { useState } from "react";

import Layout from "../Layout";

import '../style/style.css';

import { Redirect } from "react-router";

import {login} from '../../api/apiAuth';

import { showError, showLoading } from "../../utils/messages";

import {authenticate} from '../../utils/auth'

const Login = () => {

    const [values, setValues] = useState({

        email : '',

        password: '',

        error: false,

        loading: false,

        disabled: false,

        redirect: false

    });

    const {email, password, loading, error, redirect, disabled} = values;

    const handleChange = e => {

        setValues({

            ...values,

            error: false,

            [e.target.name]: e.target.value

        })

    }

    const handleSubmit = e => {

        e.preventDefault();

        setValues ({...values, error: false, loading: true, disabled: true});

        login ({email, password})

            .then(response => {

                authenticate(response.data.token, () => {

                    setValues({

                        email: '',

                        password: '',

                        success: true,

                        disabled: false,

                        loading: false,

                        redirect: true

                    })

                })

            })

        .catch(err =>{

            let errMsg = 'Something Went Wrong';

            if(err.response){

                errMsg = err.response.data;

            }else{

                errMsg = 'Something Went Wrong!';

            }

            setValues({...values, error: errMsg, disabled: false, loading: false})

        })

    }

    const signInForm = () => (

      <div className="hero">

          <div className="form-box-login">

              <div className="button-box">

                  <div id="btn"></div>

                  <button type="button" className="toggle-btn">Log In</button>

              </div>

              <form id = "login" className="input-group" onSubmit={handleSubmit}>

                    <input type="email" name="email" className="input-field" placeholder="Email" required onChange={handleChange}/>

                    <input type="password" name="password" className="input-field" placeholder="Password"  onChange={handleChange} required />

                    <button type="submit" class="submit-btn" disable = {disabled}>Login</button>

                    <a href="/register">create Account</a>

              </form>

          </div>

      </div>

    );

    const redirectUser = () => {

        if (redirect) return <Redirect to="/" />

    }

    return (

        <Layout title = "Login" className = "container col-md-8 offset-md-2">

            {redirectUser()}

            {showLoading(loading)}

            {showError(error, error)}

            <h3>Login here</h3>

            <hr />

            {signInForm()}

            <hr/>

        </Layout>

    );

}

export default Login;

**user/register.js:**

import { useState } from "react";

import Layout from "../Layout";

import { showError, showLoading, showSuccess } from "../../utils/messages";

import '../style/style.css'

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

import { register } from "../../api/apiAuth";

const Register = () => {

    const [values, setValues] = useState({

        name: '',

        email: '',

        batchNo: '',

        BMDCregNo: '',

        phoneNo: '',

        bloodGroup: '',

        ChamberName: '',

        Designation: '',

        Address: '',

        password: '',

        TransactionID : '',

        error: false,

        loading: false,

        disabled: false,

        success: false

    });

    const {name, email, batchNo, BMDCregNo ,phoneNo ,bloodGroup,ChamberName,Designation,Address,password,TransactionID,success, error, loading, disabled} = values;

    const handleChange = e => {

        setValues({

            ...values,

            error: false,

            [e.target.name]: e.target.value

        })

    }

    const handleSubmit = e => {

        e.preventDefault();

        setValues({...values, error: false, loading: true, disabled: true});

        register({name, email, batchNo, BMDCregNo ,phoneNo ,bloodGroup,ChamberName,Designation,Address,password,TransactionID})

        .then(response => {

           //write the success message here

            setValues({

                name: '',

                email: '',

                batchNo: '',

                BMDCregNo: '',

                phoneNo: '',

                bloodGroup: '',

                ChamberName: '',

                Designation: '',

                Address: '',

                password: '',

                TransactionID : '',

                success: true,

                disabled: false,

                loading: false

             })

        })

        .catch(err => {

            let errMsg = 'Something Went Wrong!';

            if(err.response) {

                errMsg = err.response.data;

            }else{

                errMsg = "Something Went Wrong";

            }

            setValues({...values, error: errMsg, disabled: false, loading: false})

        })

    }

    const signUpForm = () => (

        <div className="hero">

        <div className="form-box">

            <div className="button-box">

                <div id="btn"></div>

                <button type="button" class="toggle-btn" onclick="register()">Register</button>

            </div>

        <form onSubmit={handleSubmit} id="login" className="input-group">

            <input type="text" placeholder="Your Name" name="name" className="input-field" value={name} required onChange={handleChange} />

            <input type="email" placeholder="Email" name="email" className="input-field" value={email} required onChange={handleChange} />

            <input type="text" placeholder="Batch No." name="batchNo" className="input-field" value={batchNo} required onChange={handleChange} />

            <input type="number" placeholder="BMDC Registration No" name="BMDCregNo" className="input-field" value={BMDCregNo} required onChange={handleChange} />

            <input type="number" placeholder="Phone no" name="phoneNo" className="input-field" value={phoneNo} required onChange={handleChange} />

            <input type="text" placeholder="Blood Group (exm: A+ ve, B+ ve)" name="bloodGroup" className="input-field" value={bloodGroup} required onChange={handleChange} />

            <input type="text" placeholder="Chamber Name" name="ChamberName" className="input-field" value={ChamberName} required onChange={handleChange} />

            <input type="text" placeholder="Designation" name="Designation" className="input-field" value={Designation} required onChange={handleChange} />

            <input type="text" placeholder="Address" name="Address" className="input-field" value={Address} required onChange={handleChange} />

            <input type="password" placeholder="Password" name="password" className="input-field" value={password} required onChange={handleChange} />

            <input type="text" placeholder="Transaction ID" name="TransactionID" className="input-field" value={TransactionID} required onChange={handleChange} />

            <button type="submit" class="btn btn-primary" disabled={disabled}>REGISTER</button>

        </form>

        </div>

      </div>

    );

    return(

        <Layout title = "Register" className="container col-md-8 offset-md-2">

            {showSuccess()}

            {showLoading(loading)}

            {showError(error, error)}

            <h3>Register Here</h3>

            {signUpForm()}

        </Layout>

    );

}

export default Register;

**Menu.js:**

import {Link, withRouter} from 'react-router-dom';

import './style/top.css'

import {signout, isAuthenticated} from '../utils/auth'

const isActive = (history, path) => {

  if(history.location.pathname === path){

    return {color: '#ff9900'}

  }else{

    return {color: 'white'}

  }

}

const Menu = ({history}) => {

    return(

        <div class="wrapper row0" >

        <div id="topbar" class="hoc clear">

          <div class="fl\_left">

            <ul class="nospace">

              <li><i class="fas fa-phone rgtspace-5"></i> +088 01633749109</li>

              <li><i class="far fa-envelope rgtspace-5"></i> cdcaa.page@gmail.com</li>

            </ul>

          </div>

          <div class="fl\_right" >

            <ul class="nospace">

              <li><i class="fas fa-sign-in-alt"></i>

                    <Link style={isActive(history, '/')} to="/">Home</Link>

              </li>

              {!isAuthenticated() && (<>

              <li><i class="fas fa-edit"></i>

                <Link style={isActive(history, '/login')} to="/login">Login</Link>

              </li>

              </> )}

              {isAuthenticated() && (<>

              <li><i class="fas fa-edit"></i>

                <Link  style = {{cursor: 'pointer', color: 'grey'}} onClick={() => {

                  signout(() => {

                    history.push('/login')

                  });

                }}> Log Out</Link>

              </li>

              </> )}

            </ul>

          </div>

        </div>

        <div class="wrapper row1"  >

    <header id="header" class="hoc clear" >

      <div id="logo" class="fl\_left" >

        <h1><a href="/">CDCAA</a></h1>

      </div>

      <nav id="mainav" class="fl\_right">

        <ul class="clear">

          <li class="active"><a href="/"><b>Home</b></a></li>

          <li><a class="drop" href="#">ORGANIZATION</a>

            <ul>

              <li><a href="pages/gallery.html">Journey of CDCAA</a></li>

              <li><a href="pages/full-width.html">Excutive Committee</a></li>

              <li><a href="pages/sidebar-left.html">Sub Committee</a></li>

              <li><a href="pages/sidebar-right.html">Past Leaders</a></li>

            </ul>

          </li>

          <li><a class="drop" href="#">Activities</a>

            <ul>

              <li><a href="#">Soical Responsibilites</a></li>

            </ul>

          </li>

          <li><a href="C:\Users\Asus\Desktop\City Dental\CDCAA\form.html">MY PROFILE</a></li>

          <li><a href="#">NEWS</a></li>

          <li><a href="#">NOTICE</a></li>

          <li><a href="C:\Users\Asus\Desktop\City Dental\CDCAA\pages\gallery.html">GALLERY</a></li>

          <li><a href="#">CONTACT US</a></li>

        </ul>

      </nav>

    </header>

    </div>

      </div>

)

}

export default withRouter(Menu);

**home/Home.js:**

import Layout from '../Layout';

import pic2 from '../img/pic2.jpg'

import pic3 from '../img/pic3.jpg'

import pic4 from '../img/pic4.jpg'

import pic21 from '../img/pic21.jpg'

import {Link, withRouter} from 'react-router-dom';

import {isAuthenticated, signout} from './auth'

const Home = ({history}) => {

    return (

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

             <div id="pageintro" class="hoc clear" style={{backgroundImage: `url(${pic21})`,backgroundRepeat: 'no-repeat',backgroundSize: 'contain',

            overflow: 'hidden', opacity: 0.5}}>

      <article>

        <h3 class="heading">City Dental Alumni Association</h3>

        <p><b>City Dental College Alumni Association <br/>

                        Est. 2017 <br/>

            Non-profit Volunteer Organization <br/>

           Founded by the Former Students of City Dental College and Hospital</b></p>

           {!isAuthenticated() && (<>

            <footer><a className="btn" href="/register">REGISTRATION</a></footer>

           </>)}

           {isAuthenticated() && (<>

                <Link className="btn" style = {{cursor: 'pointer', color: 'white'}} onClick={() => {

                  signout(() => {

                    history.push('/login')

                  });

                }}> Log Out</Link>

              </> )}

      </article>

    </div>

   <section id="introblocks">

      <ul class="nospace group btmspace-80">

        <li class="one\_third first">

          <figure><a class="imgover" href="#"><img src={pic2} alt="" /></a>

            <figcaption>

              <h6 class="heading">Free Medical and Dental Camp</h6>

              <div>

                <p>We are going to make free dental cheack up camp with our doctors from CDC alumni association.</p>

              </div>

            </figcaption>

          </figure>

        </li>

        <li class="one\_third">

          <figure><a class="imgover" href="#"><img src={pic3} alt="" /></a>

            <figcaption>

              <h6 class="heading">Cricket Tournament </h6>

              <div>

                <p>The members of City Dental Alumni Association organised a friendly cricket match and gave the prizes among the winer.</p>

              </div>

            </figcaption>

          </figure>

        </li>

        <li class="one\_third">

          <figure><a class="imgover" href="#"><img src={pic4} alt="" /></a>

            <figcaption>

              <h6 class="heading">ইফতার ২০২১ </h6>

              <div>

                <p>ছিন্ন মূল ও পথ শিশূদের ইফতার বিতরণ, কলেজের ৩য় ও চতুর্থ শ্রেণীর কর্মচারীদের ঈদ সামগ্রী উপহার প্রদান ।</p>

              </div>

            </figcaption>

          </figure>

        </li>

      </ul>

    </section>

        </Layout>

    )

}

export default withRouter (Home);

**home/auth.js:**

import jwt\_decode from 'jwt-decode';

export const authenticate = (token, cb) => {

    if (typeof window !== 'undefined') {

        localStorage.setItem('jwt', JSON.stringify(token));

        cb();

    }

}

export const isAuthenticated = () => {

    if (typeof window === 'undefined') return false;

    if (localStorage.getItem('jwt')) {

        const { exp } = jwt\_decode(JSON.parse(localStorage.getItem('jwt')));

        if ((new Date()).getTime() <= exp \* 1000) {

            return true;

        } else {

            localStorage.removeItem('jwt');

            return false

        }

    } else return false;

}

export const userInfo = () => {

    const jwt = JSON.parse(localStorage.getItem('jwt'));

    const decoded = jwt\_decode(jwt);

    return { ...decoded, token: jwt }

}

export const signout = cb => {

    if (typeof window !== 'undefined') {

        localStorage.removeItem('jwt');

        cb();

    }

}

**Utils/auth.js:**

import jwt\_decode from 'jwt-decode';

export const authenticate = (token, cb) => {

    if (typeof window !== 'undefined') {

        localStorage.setItem('jwt', JSON.stringify(token));

        cb();

    }

}

export const isAuthenticated = () => {

    if (typeof window === 'undefined') return false;

    if (localStorage.getItem('jwt')) {

        const { exp } = jwt\_decode(JSON.parse(localStorage.getItem('jwt')));

        if ((new Date()).getTime() <= exp \* 1000) {

            return true;

        } else {

            localStorage.removeItem('jwt');

            return false

        }

    } else return false;

}

export const userInfo = () => {

    const jwt = JSON.parse(localStorage.getItem('jwt'));

    const decoded = jwt\_decode(jwt);

    return { ...decoded, token: jwt }

}

export const signout = cb => {

    if (typeof window !== 'undefined') {

        localStorage.removeItem('jwt');

        cb();

    }

}

**Utils/config.js:**

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

**utils/messages.js:**

export const showError = (error, msg) => {

    if (error) return <div className="alert alert-danger">{msg}</div>

}

export const showSuccess = (success, msg) => {

    if (success) return <div className="alert alert-success">{msg}</div>

}

export const showLoading = loading => {

    if (loading) return <div className="alert alert-info">Loading.....</div>

}

Now, we need private routes for dashboard implementation.

Create **src/components/protectRoutes/protectRoutes.js:**

import { Route, redirect, Redirect } from "react-router";

import {isAuthenticated} from '../../utils/auth';

const PrivateRoute = ({children, ...rest}) => {

    return (

        <Route

            {...rest}

            render={({location})=>

            isAuthenticated() ? (

                children

            ) : (

                <Redirect

                    to={{

                        pathname: "/login",

                        state: {from: location}

                    }}

                />

            )

        }

        />

    );

}

export default PrivateRoute;

Created dashboard for user and admin

# For users:

User/dashboard.js:

import Layout from '../Layout';

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

import { userInfo } from '../../utils/auth';

import unnamed from '../img/image/unnamed.jpg'

const Dashboard = () => {

    const { name, email, role, Address, BMDCregNo, ChamberName, Designation, UserStatus, UserType, batchNo, bloodGroup, phoneNo, timeRemain, \_id} = userInfo();

    const UserLinks = () => {

        return (

            <div>

            <div className="card" style={{margin: 50}}>

                <img src={unnamed} height = "600px" width = "600px"/>

            </div>

            <div className="card">

                <ul className="list-group">

                    <li className="list-group-item">

                        <Link className="nav-link" to="#">All News</Link>

                    </li>

                    <li className="list-group-item">

                        <Link className="nav-link" to="#">Update Profile</Link>

                    </li>

                </ul>

            </div>

            </div>

        )

    };

    const UserInfo = () => (

        <div className="card mb-5">

            <h3 className="card-header">User Information</h3>

            <ul className="list-group">

                <li className="list-group-item">ID: {\_id}</li>

                <li className="list-group-item">Name: {name}</li>

                <li className="list-group-item">Email: {email}</li>

                <li className="list-group-item">Role: {role}</li>

                <li className="list-group-item">Address: {Address}</li>

                <li className="list-group-item">BMDC registration No: {BMDCregNo}</li>

                <li className="list-group-item">Chamber Name: {ChamberName}</li>

                <li className="list-group-item">Designation: {Designation}</li>

                <li className="list-group-item">User Status: {UserStatus}</li>

                <li className="list-group-item">User Type: {UserType}</li>

                <li className="list-group-item">Batch No: {batchNo}</li>

                <li className="list-group-item">Blood Group: {bloodGroup}</li>

                <li className="list-group-item">Phone No: {phoneNo}</li>

                <li className="list-group-item">Registration Date: {timeRemain}</li>

            </ul>

        </div>

    );

    return (

        <Layout title="Dashboard" className="container-fluid">

            <div className="row">

                <div className="col-sm-3">

                    <UserLinks />

                </div>

                <div className="col-sm-9">

                    <UserInfo />

                </div>

            </div>

        </Layout>

    )

}

export default Dashboard;

**utils.auth.js:**

import jwt\_decode from 'jwt-decode';

export const authenticate = (token, cb) => {

    if (typeof window !== 'undefined') {

        localStorage.setItem('jwt', JSON.stringify(token));

        cb();

    }

}

export const isAuthenticated = () => {

    if (typeof window === 'undefined') return false;

    if (localStorage.getItem('jwt')) {

        const { exp } = jwt\_decode(JSON.parse(localStorage.getItem('jwt')));

        if ((new Date()).getTime() <= exp \* 1000) {

            return true;

        } else {

            localStorage.removeItem('jwt');

            return false

        }

    } else return false;

}

export const userInfo = () => {

    const jwt = JSON.parse(localStorage.getItem('jwt'));

    const decoded = jwt\_decode(jwt);

    return { ...decoded, token: jwt }

}

export const signout = cb => {

    if (typeof window !== 'undefined') {

        localStorage.removeItem('jwt');

        cb();

    }

}

## ADMIN:

**Create src/api/apiAdmin.js:**

import { API } from '../utils/config';

import axios from 'axios';

export const createActivity = (token, data) => {

    return axios.post(`${API}/activity`, data, {

        headers: {

            "Content-Type": "application/json",

            "Authorization": `Bearer ${token}`

        }

    })

}

export const createDetail = (token, data) => {

    return axios.post(`${API}/detail`, data, {

        headers: {

            "Content-Type": "application/json",

            "Authorization": `Bearer ${token}`

        }

    })

}

export const getActivity = () => {

    return axios.get(`${API}/activity`)

}

## **utils/config:**

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

**create component/admin/AdminDashboard.js:**

import Layout from '../Layout';

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

import { userInfo } from '../../utils/auth';

import unnamed from '../img/image/unnamed.jpg'

const AdminDashboard = () => {

    const { name, email, role, Address, BMDCregNo, ChamberName, Designation, UserStatus, UserType, batchNo, bloodGroup, phoneNo, timeRemain, \_id} = userInfo();

    const UserLinks = () => {

        return (

            <div>

            <div className="card" style={{margin: 50}}>

                <img src={unnamed} height = "600px" width = "600px"/>

            </div>

            <div className="card">

                <h4 className="card-header">User Links</h4>

                <ul className="list-group">

                    <li className="list-group-item">

                        <Link className="nav-link" to="/create/activity">Create Activity</Link>

                    </li>

                    <li className="list-group-item">

                        <Link className="nav-link" to="/create/detail">Create Details</Link>

                    </li>

                    <li className="list-group-item">

                        <Link className="nav-link" to="#">In review users</Link>

                    </li>

                    <li className="list-group-item">

                        <Link className="nav-link" to="#">All users</Link>

                    </li>

                </ul>

            </div>

            </div>

        )

    };

    const UserInfo = () => (

        <div className="card mb-5">

            <h3 className="card-header">User Information</h3>

            <ul className="list-group">

            <li className="list-group-item">ID: {\_id}</li>

                <li className="list-group-item">Name: {name}</li>

                <li className="list-group-item">Email: {email}</li>

                <li className="list-group-item">Role: {role}</li>

                <li className="list-group-item">Address: {Address}</li>

                <li className="list-group-item">BMDC registration No: {BMDCregNo}</li>

                <li className="list-group-item">Chamber Name: {ChamberName}</li>

                <li className="list-group-item">Designation: {Designation}</li>

                <li className="list-group-item">User Status: {UserStatus}</li>

                <li className="list-group-item">User Type: {UserType}</li>

                <li className="list-group-item">Batch No: {batchNo}</li>

                <li className="list-group-item">Blood Group: {bloodGroup}</li>

                <li className="list-group-item">Phone No: {phoneNo}</li>

                <li className="list-group-item">Registration Date: {timeRemain}</li>

            </ul>

        </div>

    );

    return (

        <Layout title="Dashboard" className="container-fluid">

            <div className="row">

                <div className="col-sm-3">

                    <UserLinks />

                </div>

                <div className="col-sm-9">

                    <UserInfo />

                </div>

            </div>

        </Layout>

    )

}

export default AdminDashboard;

**src/protectedRoutes/AdminRoute.js:**

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

import { isAuthenticated, userInfo } from '../../utils/auth';

const AdminRoute = ({ children, ...rest }) => {

    const { role } = userInfo();

    return (

        <Route

            {...rest}

            render={({ location }) =>

                isAuthenticated() && role === 'admin' ? (

                    children

                ) : (

                    <Redirect

                        to={{

                            pathname: "/",

                            state: { from: location }

                        }}

                    />

                )

            }

        />

    );

}

export default AdminRoute;

create **protectRoutes/PrivateRoute.js:**

import { Route, redirect, Redirect } from "react-router";

import {isAuthenticated} from '../../utils/auth';

const PrivateRoute = ({children, ...rest}) => {

    return (

        <Route

            {...rest}

            render={({location})=>

            isAuthenticated() ? (

                children

            ) : (

                <Redirect

                    to={{

                        pathname: "/login",

                        state: {from: location}

                    }}

                />

            )

        }

        />

    );

}

export default PrivateRoute;

**Menu.js:**

import {Link, withRouter} from 'react-router-dom';

import './style/top.css'

import {signout, isAuthenticated, userInfo} from '../utils/auth'

const isActive = (history, path) => {

  if(history.location.pathname === path){

    return {color: '#ff9900'}

  }else{

    return {color: 'white'}

  }

}

const Menu = ({history}) => {

    return(

        <div class="wrapper row0" >

        <div id="topbar" class="hoc clear">

          <div class="fl\_left">

            <ul class="nospace">

              <li><i class="fas fa-phone rgtspace-5"></i> +088 01633749109</li>

              <li><i class="far fa-envelope rgtspace-5"></i> cdcaa.page@gmail.com</li>

            </ul>

          </div>

          <div class="fl\_right" >

            <ul class="nospace">

              <li><i class="fas fa-sign-in-alt"></i>

                    <Link style={isActive(history, '/')} to="/">Home</Link>

              </li>

              {!isAuthenticated() && (<>

              <li><i class="fas fa-edit"></i>

                <Link style={isActive(history, '/login')} to="/login">Login</Link>

              </li>

              </> )}

              {isAuthenticated() && (<>

              <li><i class="fas fa-edit"></i>

                <Link  style = {{cursor: 'pointer', color: 'grey'}} onClick={() => {

                  signout(() => {

                    history.push('/login')

                  });

                }}> Log Out</Link>

              </li>

              </> )}

            </ul>

          </div>

        </div>

        <div class="wrapper row1"  >

    <header id="header" class="hoc clear" >

      <div id="logo" class="fl\_left" >

        <h1><a href="/">CDCAA</a></h1>

      </div>

      <nav id="mainav" class="fl\_right">

        <ul class="clear">

          <li class="active"><a href="/"><b>Home</b></a></li>

          <li><a class="drop" href="#">ORGANIZATION</a>

            <ul>

              <li><a href="pages/gallery.html">Journey of CDCAA</a></li>

              <li><a href="pages/full-width.html">Excutive Committee</a></li>

              <li><a href="pages/sidebar-left.html">Sub Committee</a></li>

              <li><a href="pages/sidebar-right.html">Past Leaders</a></li>

            </ul>

          </li>

          <li><a class="drop" href="#">Activities</a>

            <ul>

              <li><a href="#">Soical Responsibilites</a></li>

            </ul>

          </li>

        {isAuthenticated() && (<>

          <li>

          <Link style={isActive(history, `/${userInfo().role}/dashboard`)} to={`/${userInfo().role}/dashboard`}>Dashboard</Link>

          </li>

          </>)}

          <li><a href="#">NEWS</a></li>

          <li><a href="#">NOTICE</a></li>

          <li><a href="C:\Users\Asus\Desktop\City Dental\CDCAA\pages\gallery.html">GALLERY</a></li>

          <li><a href="#">CONTACT US</a></li>

        </ul>

      </nav>

    </header>

    </div>

      </div>

)

}

export default withRouter(Menu);

**Main.js:**

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

import Home from './home/Home'

import Register from './user/Register';

import Login from './user/login'

import PrivateRoute from '../components/protectRoutes/PrivateRoute';

import Dashboard from './user/Dashboard'

import AdminRoute from '../components/protectRoutes/AdminRoute';

import CreateActivity from './admin/CreateActivity';

import CreateDetail from './admin/CreateDetail';

import AdminDashboard from './admin/AdminDashboard';

const Main = () => {

    return (

    <div>

        <Switch>

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

            <Route path = "/login" exact component={Login} />

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

            <PrivateRoute exact path="/user/dashboard">

                    <Dashboard />

            </PrivateRoute>

            <AdminRoute exact path="/admin/dashboard">

                <AdminDashboard />

            </AdminRoute>

            <AdminRoute exact path="/create/activity">

                <CreateActivity />

            </AdminRoute>

            <AdminRoute exact path="/create/detail">

                <CreateDetail />

            </AdminRoute>

        </Switch>

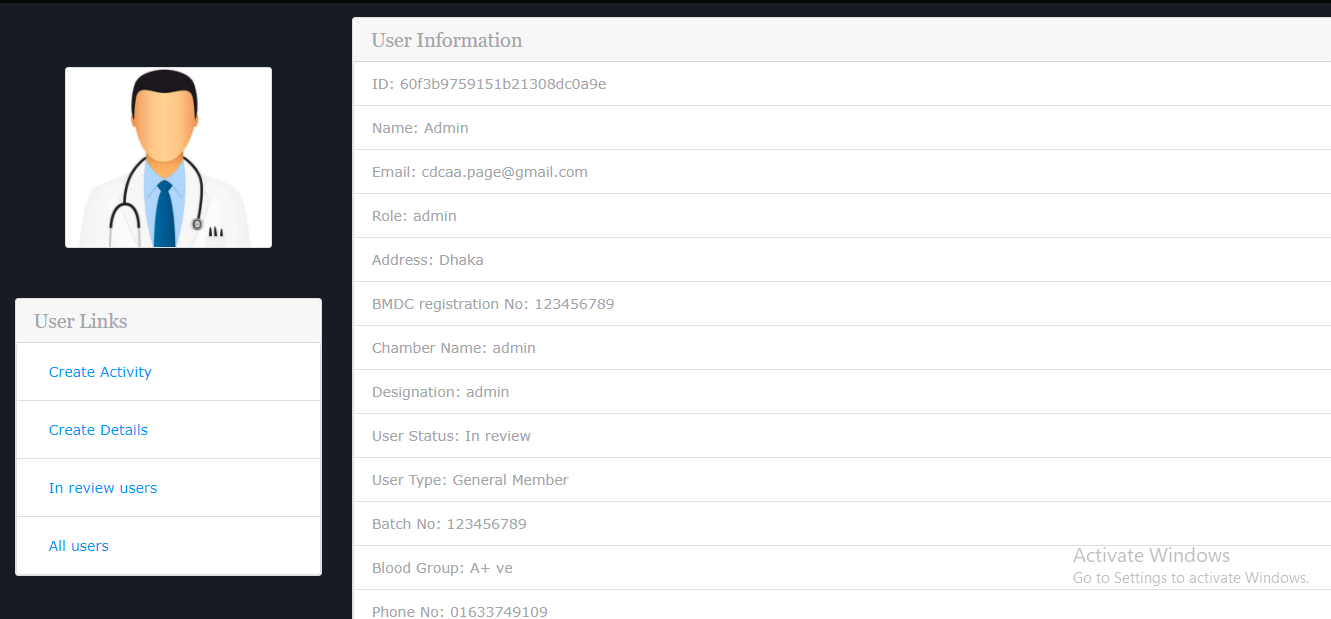
    </div>

    )

}

export default Main;

Fetching all users for admin dashboard



1st: request in postman: localhost:3001/api/allusers

and get all the users.

# In backend:

userControllers.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(req.body, ['name', 'email', 'batchNo', 'BMDCregNo', 'phoneNo','bloodGroup', 'ChamberName', 'Designation', 'Address', 'password','UserStatus','timeRemain', '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"])

    });

}

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

    const users = await User.find()

        .sort({name: 1});

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

}

userRouter.js:

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

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

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

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

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

router.route('/allusers')

    .get([authorize, admin], getUsers);

module.exports = router;

Frontend

**Admin/allusers.js:**

import { useState, useEffect } from 'react';

import Layout from '../Layout';

import Card from './Card';

//import CheckBox from './CheckBox';

//import RadioBox from './RadioBox';

//import { showError, showSuccess } from '../../utils/messages';

//import { getCategories, getProducts, getFilteredProducts } from '../../api/apiProduct';

import { getUsers } from '../../api/apiUsers';

import { isAuthenticated, userInfo } from '../../utils/auth';

const AllUsers = () => {

    const [users, setUsers] = useState([]);

    useEffect(() => {

        getUsers()

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

            .catch();

    }, [])

    return (

        <Layout title="Home Page" className="container-fluid">

            <div style={{ width: "100%" }}>

            </div>

            <div className="row">

                {users && users.map(user => <Card user={user} key={user.\_id}

                    />)}

            </div>

        </Layout>

    )

}

export default AllUsers;

**card.js:**

import React from 'react';

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

import { API } from '../../utils/config';

import unnamed from "../img/image/unnamed.jpg"

const Card = ({ user }) => {

    const titleStyle = {

        display: "block",

        textOverflow: "ellipsis",

        wordWrap: "break-word",

        overflow: "hidden",

        maxHeight: "2em",

        lineHeight: "1em"

    }

    const imgStyle = {

        height: 250,

        objectFit: "cover",

        objectPosition: "0px 0px"

    }

    return (

        <div className="col-md-3 col-sm-4 col-xs-12 mb-3">

            <div className="card">

                <img

                    src={unnamed}

                    alt={user.name}

                    style={imgStyle}

                    className="card-img-top"

                />

                <div className="card-body">

                    <div style={{ minHeight: "3em" }}>

                        <p style={titleStyle}>{user.name}</p>

                    <Link to={`/allusers/${user.\_id}`}>

                        <button className="btn btn-outline-warning btn-sm">View Details</button>

                    </Link>

                    </div>

                </div>

            </div>

        </div>

    )

}

export default Card;

**in main.js:**

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

import Home from './home/Home'

import Register from './user/Register';

import Login from './user/login'

import PrivateRoute from '../components/protectRoutes/PrivateRoute';

import Dashboard from './user/Dashboard'

import AdminRoute from '../components/protectRoutes/AdminRoute';

import CreateActivity from './admin/CreateActivity';

import CreateDetail from './admin/CreateDetail';

import AllUsers from './admin/allUsers';

import AdminDashboard from './admin/AdminDashboard';

const Main = () => {

    return (

    <div>

        <Switch>

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

            <Route path = "/login" exact component={Login} />

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

            <PrivateRoute exact path="/user/dashboard">

                    <Dashboard />

            </PrivateRoute>

            <AdminRoute exact path="/admin/dashboard">

                <AdminDashboard />

            </AdminRoute>

            <AdminRoute exact path="/create/activity">

                <CreateActivity />

            </AdminRoute>

            <AdminRoute exact path="/create/detail">

                <CreateDetail />

            </AdminRoute>

            <AdminRoute exact path="/allusers">

                <AllUsers />

            </AdminRoute>

        </Switch>

    </div>

    )

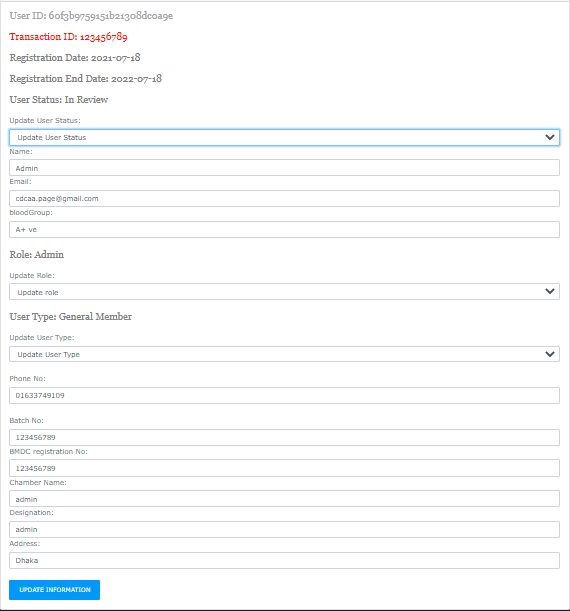
}

export default Main;

**ShippingAddress ta view user a use kora. Backend implemented.**

localhost:3001/api/user/allusers/60f4088fdc89013240d57355

Now admin panel all user’s info + update



In backend:

**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(req.body, ['name', 'email', 'batchNo', 'BMDCregNo', 'phoneNo','bloodGroup', 'ChamberName', 'Designation', 'Address', 'password','UserStatus','timeRemain', '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"])

    });

}

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

    const userId = req.params.id;

    const user = await User.findById(userId)

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

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

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

}

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

    const users = await User.find()

        .sort({name: 1});

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

}

Define the route:

**userRouter.js**

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

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

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

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

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

router.route('/allusers')

    .get(getUsers);

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

    .get(getUserById);

module.exports = router;

**admin/UniqueUser.js:**

import React, {userState, useEffect, useState} from 'react';

import Layout from '../Layout';

import { getUserDetails, updateProfile } from '../../api/apiUsers'

import { userInfo } from '../../utils/auth';

const UniqueUser = () => {

    let str = window.location.href;

    const myArr = str.split('/');

    const [values, setValues] = useState({

        \_id: '',

        name: '',

        email: '',

        batchNo: '',

        BMDCregNo: '',

        phoneNo: '',

        bloodGroup: '',

        ChamberName: '',

        Designation: '',

        Address: '',

        TransactionID: '',

        role: '',

        UserType: '',

        UserStatus: '',

        timeRemain: ''

    });

    const [disabled, setDisabled] = useState(false);

    const [redirect, setRedirect] = useState(false);

    const {

        \_id,

        name,

        email,

        batchNo,

        BMDCregNo,

        phoneNo,

        bloodGroup,

        ChamberName,

        Designation,

        Address,

        TransactionID,

        role,

        UserType,

        UserStatus,

        timeRemain,

    } = values;

    useEffect(() => {

        getUserDetails(myArr[4])

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

            .catch(err => {

            })

    }, []);

    const handleChange = (e) => {

        setValues({

            ...values,

            [e.target.name]: e.target.value,

        })

    }

    const handleSubmit = (e) => {

        e.preventDefault();

        setDisabled(true);

        updateProfile(myArr[4], values)

            .then(response => {

                if(response.status === 200){

                setRedirect(true);

                }

            })

            .catch(err => setDisabled(false));

    }

    var d = {timeRemain};

    const finish = d.timeRemain.split('-');

    let finishYear = `${parseInt(finish[0])+1}-${finish[1]}-${finish[2]}`;

const profileForm = () => (<form onSubmit={handleSubmit}>

        <h3 style={{textStyle: 'bold'}}>User ID: {\_id}</h3>

        <h3 style={{color: 'red'}}>Transaction ID: {TransactionID}</h3>

        <h3 style={{color: '#808080'}}>Registration Date: {timeRemain}  </h3>

        <h3 style={{color: '#808080'}}>Registration End Date: {finishYear}  </h3>

        <h3 style={{color: '#808080'}}>User Status: {UserStatus}  </h3>

        <label className="text-muted">Update User Status: </label>

        <select name={UserStatus}

        onChange={handleChange} className="form-control">

                <option>Update User Status</option>

                <option value="In review">In review</option>

                <option value="Active">Active</option>

                <option value="Renew">Renew</option>

        </select>

        <label className="text-muted">Name: </label>

        <input type="text" name={name} value={name} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Email: </label>

        <input type="email" name={email} value={email} required className="form-control" onChange={handleChange} />

        <label className="text-muted">bloodGroup: </label>

        <input type="text" name={bloodGroup} value={bloodGroup} required className="form-control" onChange={handleChange} />

        <br/>

        <h3 style={{color: '#808080'}}>Role: {role}  </h3>

        <label className="text-muted">Update Role: </label>

        <select name={role}

        onChange={handleChange} className="form-control">

                <option>Update role</option>

                <option value="user">user</option>

                <option value="admin">admin</option>

        </select>

        <br/>

        <h3 style={{color: '#808080'}}>User Type: {UserType}  </h3>

        <label className="text-muted">Update User Type: </label>

        <select name={UserType}

        onChange={handleChange} className="form-control">

                <option>Update User Type</option>

                <option value="Life Member">Life Member</option>

                <option value="General Member">General Member</option>

        </select>

        <br/>

        <label className="text-muted">Phone No: </label>

        <input type="number" name={phoneNo}  value={phoneNo} required className="form-control" onChange={handleChange} />

        <br/>

        <label className="text-muted">Batch No: </label>

        <input type="text" name={batchNo} value={batchNo} required className="form-control" onChange={handleChange} />

        <label className="text-muted">BMDC registration No: </label>

        <input type="text" name={BMDCregNo} value={BMDCregNo} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Chamber Name: </label>

        <input type="text" name={ChamberName} value={ChamberName} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Designation: </label>

        <input type="text" name={Designation} value={Designation} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Address: </label>

        <input type="text" name={Address} value={Address} required className="form-control" onChange={handleChange} />

        <br />

        <button type="submit" className="btn btn-primary btn-sm" disabled={disabled}>Update Information</button>

    </form>)

    return (<>

        <Layout title="Profile" description="Doctor's Profile" className="container">

            <div className="container">

                <div className="row">

                    <div className="col-md-12">

                    <a href="/admin/dashboard">Go to Dashboard</a> <br />

                            <a href="/allusers">Go to AllUsers</a>

                        <div className="card mb-5" style={{height: 'auto'}}>

                            <div className="card-header">Details</div>

                                <div className="card-body">

                                    {profileForm()}

                                </div>

                        </div>

                    </div>

                </div>

            </div>

        </Layout>

    </>);

}

export default UniqueUser;

**card.js:**

import React from 'react';

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

import { API } from '../../utils/config';

import unnamed from "../img/image/unnamed.jpg"

const Card = ({ user }) => {

    const titleStyle = {

        display: "block",

        textOverflow: "ellipsis",

        wordWrap: "break-word",

        overflow: "hidden",

        maxHeight: "2em",

        lineHeight: "1em"

    }

    const imgStyle = {

        height: 250,

        objectFit: "cover",

        objectPosition: "0px 0px"

    }

    return (

        <div className="col-md-3 col-sm-4 col-xs-12 mb-3">

            <div className="card">

                <img

                    src={unnamed}

                    alt={user.name}

                    style={imgStyle}

                    className="card-img-top"

                />

                <div className="card-body">

                    <div style={{ minHeight: "3em" }}>

                        <p style={titleStyle}>{user.name}</p>

                        <p style={titleStyle}>{user.UserStatus}</p>

                    <Link to={`/allusers/${user.\_id}`}>

                        <button className="btn btn-outline-warning btn-sm">View Details</button>

                    </Link>

                    </div>

                </div>

            </div>

        </div>

    )

}

export default Card;

api/apiUsers.js:

import { API } from '../utils/config';

import axios from 'axios';

export const getUsers = () => {

    return axios.get(`${API}/user/allusers`)

}

export const getUserDetails = (id) => {

    return axios.get(`${API}/user/allusers/${id}`)

}

Now writing codes for updating user profile:

**userControllers.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(req.body, ['name', 'email', 'batchNo', 'BMDCregNo', 'phoneNo','bloodGroup', 'ChamberName', 'Designation', 'Address', 'password','UserStatus','timeRemain', '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"])

    });

}

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

    const userId = req.params.id;

    const user = await User.findById(userId)

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

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

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

}

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

    const users = await User.find()

        .sort({createdAt: -1});

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

}

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

    const {\_id,name,email,batchNo,BMDCregNo,phoneNo,bloodGroup,ChamberName,Designation,Address,role,UserType,UserStatus} = \_.pick (req.body, ["\_id",

        "name",

        "email",

        "batchNo",

        "BMDCregNo",

        "phoneNo",

        "bloodGroup",

        "ChamberName",

        "Designation",

        "Address",

        "role",

        "UserType",

        "UserStatus"

        ]);

        await User.updateMany({ \_id: \_id}, {name: name ,email: email,batchNo: batchNo,BMDCregNo: BMDCregNo,phoneNo: phoneNo,bloodGroup: bloodGroup,ChamberName: ChamberName,Designation: Designation,Address: Address,role: role,UserType: UserType,UserStatus: UserStatus});

        return res.status(200).send("Information updated!");

}

**Routes/userRouters.js:**

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

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

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

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

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

router.route('/allusers')

    .get(getUsers);

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

    .get(getUserById)

    .put(updateUserProfile);

module.exports = router;

**frontend**:

**src/api/apiUsers.js:**

import { API } from '../utils/config';

import axios from 'axios';

export const getUsers = () => {

    return axios.get(`${API}/user/allusers`)

}

export const getUserDetails = (id) => {

    return axios.get(`${API}/user/allusers/${id}`)

}

export const updateProfile = (token, data) => {

    return axios.put(`${API}/user/allusers/${data.\_id}`, data, {

        headers: {

            "Content-Type": "application/json",

            "Authorization": `Bearer ${token}`

        }

    })

}

uniqueUsers.js:

import React, {userState, useEffect, useState} from 'react';

import Layout from '../Layout';

import { getUserDetails, updateProfile } from '../../api/apiUsers'

import { userInfo } from '../../utils/auth';

const UniqueUser = () => {

    let str = window.location.href;

    const myArr = str.split('/');

    const [values, setValues] = useState({

        \_id: '',

        name: '',

        email: '',

        batchNo: '',

        BMDCregNo: '',

        phoneNo: '',

        bloodGroup: '',

        ChamberName: '',

        Designation: '',

        Address: '',

        TransactionID: '',

        role: '',

        UserType: '',

        UserStatus: '',

        timeRemain: ''

    });

    const [disabled, setDisabled] = useState(false);

    const [redirect, setRedirect] = useState(false);

    const {

        \_id,

        name,

        email,

        batchNo,

        BMDCregNo,

        phoneNo,

        bloodGroup,

        ChamberName,

        Designation,

        Address,

        TransactionID,

        role,

        UserType,

        UserStatus,

        timeRemain,

    } = values;

    useEffect(() => {

        getUserDetails(myArr[4])

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

            .catch(err => {

            })

    }, []);

    const handleChange = (e) => {

        setValues({

            ...values,

            [e.target.name]: e.target.value,

        })

    }

    const handleSubmit = (e) => {

        e.preventDefault();

        setDisabled(true);

        updateProfile(userInfo().token, values)

            .then(response => {

                if(response.status === 200){

                setRedirect(true);

                alert("Information Updated!");

                window.location.reload();

                }

            })

            .catch(err => setDisabled(false));

    }

    var d = {timeRemain};

    const finish = d.timeRemain.split('-');

    let finishYear = `${parseInt(finish[0])+1}-${finish[1]}-${finish[2]}`;

    const profileForm = () => (<form onSubmit={handleSubmit}>

        <h3 style={{textStyle: 'bold'}}>User ID: {\_id}</h3>

        <h3 style={{color: 'red'}}>Transaction ID: {TransactionID}</h3>

        <h3 style={{color: '#808080'}}>Registration Date: {timeRemain}  </h3>

        <h3 style={{color: '#808080'}}>Registration End Date: {finishYear}  </h3>

        <h3 style={{color: '#808080'}}>User Status: {UserStatus}  </h3>

        <label className="text-muted">Update User Status: </label>

        <select name= "UserStatus"

        onChange={handleChange} className="form-control">

                <option>Update User Status</option>

                <option value="In review">In review</option>

                <option value="Active">Active</option>

                <option value="Renew">Renew</option>

        </select>

        <label className="text-muted">Name: </label>

        <input type="text" name="name" value={name} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Email: </label>

        <input type="email" name="email" value={email} required className="form-control" onChange={handleChange} />

        <label className="text-muted">bloodGroup: </label>

        <input type="text" name="bloodGroup" value={bloodGroup} required className="form-control" onChange={handleChange} />

        <br/>

        <h3 style={{color: '#808080'}}>Role: {role}  </h3>

        <label className="text-muted">Update Role: </label>

        <select name="role"

        onChange={handleChange} className="form-control">

                <option>Update role</option>

                <option value="user">user</option>

                <option value="admin">admin</option>

        </select>

        <br/>

        <h3 style={{color: '#808080'}}>User Type: {UserType}  </h3>

        <label className="text-muted">Update User Type: </label>

        <select name="UserType"

        onChange={handleChange} className="form-control">

                <option>Update User Type</option>

                <option value="Life Member">Life Member</option>

                <option value="General Member">General Member</option>

        </select>

        <br/>

        <label className="text-muted">Phone No: </label>

        <input type="number" name="phoneNo"  value={phoneNo} required className="form-control" onChange={handleChange} />

        <br/>

        <label className="text-muted">Batch No: </label>

        <input type="text" name="batchNo" value={batchNo} required className="form-control" onChange={handleChange} />

        <label className="text-muted">BMDC registration No: </label>

        <input type="text" name="BMDCregNo" value={BMDCregNo} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Chamber Name: </label>

        <input type="text" name="ChamberName" value={ChamberName} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Designation: </label>

        <input type="text" name="Designation" value={Designation} required className="form-control" onChange={handleChange} />

        <label className="text-muted">Address: </label>

        <input type="text" name="Address" value={Address} required className="form-control" onChange={handleChange} />

        <br />

        <button type="submit" className="btn btn-primary btn-sm" disabled={disabled}>Update Information</button>

    </form>)

    return (<>

        <Layout title="Profile" description="Doctor's Profile" className="container">

            <div className="container">

                <div className="row">

                    <div className="col-md-12">

                    <a href="/admin/dashboard">Go to Dashboard</a> <br />

                            <a href="/allusers">Go to AllUsers</a>

                        <div className="card mb-5" style={{height: 'auto'}}>

                            <div className="card-header">Details</div>

                                <div className="card-body">

                                    {profileForm()}

                                </div>

                        </div>

                    </div>

                </div>

            </div>

        </Layout>

    </>);

}

export default UniqueUser;

Delete user using ID:

In backend:

**userControllers.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(req.body, ['name', 'email', 'batchNo', 'BMDCregNo', 'phoneNo','bloodGroup', 'ChamberName', 'Designation', 'Address', 'password','UserStatus','timeRemain', '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"])

    });

}

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

    const userId = req.params.id;

    const user = await User.findById(userId)

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

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

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

}

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

    const users = await User.find()

        .sort({createdAt: -1});

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

}

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

    const {\_id,name,email,batchNo,BMDCregNo,phoneNo,bloodGroup,ChamberName,Designation,Address,role,UserType,UserStatus} = \_.pick (req.body, ["\_id",

        "name",

        "email",

        "batchNo",

        "BMDCregNo",

        "phoneNo",

        "bloodGroup",

        "ChamberName",

        "Designation",

        "Address",

        "role",

        "UserType",

        "UserStatus"

        ]);

        await User.updateMany({ \_id: \_id}, {name: name ,email: email,batchNo: batchNo,BMDCregNo: BMDCregNo,phoneNo: phoneNo,bloodGroup: bloodGroup,ChamberName: ChamberName,Designation: Designation,Address: Address,role: role,UserType: UserType,UserStatus: UserStatus});

        return res.status(200).send("Information updated!");

}

//delete user using ID

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

    const \_id = req.params.id;

    await User.deleteOne({\_id: \_id});

    return res.status(200).send("Deleted!");

}

userRouter.js:

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

const {signIn, signUp, getUsers, getUserById, updateUserProfile, deleteUser} = require('../controllers/userControllers')

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

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

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

router.route('/allusers')

    .get(getUsers);

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

    .get(getUserById)

    .put(updateUserProfile)

    .delete(deleteUser);

module.exports = router;

Add photo to profile page:

Make a collection for **photo**. Unique \_id, user\_id (from JWT), photo.

ISSUE: if photo is uploaded or not (check)

Backend:

**Model/photo.js:**

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

const Joi = require('joi');

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

    user\_id: {

        type: Schema.Types.ObjectId,

        ref: 'Activity',

        require: true

    },

    photo: {

        data: Buffer,

        contentType: String,

    }

}, {timestamps: true}));

module.exports.validate1 = image => {

    const schema = Joi.object({

        user\_id: Joi.string().required(),

    });

    return schema.validate(image);

}

In userController.js:

const bcrypt = require('bcrypt');

const \_ = require('lodash');

const fs = require('fs');

const formidable = require ('formidable');

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

const {Image, validate1} = require('../model/photo');

module.exports.createImage = 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 } = validate1(\_.pick(fields, ["user\_id"]));

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

        const image = new Image(fields);

        if(files.photo) {

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

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

                image.photo.data = data;

                image.photo.contentType = files.photo.type;

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

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

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

                        message: "Image Created Successfully",

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

                    })

                })

            })

        }else{

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

        }

    })

}

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','UserStatus','timeRemain', '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"])

    });

}

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

    const userId = req.params.id;

    const user = await User.findById(userId)

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

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

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

}

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

    const users = await User.find()

        .sort({createdAt: -1});

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

}

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

    const {\_id,name,email,batchNo,BMDCregNo,phoneNo,bloodGroup,ChamberName,Designation,Address,role,UserType,UserStatus} = \_.pick (req.body, ["\_id",

        "name",

        "email",

        "batchNo",

        "BMDCregNo",

        "phoneNo",

        "bloodGroup",

        "ChamberName",

        "Designation",

        "Address",

        "role",

        "UserType",

        "UserStatus"

        ]);

        await User.updateMany({ \_id: \_id}, {name: name ,email: email,batchNo: batchNo,BMDCregNo: BMDCregNo,phoneNo: phoneNo,bloodGroup: bloodGroup,ChamberName: ChamberName,Designation: Designation,Address: Address,role: role,UserType: UserType,UserStatus: UserStatus});

        return res.status(200).send("Information updated!");

}

//delete user using ID

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

    const \_id = req.params.id;

    await User.deleteOne({\_id: \_id});

    return res.status(200).send("Deleted!");

}

In userRouter.js:

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

const {signIn, signUp, getUsers, getUserById, updateUserProfile, deleteUser, createImage} = require('../controllers/userControllers')

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

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

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

router.route('/allusers')

    .get(getUsers);

router.route('/photos')

    .post([authorize],createImage);

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

    .get(getUserById)

    .put(updateUserProfile)

    .delete(deleteUser);

module.exports = router;

**now get Image:**

**userRouter.js:**

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

const {signIn,

     signUp,

      getUsers,

      getUserById,

      updateUserProfile,

      deleteUser,

      createImage,

      getPhoto} = require('../controllers/userControllers')

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

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

router.route('/signup')

    .post(signUp);

router.route('/signin')

    .post(signIn);

router.route('/allusers')

    .get(getUsers);

router.route('/photos')

    .post([authorize],createImage);

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

    .get(getPhoto);

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

    .get(getUserById)

    .put(updateUserProfile)

    .delete(deleteUser);

module.exports = router;

userControllers.js:

const bcrypt = require('bcrypt');

const \_ = require('lodash');

const fs = require('fs');

const formidable = require ('formidable');

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

const {Image, validate1} = require('../model/photo');

var morgan = require('morgan')

module.exports.createImage = 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 } = validate1(\_.pick(fields, ["user\_id"]));

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

        const image = new Image(fields);

        if(files.photo) {

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

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

                image.photo.data = data;

                image.photo.contentType = files.photo.type;

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

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

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

                        message: "Image Created Successfully",

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

                    })

                })

            })

        }else{

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

        }

    })

}

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

    const photoId = req.params.id;

    const user = await Image.findById(photoId)

        .select({ photo: 1, \_id: 0})

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

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

}

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','UserStatus','timeRemain', '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"])

    });

}

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

    const userId = req.params.id;

    const user = await User.findById(userId)

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

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

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

}

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

    const users = await User.find()

        .sort({createdAt: -1});

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

}

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

    const {\_id,name,email,batchNo,BMDCregNo,phoneNo,bloodGroup,ChamberName,Designation,Address,role,UserType,UserStatus} = \_.pick (req.body, ["\_id",

        "name",

        "email",

        "batchNo",

        "BMDCregNo",

        "phoneNo",

        "bloodGroup",

        "ChamberName",

        "Designation",

        "Address",

        "role",

        "UserType",

        "UserStatus"

        ]);

        await User.updateMany({ \_id: \_id}, {name: name ,email: email,batchNo: batchNo,BMDCregNo: BMDCregNo,phoneNo: phoneNo,bloodGroup: bloodGroup,ChamberName: ChamberName,Designation: Designation,Address: Address,role: role,UserType: UserType,UserStatus: UserStatus});

        return res.status(200).send("Information updated!");

}

//delete user using ID

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

    const \_id = req.params.id;

    await User.deleteOne({\_id: \_id});

    return res.status(200).send("Deleted!");

}

**START**: front end….main project: AdminDashboard and copy from CreateProduct.js handleChange and handleSubmit