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

For creating: npm init -y

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);*

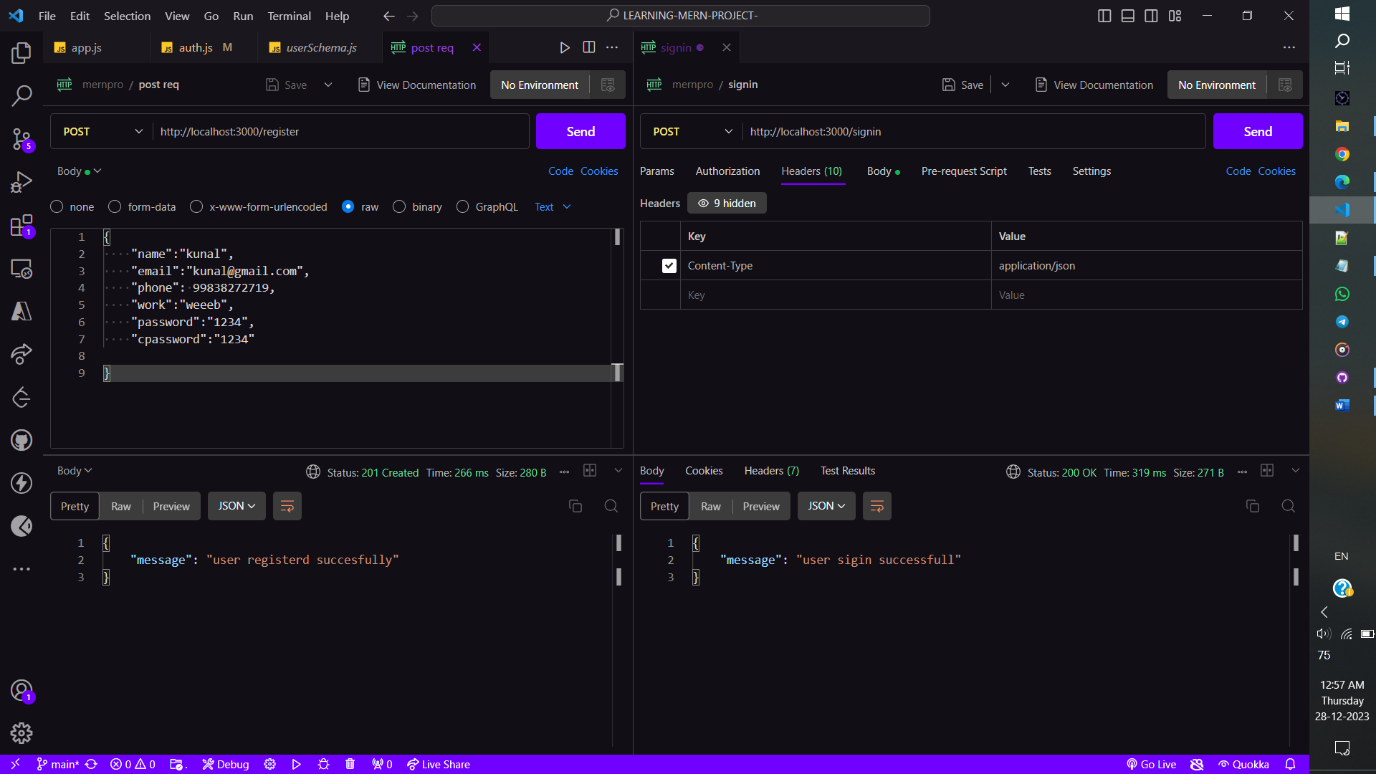
*// res.json({message:req.body});*

*//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**

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