await and async are keywords used in asynchronous programming to handle asynchronous operations.

async Function:

The async keyword is used to define a function as asynchronous. When a function is

marked as async, it means that it can contain await expressions inside it.

await Expression

The await keyword is used inside an async function to wait for the completion of a promise or a coroutine. It allows the program to pause and wait for the asynchronous operation to complete before proceeding to the next line of code.

In the context of a response, if you are dealing with asynchronous operations, usingasync functions and await expressions can make your code more readable and maintainable. It helps in handling operations that might takesome time to complete, such as fetching data from an API, reading from a file, or making network requests, without blockingthe execution of the entire program. This is particularly useful in scenarios where waiting for a response or data retrieval could cause significant delays if done synchronously.

Package.json will store all the dependencies of all the package used in form of meta data

For creating: npm init -y

**BACKEND-START**

npm i express

install nodemon to run app.js

simple -> node app.js (no autorunning)

nodemon -> **nodemon app.js** ( auto-run, for running )

video 4

Middleware can be used to implement authentication and authorization checks before allowing access to certain routes or resources. For example, you might have middleware that verifies a user's token before granting access to protected routes, like trying to ascess aboutMe of a user without login it redirects user and ask to signIn first before able to see aboutMe

cmd -> mkdir server

Move app.js, node modules, package.json, package-lock.json to server

Middleware stage ( app.js )----------------------------------------------------------

const express = require('express');

const app = express();

*// Middleware*

const middleware =(req,res, next) => {

    console.log('Hello middleware, checking verification');

    next(); *//it allows to move in next phase after verfying*

}

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

    res.send('heyyyyyyyyyx');

})

app.get('/about',middleware, (req,res) => {

    console.log('about verified by middleware');

    res.send('about you................');

})

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

    res.send('contact');

})

*//console.log('beeeeeeeeeees');*

app.listen(3000, () => {

    console.log('server is running in port 3000 ');

} )

Video 5

Connect backend to MongoDB database

Visit mogodb web

Create new project

Create new cluster

Create database -> mernstack

collection name->user

App.js

const mongoose = require('mongoose');

const express = require('express');

const app = express();

const DB = 'mongodb+srv://akashghosh256:akash1234@cluster0.yztdwsf.mongodb.net/?retryWrites=true&w=majority';

mongoose.connect(DB).then(() =>{

    console.log('connection successful');

}).catch((err) => console.log('no connection'));

Video 6 – secure code

Install server> npm i dotenv

The use of **config.env** likely refers to a configuration file or module where environmental variables are stored for your Node.js application. Environmental variables are often used to store configuration settings that can vary between development, testing, and production environments without modifying the source code.

(If system doesn’t finds the config.env then it will create a new one)

Server-cmd -> type null > config.env

Move the password and db link in config,env and set ascess path for it

dotenv.config({path: './config.env'});

now move the database part to new folder

server>create folder db>create file conn.js

video 7

server>create folder models>create file userSchema.js

It will store format of how data is stored in MongoDB

video 8

server > create router folder > create auth.js

**React have its own router in frontend**

**exprees is router for backend**

video 8

Try to GET data using postman from express.js

Watch video how create your own restful api Thapa technical

Video 9

Just learnt how to add data to mongodb, file used auth.js

const express = require('express');

const router = express.Router();

*//connecting to db for registering users and also checking its a new user*

require('../db/conn');

const User = require("../models/userSchema");

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

    res.send('hello from express router auth.js');

})

*// To get data from frontend always do same*

router.post('/register', (req,res) =>{

const { name, email, phone, work, password, cpassword} = req.body;

if( !name|| !email|| !phone|| !work|| !password|| !cpassword){

    return res.status(422).json({error:" fill all the fields"});

}

*// verifying email is new for new registration, responds in true or false*

User.findOne({email:email}).then((userExist) =>{

    if(userExist){

        return res.status(422).json({error:" Email already exist"});

    }

*// else not used*

    const user = new User({name, email, phone, work, password, cpassword});

    user.save().then(() => {

         res.status(201).json({message:"user registerd succesfully"});

     }).catch((err) => res.status(500).json({error:"failed to register because of database error"})); *// if any error happens during registering catch function will take care*

})

*// console.log(req.body);*

*// console.log(email);*

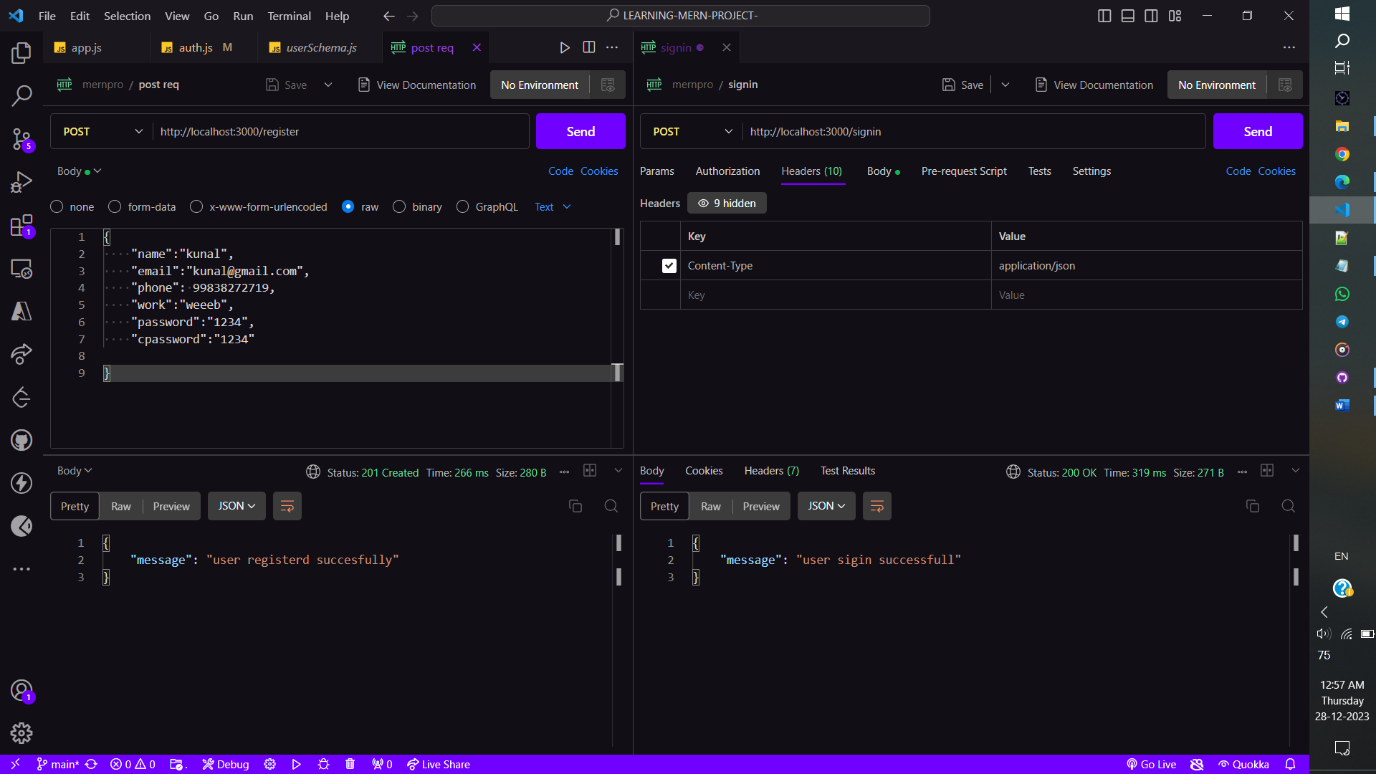
*// res.json({message:req.body});*

*//res.send("mera hnnn");*

});

module.exports = router;

video 12

create login/signin route for backend and test with postman

video 13

securing password ( Auth.js and userSchema.js )

cmd-> npm i bcryptjs

[bcrypt - npm (npmjs.com)](https://www.npmjs.com/package/bcrypt)

Guys for hashing use this.password = bcrypt.hashSync(this.password, 12); this.cpassword = bcrypt.hashSync(this.cpassword, 12);

Video 14 checking hash password and trying signin

if(userLogin){

const isMatch = await bcrypt.compare(password, userLogin.password);

if(!isMatch){

*// invalid password*

    res.status(400).json({error:"Invalid Credentials"});

}

else{

*// All good*

    res.status(400).json({error:"user signin successfull"});

}}

else{

*//invalid email*

    res.status(400).json({error:"Invalid Credentials"});

}

console.log(userLogin);

if(!userLogin){

res.status(400).json({ error: "user error"});

}

else{

    res.json({ message: "user sigin successfull"});

}

}

catch(err){

console.log(err);

        }

});

Video 15

[Using cookies to store JWT for authentication and authorization in a MERN stack app | by Alfurquan Zahedi | Medium](https://medium.com/@zahedialfurquan20/using-cookies-to-store-jwt-for-authentication-and-authorization-in-a-mern-stack-app-a58d7a5d6b6e)

Here we will authenticate user is login before ascessing aboutMe page, we will generate a web token during login and this token will help us to give ascess to all the pages to the login user and verify.

1. Generate JWT Token and stored it in Database
2. How to stored the token in cookies
3. Get Token from cookie and verify the user

Server-cdm -> npm i jsonwebtoken

If Return promises use **await**

Error 1

User

**You**

Error [ERR\_HTTP\_HEADERS\_SENT]: Cannot set headers after they are sent to the client at new NodeError (node:internal/errors:406:5) at ServerResponse.setHeader (node:\_http\_outgoing:652:11) at ServerResponse.header (D:\GitHub Desktop\LEARNING-MERN-PROJECT-\server\node\_modules\express\lib\response.js:794:10) at ServerResponse.send

ChatGPT

**ChatGPT**

The "Cannot set headers after they are sent to the client" error in Node.js typically occurs when you try to send a response to the client after you've already sent a response. This usually happens when you attempt to send multiple responses for a single HTTP request. In Express, the response is sent to the client with methods like **res.send()**, **res.json()**, or **res.end()**.

Solved by sending only one response for a particular case

Error 2

Bro, shouldn't we just send the generated token inside the headers using res.headers instead of saving it every time in db. I saw that you used to concat, which means the token is going to be saved every time a user logins. That would use the entire database if we have a large user base. Do you have a solution for this?

I solved it

Video 16

res.cookie("mernjwttoken", token,{

    expires: new Date(Date.now() + 300000), *//5 min*

    httpOnly:true

});

Saving token in cookies, token is getting saved in server side you can check in **postman cookie section**

Video 17

**FRONTEND START CMD**

Now create react app cmd + **npx create-react-app folder\_name (here client)**

**cd client**

**npm start /yarn start**

**video 18**

client cdm-> npm i [bootstrap@5.0.1](mailto:bootstrap@5.0.1)

or

add bootstrap in client>public>index.html

this index.html page renders out page in frontend

client>src>create folder components

components>create files Navbar.js About.js Home.js Contact.js

React file shortcut for dummy code -> rafce + tab

Video 19

To load pages without refresh

Client + cdm -> npm i react-router-dom

Navbar.js

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

this will load pages without refreshing

and replace all anchor(<a>) tag to “Navlink” and “href” to “to”

eg:

    <a class="navbar-brand" href="/">Navbar</a>

To

    <NavLink class="navbar-brand" to="/">Navbar</NavLink>

Change “class” to “className”

Video 20

Add css by directly doing changes in already given App.css(if require remove all given contents)

Style,font etc all will be handle just import it on App.js

Video 28

Create a error page components/Errorpage.js

And create route in App.js for error page

 <Route path="\*" element={<><Navbar /><div className='container'><Errorpage /></div></>} />

Video 29

Get changes from frontend signup page using changeState of react

We use only one useSate for everything

const Signup = () => {

  const [user, setUser] = useState({

    name: "",

    email: "",

    phone: "",

    work: "",

    password: "",

    cpassword: "",

  });

let name,value;

const handleInputs = (e) => {

*// console.log(e);*

  name = e.target.name;

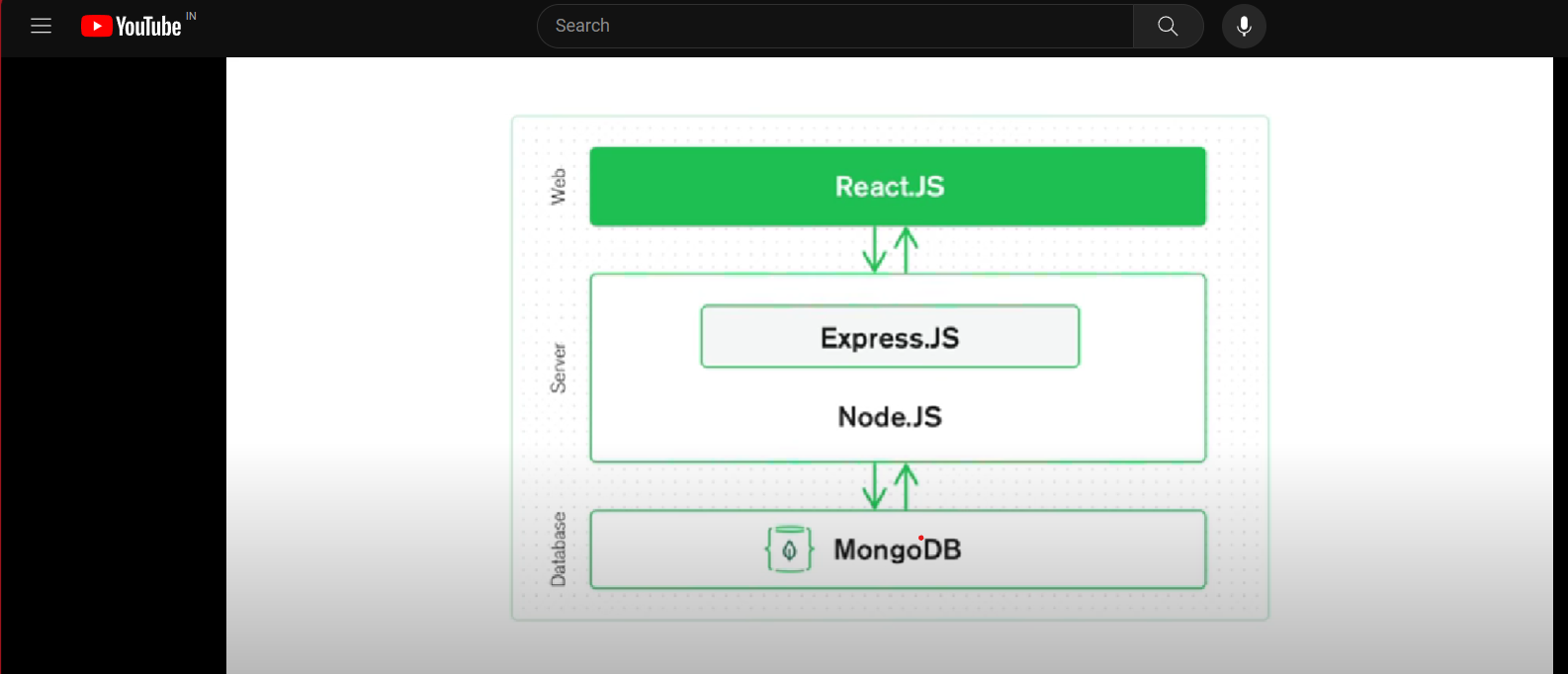
  value = e.target.value;

setUser({...user, [name]:value})

};

We can check changes in inspect+components

Video 30



Change client port from package.json

 "scripts": {

    "start": "set PORT=5000 && react-scripts start", // SETTING DIFFERENT PORT FROM SERVER

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

**We need to add proxy port in client/package.json for telling the client to send request for registration in server port 3000**

**Start client and server together**

**Server+cmd-> npm install cors**

**CORS (Cross-Origin Resource Sharing): Ensure that your server (running on http://localhost:3000) is configured to allow requests from your React application (running on a different port, like http://localhost:5000 by default). You may need to configure CORS on your server to allow requests from the client's origin.**

Do changes in server/app.js

const express = require('express');

const cors = require('cors');

const app = express();

// Use CORS middleware

app.use(cors());

// ... rest of your server setup

**Do**

**Do**      <form method="POST">

**Video 32**

1. Authentication in MERN
2. Verifying About Page for authentication
3. Get About Page Dynamic Data

Now we will start working on server first

Server create folder middleware + create file authenticate.js

Just backend completed

Video 33 (main video verifying for about page)

For cookie

Server+cmd -> npm install cookie-parser

For sending cookie from frontend to backend

set

 <form method="GET">

For about.js

Video 34

To get asccess to user data we cant directly ascess its data, we need to use state in react called **usedState**

The **useState** hook allows you to manage and update the state of user-related data within functional components in a React application.

 const [userData, setUserData] = useState();

*// work for authentication ----------------------*

  const navigate = useNavigate();

 const [userData, setUserData] = useState();  *//useState({}) declaring empty object*

  const callAboutPage = async () => {

    try{

const res = await fetch('/about',{

  method:"GET",

  headers:{

    Accept:"application/json",

    "Content-Type" :"application/json"

  },

*// to send cookie to backend*

  credentials:"include"

});

const data = await res.json();

console.log(data);

setUserData(data);

if(!res.status === 200 ){

  const error = new Error(res.error);

  throw error;

}

    }

    catch(err){

console.log("Error from Aboutjs ----------> ",err);

navigate('/login');

    }

  }

*// it will get call automatically when about us page is visited*

  useEffect(() => {

    callAboutPage();

  }, []);

*// end of authentication -------------------------------------------------------------*

Face error in About.js

<h5>{userData.name}</h5>

ERROR

Cannot read properties of undefined (reading 'name')

TypeError: Cannot read properties of undefined (reading 'name')

Solution :

ChatGPT

The error "Cannot read properties of undefined (reading 'name')" typically occurs when you are trying to access a property (name in this case) of an object that is undefined. This often happens when you are trying to access a property of an object before ensuring that the object is defined.

Async Data Loading:

If you are fetching data asynchronously and rendering the component before the data is available, the initial state of userData might be undefined. Make sure you have a check in place before trying to access properties:

<h5>{userData && userData.name}</h5> ---------------🡪issue solved

**Video 35**

For contact page we need to create a new route for ascessing data it will be used for both contact & home page

Go to server/route/auth.js

And create route -

*// this path is for both contact and home page*

router.get('/getdata',authenticate , (req,res) => {

    console.log('getting data from auth js');

    res.send(req.rootUser);

});

**Video 36**

Now we have to save the message of contact page in db

So we need to first save data in STATE like this

 value={userData.message}

 onChange={handleInputs}

Now for getting the value like name, email from textbox we add new attribute

  name="name"

  onChange={handleInputs}

Now send the data backend for message 11:40

16:30

Now remove the sample path from server/app.js of “/contact” and move to server/router/auth.js

Change userSchema.js for storing message