**PART – 03 – Creating Simple Server by USING Express js**

**Step (1):** Create a project folder with any name (suppose ‘learnmernstack’) than inside this folder create another folder with any name(suppose ‘server’) for backend.

**Step (2):** Open terminal and goto insise server folder and run below 2 command:

npm init -y // To install package

npm i express // To install express js

**Step(3):** Inside ‘server’ folder create a new file with any name (suppose server.js)

Open server.js file and paste below code:

const express = require("express");

const app = express();

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

    res.status(200).send('welcome to mern series');

});

app.get("/register",(req,res)=>{

    res.status(200).send('welcome to register in mern series');

});

const PORT = 5000;

app.listen(PORT,()=>{

    console.log(`server is running at PORT:${PORT}`);

});

**Step(4):** We can run server.js file by using below command:

npm run start or npm run dev

Note:

1. if we will do any changes in server.js file then we need to run again this file.
2. To check in browser hit below url:

localhost:5000 or localhost:5000/register

**Step(5):** If we want to auto update and run server.js file then we can install nodemon inside server folder by using below command. it will update automatically in background.

npm i nodemon

Note:

1. Now we will use below command inside server folder to run server.js file:

nodemon server.js

1. Open package.json file and add below code with in “scripts” key after giving comma(,)

"start" : "nodemon server.js"

For npm run strart

OR

"dev" : "nodemon server.js"

For npm run dev

**PART - 04** - **Creating Routes**

**Step(6)**  Create a folder ‘router’ inside server folder than create a file ‘auth-router.js’ inside router folder.

Open auth-router.js file paste below code:

const express = require("express");

const router = express.Router();

// router.get("/",(req,res)=>{

//      res.status(200).send('welcome to mern series using Routers');

// });

// OR

router.route("/").get((req,res)=>{

res.status(200).send('welcome to mern series using Routers mm');

});

module.exports = router;

**Step(7)** Open server.js file and add below code after “const app = express();”

const router = require("./router/auth-router");

app.use("/api/auth",router);

comment below code of server.js file

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

res.status(200).send('welcome to mern series');

});

app.get("/register",(req,res)=>{

res.status(200).send('welcome to register in mern series');

});

Open terminal >> go to inside server folder by cd command >> run below command:

**npm run start**

then open browser and hit below url:

[**http://localhost:5000/api/auth**](http://localhost:5000/api/auth)

**output:** welcome to mern series using Routers

**PART - 05** – **Controllers in Express js**

**Step(8)** Create a folder name of **controllers** inside server folder,than inside controllers folder, create **auth-controller.js** file inside controllers folder.

Paste below codes inside auth-controller.js file:

const home = async (req,res) => {

    try

    {

        res.status(200).send('welcome to mern series using Routers home');

    }

    catch(error)

    {

        console.log(error);

    }

}

 const register = async (req,res)=>{

    try

    {

        res.status(200).send('welcome to Registration page using controller');

    }

    catch(error)

    {

        res.status(400).send({msg:"Page not Found"});

    }

 }

 module.exports = { home,register }

open auth-router.js file and follow below instruction:

1. Add below code after const router = express.Router();

const authcontrollers = require("../controllers/auth-controller.js");

1. Comment below code:

router.route("/").get((req,res)=>{

res.status(200).send('welcome to mern series using Routers mm');

});

1. Add below codes before module.exports = router;

router.route("/home").get(authcontrollers.home);

router.route('/register').get(authcontrollers.register);

**Step(9)** open server.js and follow below instructins:

To run execute below command:

**npm run start**

than hit below path on browser:

http://localhost:5000/api/auth/home

**Note:**

1. If required then install **npm i controller**
2. If we want to use class then paste below code in auth-controller.js file and comment existing code:

class AccountController {

    signUp = (req, res) => {

        res.send("this is signup method");

    };

    signIn = (req, res) => {

        res.send("this is signin method");

    }

}

module.exports = new AccountController();

than paste below code in auth-router.js file and comment existing code:

const express = require("express");

const router = new express.Router();

const AccountController = require("../controllers/auth-controller.js");

router.get('/signup', AccountController.signUp);

router.get('/signin', AccountController.signIn);

module.exports = router;

**PART - 06** – **POSTMAN**

login postman by below link:

[www.postman.com](http://www.postman.com)

goto home >> workspaces >> create workspace >> Next >> put workspace name suppose: “testmern” >> Create >> Create collection >> Add a request >> Put request name suppose: “register” >> copy the localhost url and paste here with **GET** method >> click on **Send** >> we can see output in below section.

1. Open server.js file and add below code after router variable:

app.use(express.json());

1. Open auth-router.js file and update below code:

router.route('/register').post(authcontrollers.register);

instead of

router.route('/register').get(authcontrollers.register);

1. Open auth-controller.js file and update below code:

 const register = async (req,res)=>{

    try

  {

      console.log(req.body);

      res.status(200).json({message:req.body});

    }

    catch(error)

    {

       res.status(400).json({msg:"internal server error"});

    }

 }

Instead of register function.

1. Go to postman:

Select **POST** mehod and paste below localhot url (like: <http://localhost:5000/api/auth/register>).

>>under Headers tab, create a key **Content-Type** and value is **application/json** .

>>under Body tab, select **raw** and under dropdown select **JSON**

>> paste below code in text area

{  "message":"welcome" }

>> click on send button, we can see the output in below section and also we can see output in vs code under terminal tab

Note:

1. For single row we use below syntax:

   {

        "fullname" : "vijay",

        "email" : "c@gmail.com",

        "mobileno" : 998989898,

        "password" : "yhkhkhkhk"

    }

1. For multiple rows use below syntax:

[

    {

        "fullname" : "vijay",

        "email" : "c@gmail.com",

        "mobileno" : 998989898,

        "password" : "yhkhkhkhk"

    },

    {

        "fullname" : "vijay",

        "email" : "c@gmail.com",

        "mobileno" : 998989898,

        "password" : "yhkhkhkhk"

    }

]

**PART - 07** – **Connecting Mongodb**

Install mongodb on pc. Or we can use mongodb atlas (cloud based)

Goto vs code >> drop down (on bottom right of terminal)>> PowerShell >> go to server folder by cd command >>

Run below command:

**npm i mongoose**

create a folder **utils** with in server folder and create a file **db.js** with in utils folder.

Paste below code with in db.js file:

const mongoose = require("mongoose");

const URI="mongodb+srv://videshi:12345@cluster0.jerkd8t.mongodb.net/";

//this is mongodb atlas url

const connectDb = async () =>{

    try{

        await mongoose.connect(URI);

        console.log("Database connected")

    }

    catch(error)

    {

        console.error("database connection failed " + error);

        process.exit(0);

    }

}

module.exports = connectDb;

Note:

1. In URI variable we have used mongo db atlas (cloud mongo db) connection string, where **videshi** is user name and **12345** is password. Search and Open mongodb atlas than sign in,

goto database >> connection >> Drivers >> copy the connection string and update username and password than use it.

1. We can also use local mongodb compass connection string from local server.
2. If getting port issue than we can change port number.

Open server.js file and add below code after router variable.

const connectDb = require("./utils/db");

and update below code:

connectDb().then(()=>{

    app.listen(PORT,()=>{

        console.log(`server is running at PORT:${PORT}`);

    });

}).catch((error)=>{alert(error)});

Instead of

app.listen(PORT,()=>{

console.log(`server is running at PORT:${PORT}`);

});

**PART - 08** – **Secure private data**

Open terminal >> go to inside server folder >> run below command:

**npm i dotenv**

1. In VS code , install **Dotenv Official +Vault** through extension icon.

Note: in .env file our data will hide.

1. create a file **.env** with in server folder and paste below code with in .env file:

MONGODB\_URI = mongodb+srv://videshi:12345@cluster0.jerkd8t.mongodb.net/mern\_admin

1. Open db.js file and update below code:

const URI = process.env.MONGODB\_URI;

instead of

const URI="mongodb+srv://videshi:12345@cluster0.jerkd8t.mongodb.net/";

1. Open server.js file and add below code before express variable(i.e on the top):

require("dotenv").config();

1. If getting error than restart the server.

**PART - 09** – **Creating user schema and model**

Create a folder **models** with in server folder and create a file **user-model.js** with in model folder.

Open **user-model.js** file and paste below code:

const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({

    username:{

        type:String,

        require:true,

    },

    email:{

        type:String,

        require:true,

    },

    phone:{

        type:String,

        require:true,

    },

    password:{

        type:String,

        require:true,

    },

    isAdmin:{

        type:Boolean,

        default:false,

    }

});

// define model name or collection name

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

module.exports = User;

// where RegisteredUser is table name it will show name of **registeredusers**

Note: in database, it will automatic create a table name of **registeredusers and username,email,phone,password and isAdmin are colum names.**

**PART - 10** – **Store data in database**

Run below command with in server folder:

npm i bcryptjs

Open **auth-controller.js** file and add below code on the top:

const User = require("../models/user-model");

than add below code after User variable:

const bcrypt = require("bcryptjs");

than replace the code of register function with below code:

const register = async (req,res)=>{

    try

    {

        console.log("User details",req.body);

        const{username,email,phone,password} = req.body;

        const userExist = await User.findOne({email:email});

        if(userExist)

        {

           res.status(400).json({msg:"Email already exist !!!"});

return;

        }

else

{

        const saltRound = 10;

        const has\_password = await bcrypt.hash(password,saltRound);

        console.log("haspwd",has\_password);

        const data = {

            username:username,email:email,phone:phone,password:has\_password

        }

        console.log("data",data)

        const userCreated = await User.create(data)

        .then((resp)=>{

            console.log("Response",resp);

            return res.status(200).json({msg:"Registration sussessfull !!!"});

        }).catch((err)=>{console.log("error",err)});

      }

    }

    catch(error)

    {

       console.log("Error",error)

       res.status(400).json({msg:"internal server error"});

    }

 }

Note:

1. Table or Collection
2. Row or Document
3. Colum or Field

**PART – 11-Secure password by bcrypt.js**

**[ Already included in PART-10 ]**

**PART – 12-JSON Web Token (**for authentication and authentication**)**

Open terminal and Run below command inside server folder(backend folder of project):

npm i jsonwebtoken

Open auth-controller.js file and replace below code instead of **userCreated** variable(including it’s value) with in try block of register function:

const userCreated = await User.create(data);

            if(userCreated)

            {

                res.status(202).json({ msg: "Registration Successful",getdata: userCreated, token: await userCreated.generateToken(), userId: userCreated.\_id.toString() });

            }

            else

            {

                res.status(400).json({msg:"Registration Failed"});

            }

Open .env file and declare new variable or paste below code:

JWT\_SECRET\_KEY = WORDBESTSERIESBYTHAPATECHNICAL

Open user-model.js file and paste below code after declaration of mongoose variable:

const jwt = require("jsonwebtoken");

paste below code after userSchema variable:

userSchema.methods.generateToken = async function(){

    try{

        return jwt.sign({

            userId: this.\_id.toString(),

            email: this.email,

            isAdmin: this.isAdmin,

        },process.env.JWT\_SECRET\_KEY,{expiresIn:"30d",});

    }

    catch(error)

    {

        console.error(error);

    }

};

**PART – 13 – Login Route**

Open auth-router.js file and paste below code after: router.route('/register').post(authcontrollers.register);

router.route('/login').post(authcontrollers.login);

Open auth-controller.js file and paste below code after register() function:

const login = async (req,res) =>{

    try

    {

        const{email,password} = req.body;

        const userExist = await User.findOne({ email });

        console.log(userExist);

        if(!userExist)

        {

            res.status(400).json({ message: "Email does not exist!" });

return;

        }

        const chkpasswod = await bcrypt.compare(password,userExist.password);

        if(chkpasswod)

        {

            res.status(200).json({

                msg:"Login Successfull" ,

                token: await userExist.generateToken(),

                userId: userExist.\_id.toString()

            });

        }

        else

        {

            res.status(401).json({ message: "Password does not exist!" });

        }

    }

    catch(error)

    {

        res.status(400).json({msg:"internal server error"});

    }

}

And replace below code with exports variable;

module.exports = { home,register,login }

**PART – 14 – Password check function [ ignore ]**

Open auth-controller.js file and replace below code with instead of checkpassword variable, with in try block of login function.

const checkpassword = await userExist.comparePassword(password);

Open user-model.js file and paste below code after pre() function;

userSchema.methods.comparePassword = async function(password){

    return bcrypt.compare(password,this.password);

}

**PART – 15 – Login and Registration form validation (Zod validation)**

Run below command inside server folder:

**npm i zode OR** npm i zod-validation-error

create a folder **validators** inside server folder and create a file **auth-validator.js** inside validators folder.

Create a folder **middlewares** inside server folder and create a file **validate-middleware.js** file inside middlewares folder.

Open auth-validator.js file and paste below codes:

const{z} = require("zod");

const signupSchema = z

    .object({

     username: z.string({required\_error:"Name is required"})

               .trim()

               .min(3,{message:"Name must be atlist 3 characters"})

               .max(255,{ message:"Name must not be more than 255 characters"}),

    email: z.string({required\_error:"Email is required"})

            .trim().email({message:"Invalid email address"})

            .min(3,{message:"Email must be atlist 3 characters"})

            .max(255,{ message:"Email must not be more than 255 characters"}),

    phone: z.string({required\_error:"Phone is required"})

           .trim().min(10,{message:"Phone no. must be atlist 10 characters"})

           .max(20,{ message:"Phone no. must not be more than 20 characters"}),

    password: z.string({required\_error:"Password is required"})

              .trim().min(7,{message:"Password must be atlist 7 characters"})

              .max(1024,{ message:"Password must not be more than 1024 characters"})

    });

module.exports = signupSchema;

Note: username, email, phone and password are colum names of registration table.

Open validate-middleware.js file and paste below codes:

const validate = (schema) => async (req,res,next) =>{

    try

    {

        const parseBody = await schema.parseAsync(req.body);

        req.body = parseBody;

        next();

    }

    catch(err)

    {

        const message = err.errors[0].message;

        res.status(400).json({msg:message});

    }

}

module.exports = validate;

Open auth-router.js file and paste below code after authcontrollers variable:

const signupSchema = require("../validators/auth-validator.js");

const validate = require("../middlewares/validate-middleware.js");

and replace below code instead of router.route('/register').post(authcontrollers.register);

router.route('/register').post(validate(signupSchema),authcontrollers.register);

**PART – 16 – Errors Middleware (we can show all file errors in one file)**

**[ Currently not required when require we will use ]**

**[ Back end part finished ]**

Create a file **error-middleware.js** inside middlewares folder.

Open error-middleware.js file and paste below codes:

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

    const status = err.status || 500;

    const message = err.message || "BACKEND ERROR";

    const extraDetails = err.extraDetails || "Error from backend";

    return res.status(status).json({ message,extraDetails });

};

module.exports = errorMiddleware;

Open validate-middleware.js file and replace below code instead of catch block:

    catch(err)

    {

        const status = 422;

        const message = "Fill the input properly";

        const extraDetails = err.errors[0].message;

        const error = { status,message,extraDetails };

        //res.status(400).json({msg:message});

        next(error);

    }

Open server.js file and paste below code before PORT Declaration:

app.use(errorMiddleware);

Note: similaraly we can create for all other sections.

**PART – 17 – Contact Form**

Create a file contact-model.js file inside models folder. Open contact-model.js file and paste below codes:

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

const contactSchema = new Schema({

    username:{ type:String, required:true},

    email:{ type:String, required:true},

    message:{ type:String, required:true}

});

const Contact = new model("Contact",contactSchema);

module.exports = Contact;

Create a file contact-router.js file inside router folder. Open contact-router.js file and paste below code:

const express = require("express");

const router = express.Router();

const contactForm = require("../controllers/contact-controller");

router.route("/contact").post(contactForm);

module.exports = router;

Create a file contact-controller.js file inside controllers folder. Open contact-controller.js file and paste below code:

const Contact = require("../models/contact-model");

const contactForm = async (req,res) =>{

    try{

        const response = req.body;

        await Contact.create(response);

        return res.status(200).json({message:"message send successfully"});

    }

    catch(error)

    {

        return res.status(500).json({message:"message not delivered"});

    }

}

module.exports = contactForm;

Open server.js file and update below code instead of const router = require("./router/auth-router");

const authRoute = require("./router/auth-router");

again update below code instead of app.use("/api/auth",router);

app.use("/api/auth",authRoute);

add below code after authRoute variable:

const contactRoute = require("./router/contact-router");

again add below code after app.use("/api/auth",authRoute);

app.use("/api/form",contactRoute);

**PART – 18 – Overview about how to download source code**

**PART – 19 – Overview about how to create front-end part by react.js**

**PART – 20 – React Router DOM**

**[ Start Front end Part ]**

Run below command inside client folder :

npm i react-router-dom

create a new folder **pages** inside src folder than create Home.js , About.js , Contact.js , Login.js , Register.js and Service.js than

with default code.

Open App.js file and paste below code:

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

import Home from './pages/Home';

import About from './pages/About';

import Service from './pages/Service';

import Contact from './pages/Contact';

import Register from './pages/Register';

import Login from './pages/Login';

function App() {

  return (

   <>

      <BrowserRouter>

          <Routes>

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

              <Route path="/about" element={ <About /> }  />

              <Route path="/service" element={ <Service /> }  />

              <Route path="/contact" element={ <Contact /> }  />

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

              <Route path="/login" element={ <Login /> }  />

          </Routes>

      </BrowserRouter>

   </>

  );

}

export default App;

Note: (1) Agar ham **export default** karenge to ham direct import kar sakte hai by name, Agar ham sirf **default** likhenge to curly bracket ke andar import karna padega.

1. CTRL+SPACE for suggestion to import

**PART – 21 – Creating Navbar**

Create a new folder **components** inside src folder than create a file **Navbar.js** inside components folder.

Open Navbar.js file and paste below code:

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

const Navbar = ()=>{

    return(

        <>

            <nav>

                <ul>

                    <li><NavLink to="/home">Home</NavLink></li>

                    <li><NavLink to="/about">About</NavLink></li>

                    <li><NavLink to="/contact">Contact</NavLink></li>

                    <li><NavLink to="/register">Register</NavLink></li>

                    <li><NavLink to="/login">Login</NavLink></li>

                </ul>

            </nav>

        </>

    )

}

export default Navbar;

Open App.js file and import Navbar:

import Navbar from './components/Navbar';

than add below component with in <BrowserRouter> tag, before <Routes> tag

<Navbar />

Note: if we will store image folder inside public folder then we can direct access the images.

**PART – 22-Create registration page and store data in state**

Open register.js file and paste below code:

import React, { useState } from "react";

const Register=()=>{

    const[user,setUser] = useState({ //calling on pageload

        username:"",

        email:"",

        phone:"",

        password:""

    });

    const handleInput = (e)=>{ //calling on typing

        console.log(e);

        let name = e.target.name;

        let value = e.target.value;

        setUser({

            ...user,

            [name]:value

        })

    };

    const handleSubmit = (e) =>{ //calling on submit

        e.preventDefault();       //to stop page refresh

        console.log(user);

    };

    return(

        <>

          <form onSubmit={handleSubmit}>

             <h1>Register page</h1>

             <br/>

             <div>

                <label>Username</label>

                <input type="text" name="username" id="usernmae" value={user.username} onChange={handleInput} autoComplete="off"  />

             </div>

             <br/>

             <div>

                <label>Email</label>

                <input type="email" name="email" id="email" value={user.email} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Phone</label>

                <input type="number" name="phone" id="phone" value={user.phone} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Password</label>

                <input type="password" name="password" id="password" value={user.password} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <button type="submit">Register</button>

             </div>

          </form>

        </>

    )

}

export default Register;

**PART – 23- Create Login page and store data in state**

Open Login.js file and paste below code:

import React, { useState } from "react";

const Login=()=>{

    const[user,setUser] = useState({

        email:"",

        password:""

    });

    const handleInput = (e)=>{

        console.log(e);

        let name = e.target.name;

        let value = e.target.value;

        setUser({

            ...user,

            [name]:value

        })

    };

    const handleSubmit = (e) =>{

        e.preventDefault();       //to stop page refresh

        console.log(user);

    };

    return(

        <>

          <form onSubmit={handleSubmit}>

             <h1>Login page</h1>

             <br/>

             <div>

                <label>Email</label>

                <input type="email" name="email" id="email" value={user.email} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Password</label>

                <input type="password" name="password" id="password" value={user.password} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <button type="submit">Register</button>

             </div>

          </form>

        </>

    )

}

export default Login;

**PART – 24- Home page designing**

**PART – 25- Create Contact page and store data in state**

**PART – 26 – Creating 404 Error Page**

**PART – 27– Connect React with Node and Mongo DB**

already seen, check register.js , server.js, auth-controller.js, and other pages

register.js,

Open register.js file and import below :

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

and paste below code instead of handleSubmit function:

    const navigate = useNavigate();             //Added

    const handleSubmit = async (e) =>{

        e.preventDefault();       //to stop page refresh

        console.log(user);

        try{

            const response = await fetch('http://localhost:5000/api/auth/register',{

                method:"POST",

                headers:{

                    'Content-Type':'application/json'

                },

                body:JSON.stringify(user)

            });

const res\_data = await response.json();

console.log("response data",res\_data);

            if(response.ok)

            {

                setUser({

                    username:"",

                    email:"",

                    phone:"",

                    password:""

                });

                navigate("/login");

            }

            console.log(response);

        }

        catch(error)

        {

            console.log("Register:", error)

        }

    };

Than Run below command inside **server** (backend)folder:

**npm i cors** //used only for local server to handle CORS Error

Open server.js file and paste below code after express variable:

const cors = require("cors");

than Paste below code after **errorMiddleware** variable OR require("dotenv").config();

const corsOptions = {

                        origin:"http://localhost:3000",  //this is front end local url used only for local server

                        methods:"GET,POST,PUT,DELETE,PATCH,HEAD",

                        credentials: true

                    };

app.use(cors(corsOptions));

**PART – 28– Create Login Functionality**

Open auth-controller.js file and replace login function with below code:

 const login = async (req,res) =>{

    try

    {

        const{email,password} = req.body;

        const userExist = await User.findOne({ email });

        console.log("userExist",userExist);

        if(!userExist)

        {

            return res.status(400).json({ message: "Invalid Email!" });

        }

        const user = await userExist.comparePassword(password);

        //const user = await bcrypt.compare(password,userExist.password);

        console.log("password",password);

        console.log("password",userExist.password);

        //if(password === userExist.password)  // enable for using normal password

        if(user)

        {

            res.status(200).json({

                msg:"Login Successfull" ,

                token: await userExist.generateToken(),

                userId: userExist.\_id.toString()

            });

        }

        else

        {

            res.status(401).json({ message: "Invalid password" });

        }

    }

    catch(error)

    {

        res.status(400).json({msg:"internal server error"});

    }

}

Open user-model.js file and comment below code:

/\*userSchema.pre('save',async function(next){

    const user = this;

    if(!user.isModified('password'))

    {

        next();

    }

    try{

        const saltRound = await bcrypt.genSalt(10);

        const has\_password = await bcrypt.hash(user.password,saltRound);

        user.password = has\_password;

    }

    catch(error)

    {

        next(error);

    }

});  \*/

**PART – 29– Storing JWT Token data in Local Storage by Context api**

Open auth-controller.js file and replace below code instead of register function:

const register = async (req,res)=>{

    try

    {

        console.log("User details",req.body);

        const{username,email,phone,password} = req.body;

        const userExist = await User.findOne({email:email});

        if(userExist)

        {

            return res.status(400).json({msg:"Email already exist !!!"});

        }

        const saltRound = 10;

        const has\_password = await bcrypt.hash(password,saltRound);

        const data = { username:username,email:email,phone:phone,password:has\_password };

                                        /\*

                                        OR

        const data = { username:username,email:email,phone:phone,password:password };   //For Normal Password

                                        \*/

        /////////

       /\* console.log("uudata",data)

        const userCreated = await User.create(data)

        .then((resp)=>{

            console.log("Response",resp);

            return res.status(200).json({  msg:"registration sussess !!!" });

        }).catch((err)=>{console.log("error",err)}); \*/   // we can also use this code

        const userCreated = await User.create(data);

        if(userCreated)

        {

           return res.status(200).json({

                msg:"Registration Successfull" ,

                token: await userCreated.generateToken(),

                userId: userCreated.\_id.toString()

            });

        }

        else

        {

            res.status(401).json({ message: "Something went wrong" });

        }

    }

    catch(error)

    {

       console.log("Error",error)

       res.status(400).json({msg:"internal server error"});

    }

 }

Create a folder **store** inside src folder with in client(front-end) folder. Create a file **auth.js** inside store folder. Open auth.js file and paste below code:

import { createContext, useContext } from "react";

export const AuthContext = createContext();

export const AuthProvider = ({children}) =>{

    const storeTokenInLS = (serverToken)=>{

        return localStorage.setItem("token",serverToken);

    }

    return <AuthContext.Provider value={{storeTokenInLS}}>

        {children}

    </AuthContext.Provider>

};

export const useAuth=()=>{

    const authContextValue = useContext(AuthContext);

    if(!authContextValue)

    {

        throw new Error("useAuth used outside of the provider");

    }

    return authContextValue;

}

Open index.js file and replace below code instead of root.render():

root.render(

  <AuthProvider>

      <React.StrictMode>

          <App />

      </React.StrictMode>

  </AuthProvider>

);

And import below code in index.js file

import { AuthProvider } from './store/auth'; //it will automatically import

Open register.js file and add import below code on top :

import { useAuth } from "../store/auth";

than paste below code after handleInput() method:

const {storeTokenInLS} = useAuth();

than paste/add below code inside if() block of handleSubmit() function:

 storeTokenInLS(res\_data.token);

similaraly Open login.js file and add import below code on top : (if sign-in and sign-up both are different pages)

import { useAuth } from "../store/auth";

than paste below code after handleInput() method:

 const {storeTokenInLS} = useAuth();

than paste/add below code inside if() block of handleSubmit() function:

                storeTokenInLS(res\_data.token);

**To check**:

After registration or login >> inspect element >> application >> local storage >> we can see token and value.

**PART – 30 – Logout Functionality**

Open auth.js file add below code before storeTokenInLS function:

const[token,setToken] = useState(localStorage.getItem("token"));

let isLoggedIn = !! token;   //if value of token is exist then the value of isLoggedIn is true otherwise false

const LogoutUser = ()=>{

        setToken("");

        return localStorage.removeItem('token');

    }

And replace/update below code:

 return <AuthContext.Provider value={{ isLoggedIn, storeTokenInLS, LogoutUser}}>

        {children}

    </AuthContext.Provider>

Note: **useState** must be import.

Open **Navbar.js** file and below code must be import on top:

import { useAuth } from "../store/auth";

than replace Navbar() method with below code:

const Navbar = ()=>{

    const {isLoggedIn} = useAuth();

    return(

        <>

            <nav>

                <ul>

                    <li><NavLink to="/home">Home</NavLink></li>

                    <li><NavLink to="/about">About</NavLink></li>

                    <li><NavLink to="/contact">Contact</NavLink></li>

                    {

                        isLoggedIn ?  <li><NavLink to="/logout">Logout</NavLink></li>  :

                        <>

                            <li><NavLink to="/register">Register</NavLink></li>

                            <li><NavLink to="/login">Login</NavLink></li>

                        </>

                    }

                </ul>

            </nav>

        </>

    )

}

Create a file **Logout.js** inside pages folder, Open Logout.js file and paste below code:

import { useEffect } from "react";

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

import { useAuth } from "../store/auth";

const Logout=()=>{

    const { LogoutUser } = useAuth();

    useEffect(()=>{ LogoutUser(); },[]);

     /\*                 OR

     useEffect(()=>{ LogoutUser(); },[LogoutUser]);

     \*/

    return <Navigate to="/login" />

}

export default Logout;

Open App.js file and import below code:

import Logout from './pages/Logout';

than add below code with in <Routes> Tag:

<Route path="/logout" element={ <Logout /> }  />

Open Login.js file and import below code: [don’t require]

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

then add below code after handleInput() function: : [don’t require]

const navigate = useNavigate();

than add below code inside if block of handleSubmit function: : [don’t require]

navigate("/home");

**PART – 31-**

**JWT Token verification middleware + Creating routes to get loggedin user data from DataBase**

**(To check that user is logged in or NOT)**

**In server section:**

Create a new file **auth-middleware.js** inside middlewares folder. Open auth- middlewares.js file and paste below code:

const jwt = require("jsonwebtoken");

const User = require("../models/user-model");

const authMiddleware = async (req,res,next)=>{   //it iakes three parameter

    const token = req.header("Authorization");

    if(!token)

    {

        return res.status(401).json({message:"Unauthorized HTTP, Token not provided"});

    }

    const jwtToken = token.replace("Bearer","").trim();

    console.log("token from auth middleware",jwtToken);

    try{

        const isVerified = jwt.verify(jwtToken,process.env.JWT\_SECRET\_KEY);

        console.log(isVerified);

        const userData = await User.findOne({email:isVerified.email}).select({password:0});

        console.log(userData);

        req.user = userData;

        req.token = token;

        req.userID = userData.\_id;

        next();

    }

    catch(error)

    {

        return res.status(401).json({message:"Unauthorized , Invalid Token"});

    }

};

module.exports = authMiddleware;

Open auth-router.js file and paste below code after validate variable:

const authMiddleware = require("../middlewares/auth-middleware.js");

then add below code before module.exports=router;

router.route("/user").get(authMiddleware,authcontrollers.user);

Open auth-controller.js file and add below function:

const user=async (req,res)=>{

    try{

        const userData = req.user;

        console.log(userData);

        return res.status(200).json({userData});

    }

    catch(error)

    {

        console.log(`Error from the root: ${error}`);

    }

}

Then update below code:

module.exports = { home,register,login,user }

**PART – 32- Get loginned user data from database**

**Client side:**

Open auth.js file and add below code before storeTokenInLS() function:

const[user,setUser] = useState("");

Add below code after LogoutUser() function:

  const userAuthentication = async()=>{

        try

        {

            const response = await fetch("http://localhost:5000/api/auth/user",{

                method:"GET",

                headers:{Authorization:`Bearer ${token}`}

            });

            if(response.ok)

            {

                const data = await response.json();

                console.log("user data",data.userData);

                setUser(data.userData);

            }

        }

        catch(error)

        {

        }

    };

    useEffect(()=>{

        userAuthentication();

    },[]);

Then update below code :

 return <AuthContext.Provider value={{ isLoggedIn, storeTokenInLS, LogoutUser, user}}>

        {children}

    </AuthContext.Provider>

Open Contact.js file and entire page code replace with below code:

import React, { useState } from "react";

import { useAuth } from "../store/auth";

const Contact=()=>{

    const[contact,setContact] = useState({

        username:"",

        email:"",

        message:""

    });

    const[userData,setUserData]=useState(true);

    const {user} = useAuth();

    if(userData && user)

    {

        setContact({

            username:user.username,

            email:user.email,

            message:""

        });

        setUserData(false);

    }

    const handleInput = (e)=>{

        console.log(e);

        let name = e.target.name;

        let value = e.target.value;

        setContact({

            ...contact,

            [name]:value

        });

                //    OR

        // setUser((prev) => ({

        //     ...prev,

        //     [name]:value

        // }));

    };

    const handleSubmit = (e) =>{

        e.preventDefault();       //to stop page refresh

        console.log(user);

    };

    return(

        <>

       <form onSubmit={handleSubmit}>

             <h1>Contact page</h1>

             <br/>

             <div>

                <label>Username</label>

                <input type="text" name="username" id="username" value={user.username} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Email</label>

                <input type="email" name="email" id="email" value={user.email} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Message</label>

                <textarea name="message" id="message" value={user.message} onChange={handleInput} autoComplete="off" cols="50" rows={5}></textarea>

             </div>

             <br/>

             <div>

                <button type="submit">Submit</button>

             </div>

          </form>

        </>

    )

}

export default Contact;

then check contact us page after login, data will auto fill.

**PART – 33-Display Logged in user name in about us page**

Open about.js file and import below code:

import { useAuth } from "../store/auth";

then paste below code before return function:

const {user} = useAuth();

then paste below code inside return function:

<h2>Welcome {user ? `${user.username} to our site` : `to our site`}</h2>

Then check about us page after login and before login.

**PART – 34 – Store contact form data in database**

Open Contact.js file and replace entire page code with below code(we have chaged only in functionality not in html code):

import React, { useState } from "react";

import { useAuth } from "../store/auth";

const Contact=()=>{

    const defaultContactFormData = {

        username:"",

        email:"",

        message:""

    };

    const[contact,setContact] = useState(defaultContactFormData);

    const[userData,setUserData]=useState(true);

    const {user} = useAuth();

    if(userData && user)

    {

        setContact({

            username:user.username,

            email:user.email,

            message:""

        });

        setUserData(false);

    }

    const handleInput = (e)=>{

        console.log(e);

        let name = e.target.name;

        let value = e.target.value;

        setContact({

            ...contact,

            [name]:value

        });

                //    OR

        // setUser((prev) => ({

        //     ...prev,

        //     [name]:value

        // }));

    };

    const handleSubmit = async (e) =>{

        e.preventDefault();       //to stop page refresh

        console.log(user);

        try{

            const response = await fetch("http://localhost:5000/api/form/contact",{

                method:"POST",

                headers:{'Content-Type':'application/json'},

                body:JSON.stringify(contact)

            });

            if(response.ok)

            {

                setContact(defaultContactFormData);

                const data = await response.json();

                console.log(data);

                alert("Message send successfully!!");

            }

        }

        catch(error)

        {

            alert("Messenge not send",error)

        }

    };

    return(

        <>

       <form onSubmit={handleSubmit}>

             <h1>Contact page</h1>

             <br/>

             <div>

                <label>Username</label>

                <input type="text" name="username" id="username" value={user.username} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Email</label>

                <input type="email" name="email" id="email" value={user.email} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Message</label>

                <textarea name="message" id="message" value={user.message} onChange={handleInput} autoComplete="off" cols="50" rows={5}></textarea>

             </div>

             <br/>

             <div>

                <button type="submit">Submit</button>

             </div>

          </form>

        </>

    )

}

export default Contact;

then fill contact form and submit then check .

**PART – 35- Create new collection, store records manually and create all functionality in backend to show data in front-end**

**Creating new collection:**

Login mongodb atlas >> database >>browse collections >> hover on database name and click on + icon >> Put collection name in plural(suppose **services**) and create >> go to insert document >> select {} icon >> we can write manually and insert like:

[

{ “service”:”this is service1”, “price”:”this is price1” },

{ “service”:”this is service2”, “price”:”this is price2” },

{ “service”:”this is service3”, “price”:”this is price3” }

]

**Sersver side:**

Create a new file **service-model.js** inside models folder. Open service-model.js file and paste below code:

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

const serviceSchema = new Schema({

    service:{ type:String, required:true},

    description:{ type:String, required:true},

    price:{ type:String, required:true},

    provider:{ type:String, required:true}

});

const Service = new model("Service",serviceSchema);

//where Service is collection name singular form first letter capital

module.exports = Service;

Create a new file **service-router.js** inside router folder. Open service-router.js file and paste below code:

const express = require("express");

const services = require("../controllers/service-controller");

const router = express.Router();

router.route("/service").get(services);

module.exports = router;

create **service-controller.js** file inside controllers folder. Open service-controller.js file and paste below code:

const Service = require("../models/service-model");

const services = async (req,res)=>{

    try

    {

        const response = await Service.find();

        if(!response)

        {

            res.status(404).json({ msg:"No service found" });

            return;

        }

        res.status(200).json({ msg:response });

    }

    catch(error)

    {

        console.log(`service err: ${error}`)

    }

}

module.exports = services;

then open server.js file and paste below code after contactRoute variable:

const serviceRoute = require("./router/service-router");

then add below code before app.use(errorMiddleware);

app.use("/api/data",serviceRoute);

**PART – 36 – Fetch and show data from database in front-end**

**Client side:**

Open auth.js file and add below code after const[user,setUser] = useState("");

const[services,setServices] = useState("");

add below code inside useEffect() method:

getServices();

then add below code before useEffect() method:

    const getServices = async ()=>{

        try

        {

            const response = await fetch("http://localhost:5000/api/data/service",{

                method:"GET"

            });

            if(response.ok)

            {

                const data = await response.json();

                console.log(data.msg);

                setServices(data.msg);

            }

        }

        catch(error)

        {

            console.log(`Services error: ${error}`);

        }

    }

Then update below code (added services):

return <AuthContext.Provider value={{ isLoggedIn, storeTokenInLS, LogoutUser, user, services}}>

        {children}

    </AuthContext.Provider>

Open home.js file and import below code:

import { useAuth } from "../store/auth";

add below code before return function;

const {services} = useAuth();

then use below code inside return function;

           <div className="parent">

             {

                services ?

                services.map((curElem,index)=>{

                    const { price,description,provider,service }=curElem;

                    return (<div className="child" key={index}>

                        <img src="/logo192.png" alt="image" width={100} />

                        <div className="card">

                            <h5>{provider}</h5>

                            <h5>{price}</h5>

                            <h5>{service}</h5>

                            <h5>{description}</h5>

                        </div>

                    </div>);

             })

             : `Record not found`

             }

            </div>

**PART – 37 – Form Validation**

**Server side:**

Open auth-controller.js file and replace below code inside register function:

if(userExist)

        {

            return res.status(400).json({message :"Email already exist !!!"});   //updated

        }

Open auth-validator.js file and replace entire code with below code:

const{z} = require("zod");

const loginSchema = z.object({

    email: z.string({required\_error:"Email is required"})

    .trim().email({message:"Invalid email address"})

    .min(3,{message:"Email must be atlist 3 characters"})

    .max(255,"Email must not be more than 255 characters"),

    password: z.string({required\_error:"Password is required"})

              .trim().min(7,{message:"Password must be atlist 7 characters"})

              .max(1024,"Password must not be more than 1024 characters")

});

/\*

 in signupSchema we can use

 const signupSchema = loginSchema.extend({

    username: z.string({required\_error:"Name is required"})

               .trim()

               .min(3,{message:"Name must be atlist 3 characters"})

               .max(255,"Name must not be more than 255 characters"),

    phone: z.string({required\_error:"Phone is required"})

           .trim().min(10,{message:"Phone no. must be atlist 10 characters"})

           .max(20,"Phone no. must not be more than 20 characters")

 })

//that means loginSchema ka data + username and phone i.e. 4 validation add ho jayega

\*/

const signupSchema = z.object({

     username: z.string({required\_error:"Name is required"})

               .trim()

               .min(3,{message:"Name must be atlist 3 characters"})

               .max(255,"Name must not be more than 255 characters"),

    email: z.string({required\_error:"Email is required"})

            .trim().email({message:"Invalid email address"})

            .min(3,{message:"Email must be atlist 3 characters"})

            .max(255,"Email must not be more than 255 characters"),

    phone: z.string({required\_error:"Phone is required"})

           .trim().min(10,{message:"Phone no. must be atlist 10 characters"})

           .max(20,"Phone no. must not be more than 20 characters"),

    password: z.string({required\_error:"Password is required"})

              .trim().min(7,{message:"Password must be atlist 7 characters"})

              .max(1024,"Password must not be more than 1024 characters")

    });

module.exports = { signupSchema , loginSchema };

Open auth-router.js file update below code after authcontrollers variable:

const { signupSchema, loginSchema } = require("../validators/auth-validator.js");

then replace below code (added validation(loginSchema))

router.route('/login').post(validate(loginSchema),authcontrollers.login);

**Client side:**

Open Register.js file and replace below code with handleSubmit function:

    const handleSubmit = async (e) =>{

        e.preventDefault();       //to stop page refresh

        console.log(user);

        try{

            const response = await fetch('http://localhost:5000/api/auth/register',{

                method:"POST",

                headers:{

                    'Content-Type':'application/json'

                },

                body:JSON.stringify(user)

            });

            const res\_data = await response.json();

            console.log("response data",res\_data);

            if(response.ok)

            {

                storeTokenInLS(res\_data.token);

                setUser({

                    username:"",

                    email:"",

                    phone:"",

                    password:""

                });

                navigate("/login");

            }

            else{

                alert(res\_data.extraDetails ? res\_data.extraDetails : res\_data.message);

            }

            //console.log(response);

        }

        catch(error)

        {

            console.log("Register:", error)

        }

    //    await axios.post('localhost:5000/api/auth/register',user).then((res)=>{

    //         console.log('res',res?.data)

    //     }).catch((err)=>{

    //         console.log('err',err)

    //     });

    };

Open Login.js file and replace below code with handleSubmit function:

    const handleSubmit = async (e) =>{

        e.preventDefault();       //to stop page refresh

        console.log(user);

        try{

            const response = await fetch('http://localhost:5000/api/auth/login',{

                method : "POST",

                headers : {

                    'Content-Type':'application/json'

                },

                body : JSON.stringify(user)

            });

            console.log("login form",response);

            const res\_data = await response.json();

            if(response.ok)

            {

                alert("Login Success!!");

                storeTokenInLS(res\_data.token);

                setUser({ email:"",password:"" });

                navigate("/home");

            }

            else{

                alert(res\_data.extraDetails ? res\_data.extraDetails : res\_data.message);

                console.log("invalid credential");

            }

        }

        catch(error)

        {

            console.log(error);

        }

    };

**PART – 38 – implementing / integrating , react toastify dynamic alerts for validation**

Inside client folder run below command:

**npm i react-toastify**

Note: search “npm react toastify” and follow instruction

**Client side:**

Open index.js file and import below codes:

import { ToastContainer } from "react-toastify";

import 'react-toastify/dist/ReactToastify.css';

then update below code:

root.render(

  <AuthProvider>

      <React.StrictMode>

          <App />

      </React.StrictMode>

      <ToastContainer />

  </AuthProvider>

);

Open Login.js file and import below code:

import { toast } from "react-toastify";

then use **toast.success** instead of alert. // for success message

and use **toast.error** instead of alert. // for error case

Note: similaraly we can use in Register.js file

**PART – 39 – Show Logout after login**

**Client side:**

Open auth.js file and add below code inside storeTokenInLS() , before return

setToken(serverToken);

**PART – 40- API creation for getting users records for admin panel**

**Server side:**

Create a new file **admin-router.js** inside router folder. Open admin-router.js file and paste below code:

const express = require('express');

const getAllusers = require('../controllers/admin-controller');

const router = express.Router();

router.route('/users').get(getAllusers);    //press ctrl + space for auto suggestion

module.exports = router;

Create a new file **admin-controller.js** inside controllers folder. Open admin-controller.js file and paste below code:

const User = require("../models/user-model");

const getAllusers = async (req,res)=>{

    try{

        const users = await User.find({},{password:0});  //it will not fetch password colom

        console.log(users);

        if(!users || users.length===0)

        {

            res.status(404).json({message:"No record found!"});

            return;

        }

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

    }

    catch(error)

    {

        next(error);

    }

};

module.exports = getAllusers;

Open server.js file and paste below code after serviceRoute variable:

const adminRoute = require("./router/admin-router");

then paste below code before app.use(errorMiddleware);

app.use("/api/admin",adminRoute);

then we can check below api on postman using GET method:

[**http://localhost:5000/api/admin/users**](http://localhost:5000/api/admin/users)

**PART – 41 - API creation for getting contacts records for admin panel**

**Server side:**

Open admin-router.js file and replace all code with below code:

const express = require('express');

const adminController = require('../controllers/admin-controller');   //updated

const router = express.Router();

router.route('/users').get(adminController.getAllusers);    //press ctrl + space for auto suggestion     //updated

router.route('/contacts').get(adminController.getAllContacts);    //added

module.exports = router;

Open admin-controller.js file and paste below code after User variable:

const Contact = require("../models/contact-model");

then paste below code after getAllusers() function:

const getAllContacts = async (req,res)=>{

    try

    {

        const contacts = await Contact.find(); //it will fetch all records of contacts collection

        console.log(contacts);

        if(!contacts || contacts.length===0)

        {

            res.status(404).json({message:"No record found!"});

            return;

        }

        return res.status(200).json(contacts);

    }

    catch(error)

    {

        next(error);

    }

};

Then update below code:

module.exports = { getAllusers, getAllContacts} ;

then we can check below api on postman using GET method:

[**http://localhost:5000/api/admin/contacts**](http://localhost:5000/api/admin/contacts)

**PART – 42 – Adding JWT Verification in admin panel for security(to check user is logged in or not)**

**Server side:**

Open admin-router.js file and add below code after adminController variable:

const authMiddleware = require("../middlewares/auth-middleware");

then update below code:

router.route('/users').get(authMiddleware,adminController.getAllusers);

router.route('/contacts').get(authMiddleware,adminController.getAllContacts);

then we can check below api will not work without token(i.e. login) by postman:

[**http://localhost:5000/api/admin/users**](http://localhost:5000/api/admin/users)

[**http://localhost:5000/api/admin/contacts**](http://localhost:5000/api/admin/contacts)

**PART – 43 – Admin Routes and Navigation**

**Client side:**

Create a new folder **layouts** inside components folder , then create a new file **AdminLayout.js** inside layouts folder. Open AdminLayout.js file and paste below code:

import React from 'react'

import { NavLink, Outlet } from 'react-router-dom'

const AdminLayout = () => {

  return (

    <>

      <div>

          <ui>

              <li><NavLink to="/admin/users">users</NavLink></li>

              <li><NavLink to="/admin/contacts">contacts</NavLink></li>

              <li><NavLink to="/admin/services">services</NavLink></li>

              <li><NavLink to="/">home</NavLink></li>

          </ui>

      </div>

      <Outlet />

    </>

  )

}

export default AdminLayout

Create 2 new files **AdminUsers.js** and **AdminContacts.js** . Open AdminUsers.js file and paste below code:

import React from 'react'

const AdminUsers = () => {

  return (

    <div>AdminUsers</div>

  )

}

export default AdminUsers

then open AdminContacts.js file and paste below code:

import React from 'react'

const AdminContacts = () => {

  return (

    <div>AdminContacts</div>

  )

}

export default AdminContacts

Open App.js file and import below code:

import AdminLayout from './components/layouts/AdminLayout';

import AdminUsers from './pages/AdminUsers';

import AdminContacts from './pages/AdminContacts';

then paste below code inside <Routes> tag (this is nested Route):

<Route path="/admin" element={ <AdminLayout /> }>

                  <Route path="users" element={ <AdminUsers /> } />

                  <Route path="contacts" element={ <AdminContacts /> } />

              </Route>

Then we can check nested routes by using below url:

<http://localhost:3000/admin/contacts>

<http://localhost:3000/admin/users>

**PART – 44 – Fetch all data in Admin Panel**

Open auth.js file and paste below code before storeTokenInLS() function.

const authorizationToken = `Bearer ${token}`;

then below code:

 return <AuthContext.Provider value={{ isLoggedIn, storeTokenInLS, LogoutUser, user, services, authorizationToken}}>

        {children}

    </AuthContext.Provider>

Open AdminUsers.js file and replace entire code with below code:

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

import { useAuth } from '../store/auth';

const AdminUsers = () => {

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

  const { authorizationToken } = useAuth();

  const getAllUsersData = async()=>{

    try{

      const response = await fetch("http://localhost:5000/api/admin/users",{

        method:"GET",

        headers:{ Authorization : authorizationToken }

      });

      const data = await response.json();

      setUsers(data);

      console.log(`users data: ${data}`);

    }

    catch(error)

    {

      console.log(error);

    }

  };

  useEffect(()=>{

    getAllUsersData();

  },[]);

  return (

    <>

                <table cellPadding={5} cellSpacing={5} border={1} style={{borderCollapse:'collapse'}}>

                      <thead>

                          <tr>

                              <th>Name</th>

                              <th>Email</th>

                              <th>Phone</th>

                              <th>Action</th>

                          </tr>

                      </thead>

                      <tbody>

                  {

                      users ?

                      users.map((curUser,index)=>{

                      return (

                      <tr key={index}>

                          <td>{curUser.username}</td>

                          <td>{curUser.email}</td>

                          <td>{curUser.phone}</td>

                          <td>Edit , Delete</td>

                      </tr>

                      );

                    })

                    : `No record found`

                  }

             </tbody>

                  </table>

    </>

  )

}

export default AdminUsers

then login and check users page of admin panel:

[**http://localhost:3000/admin/users**](http://localhost:3000/admin/users)

**PART – 45 – Delete Records from database for admin panel**

**Server side:**

Open admin-router.js file and paste below code after adminController variable:

const authMiddleware = require("../middlewares/auth-middleware");

then add below code before module.exports = router;

router.route("/users/delete/:id").delete(authMiddleware,adminController.deleteUserById);

Open admin-controller.js file and add below code after getAllContacts() function

const deleteUserById = async(req,res)=>{

    try{

        const id = req.params.id;

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

        return res.status(200).json({ message:"Record deleted successfully!" });

    }

    catch(error)

    {

        next(error);

    }

};

Then update below code:

module.exports = { getAllusers, getAllContacts, deleteUserById};

**Client-side:**

Open AdminUsers.js file and replace below code instead of delete text:

<button onClick={()=>{ deleteUser(curUser.\_id) }}>Delete</button>

Then add below code after getAllUsersData() function:

  const deleteUser = async (id)=>{

    try{

    const response = await fetch(`http://localhost:5000/api/admin/users/delete/${id}`,{

      method:"DELETE",

      headers:{ Authorization : authorizationToken }

    });

    const data = await response.json();

    alert('record deleted');

    if(response.ok)

    {

      getAllUsersData();

    }

  }

  catch(error)

  {

    console.log(error);

  }

  };

Relogin and delete user to check.

**PART – 46 – get current user data**

**Client side:**

Open AdminUsers.js file and replace below code instead of edit text:

<Link to={`/admin/users/${curUser.\_id}/edit`}>Edit</Link>

Then import below code:

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

**Server side:**

Open admin-router.js file and paste below code before module.exports = router;

router.route("/users/:id").get(authMiddleware,adminController.getUserById);

Open admin-controller.js file and paste below code after deleteUserById() function:

const getUserById = async(req,res)=>{

    try{

        const id = req.params.id;

        const data = await User.findOne({ \_id:id },{ password:0 });

        return res.status(200).json(data);

    }

    catch(error)

    {

        next(error);

    }

};

Then update below code:

module.exports = { getAllusers, getAllContacts, deleteUserById,getUserById};

**PART – 47 – Prepare back end code to edit record**

Open admin-controller.js file and paste below code after getUserById() function:

const updateUserById = async(req,res)=>{

    try{

        const id = req.params.id;

        const updatedUserData = req.body;

        const updatedData = await User.updateOne(

            {\_id:id},

            {

                $set:updatedUserData

            });

        return res.status(200).json(updatedData);

    }

    catch(error)

    {

        next(error);

    }

};

Then update below code:

module.exports = { getAllusers, getAllContacts, deleteUserById,getUserById,updateUserById};

Open admin-router.js file and paste below code before module.exports = router;

router.route("/users/update/:id").patch(authMiddleware,adminController.updateUserById);

**PART – 48 – Edit Records**

**Client side:**

Open AdminUsers.js and replace below code instead of Edit text:

<Link to={`/admin/users/${curUser.\_id}/edit`}>Edit</Link>

Then import below code;

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

Create a new file **AdminUpdate.js** inside pages folder. Open AdminUpdate.js file and paste below code:

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

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

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

import { useAuth } from '../store/auth';

const AdminUpdate = () => {

  const navigate = useNavigate();

  const [data,setData] = useState({

    username:"",email:"",phone:""

  });

  const params = useParams();

  console.log("single params",params);

  const { authorizationToken } = useAuth();

  const getSingleUsersData = async()=>{

    try{

        const response = await fetch(`http://localhost:5000/api/admin/users/${params.id}`,{

        method:"GET",

        headers:{ Authorization : authorizationToken }

      });

      const data = await response.json();

      setData(data);

      console.log(`users single data: ${data}`);

    }

    catch(error)

    {

      console.log(error);

    }

  };

  const handleInput = (e)=>{

    let name = e.target.name;

    let value = e.target.value;

    setData({

      ...data,

      [name]:value

    });

  };

  const handleSubmit = async (e)=>{

    e.preventDefault();

    try{

      const response = await fetch(`http://localhost:5000/api/admin/users/update/${params.id}`,{

      method:"PATCH",

      headers:{

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

                  Authorization : authorizationToken

              },

      body:JSON.stringify(data)

    });

    if(response.ok)

    {

      alert("Updated successfully!");

      navigate('/admin/users');

    }

    else

    {

      alert("Not updated");

    }

  }

    catch(error)

    {

      console.log(error);

    }

  };

  useEffect(()=>{

    getSingleUsersData();

  },[]);

  return (

    <>

             <form onSubmit={handleSubmit}>

             <h1>Update Record</h1>

             <br/>

             <div>

                <label>Username</label>

                <input type="text" name="username" id="usernmae" value={data.username} onChange={handleInput} autoComplete="off"  />

             </div>

             <br/>

             <div>

                <label>Email</label>

                <input type="email" name="email" id="email" value={data.email} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <label>Phone</label>

                <input type="number" name="phone" id="phone" value={data.phone} onChange={handleInput} autoComplete="off" />

             </div>

             <br/>

             <div>

                <button type="submit">Update</button>

                <span>phone no. should be 10 digit and password length should be 7 characters</span>

             </div>

          </form>

    </>

  )

}

export default AdminUpdate

Open App.js file and import below code:

import AdminUpdate from './pages/AdminUpdate';

then update below code:

<Route path="/admin" element={ <AdminLayout /> }>

                  <Route path="users" element={ <AdminUsers /> } />

                  <Route path="users/:id/edit" element={ <AdminUpdate /> } />

                  <Route path="contacts" element={ <AdminContacts /> } />

              </Route>

Then update any record and check.

**PART – 49 – Fetching and showing contact records**

**Client side:**

Open AdminContacts.js file and replace all code with below code:

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

import {useAuth} from "../store/auth";

const AdminContacts = () => {

  const [ contactData,setContactData ] = useState([]);

  const { authorizationToken } = useAuth();

  const getContactsData = async ()=>{

    try{

      const response = await fetch("http://localhost:5000/api/admin/contacts",{

        method : "GET",

        headers : { Authorization : authorizationToken }

      });

      const data = await response.json();

      if(response.ok)

      {

        setContactData(data);

      }

    }

    catch(error)

    {

      console.log(error);

    }

  };

  useEffect(()=>{ getContactsData(); },[]);

  return (

    <>

      <h1>Contact List</h1>

      {

        contactData ?

        contactData.map((curContactData,index)=>{

                    const { username,email,message }=curContactData;

                    return (<div className="child" key={index} style={{float:'left',border:'1px solid'}}>

                        <div className="card">

                            <h5>{username}</h5>

                            <h5>{email}</h5>

                            <h5>{message}</h5>

                           <button>Delete</button>

                        </div>

                    </div>);

             })

             : `Record not found`

             }

    </>

  )

}

export default AdminContacts

then check ..

**PART – 50 – Delete contact records**

**Server side:**

Open admin-router.js file and add below code before module.exports = router;

router.route("/contacts/delete/:id").delete(authMiddleware,adminController.deleteContactById);

Open admin-controller.js file and add below code after updateUserById() function:

const deleteContactById = async(req,res)=>{

    try{

        const id = req.params.id;

        await Contact.deleteOne({ \_id:id });

        return res.status(200).json({ message:"Contact deleted successfully!" });

    }

    catch(error)

    {

        next(error);

    }

};

Then update below code:

module.exports = { getAllusers, getAllContacts, deleteUserById,getUserById,updateUserById,deleteContactById };

**Client side:**

Open AdminContacts.js file and replace delete button with below code:

<button onClick={()=>{ deleteContactById(\_id) }}>Delete</button>

Then update below code:

const { \_id,username,email,message }=curContactData;

than add below code before useEffect() function:

  const deleteContactById = async (id)=>{

    try{

      const response = await fetch(`http://localhost:5000/api/admin/contacts/delete/${id}`,{

          method : 'DELETE',

          headers : { Authorization : authorizationToken }

      });

      if(response.ok)

      {

        alert("Contact deleted successfully");

        getContactsData();

      }

    }

    catch(error)

    {

      console.log(error);

    }

  };

Then check on browser by deleting any contact record.

**PART – 51 – How to secure admin route on front – end**

(Getting Error)

**PART – 52 – General instruction about hosting / deployment**

**PART – 53 – Deployment**

**PART – 54 – To solve hosting issue**

Open auth.js file and replace below code instead of const[services,setServices] = useState([]); inside AuthProvider() function:

const[services,setServices] = useState([]);

**Note:**

Currenty we use [**http://localhost:5000/**](http://localhost:5000/)this path for backend in login,register,….and so many.

It will get issue on the time of deployment. To solve this issue we need to create a variable in .env file and need to use this variable in every page where we have use this path.

After that we need to change one place and it will reflect every place.