**Day 50,51 – theory**

**Day 52- fs module, os modules, http module**

**Types of modules**

Creation of node

Creating /home,/about endpoints in nodejs

**Day 53**- postmon and nodemon

**Express js- framework of nodejs**

To save time ,simpler, less lines of code.

Npm init

Npm I express

const express=require('express');

const app=express();

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

    res.send('hello home ')

})

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

    res.send('hello about ')

})

app.listen(3000,()=>{

    console.log('server running')

})

**Routing params:**

Whenever the data comes from client , it goes to req obj,

The data sending from server – will be res obj

Lets say im dng – localhost:3000?name=zeenath

How to get this data?

Console.log(req. query)

Console.log(req.query.name)

**Req.params( dynamic routing)**

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

    res.send('hello '+ req.params.name)

})

**Rendering html and json data :**

const express=require('express');

const app=express();

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

    res.send('<h1>Welcome to home page</h1>')

})

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

    res.send(`

    <input type="text" placeholder="username" />

    <button >Click me</button>

    `)

})

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

    res.send({

        name:'zeenath',

        class:'123'

    })

})

//more than one json obj then use array

app.get('/json2',(req,res)=>{

    res.send([{

        name:'zeenath',

        class:'123'

    },

    {

        name:'helko',

        class:859

    }])

})

app.listen(3000,()=>{

    console.log('server started')

})

linking page

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

    res.send(`

    <h1>Welcome , to link page </h1>

    <a href='/' >Go to home page </a>`

    )

})

**// to take query param into input field**

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

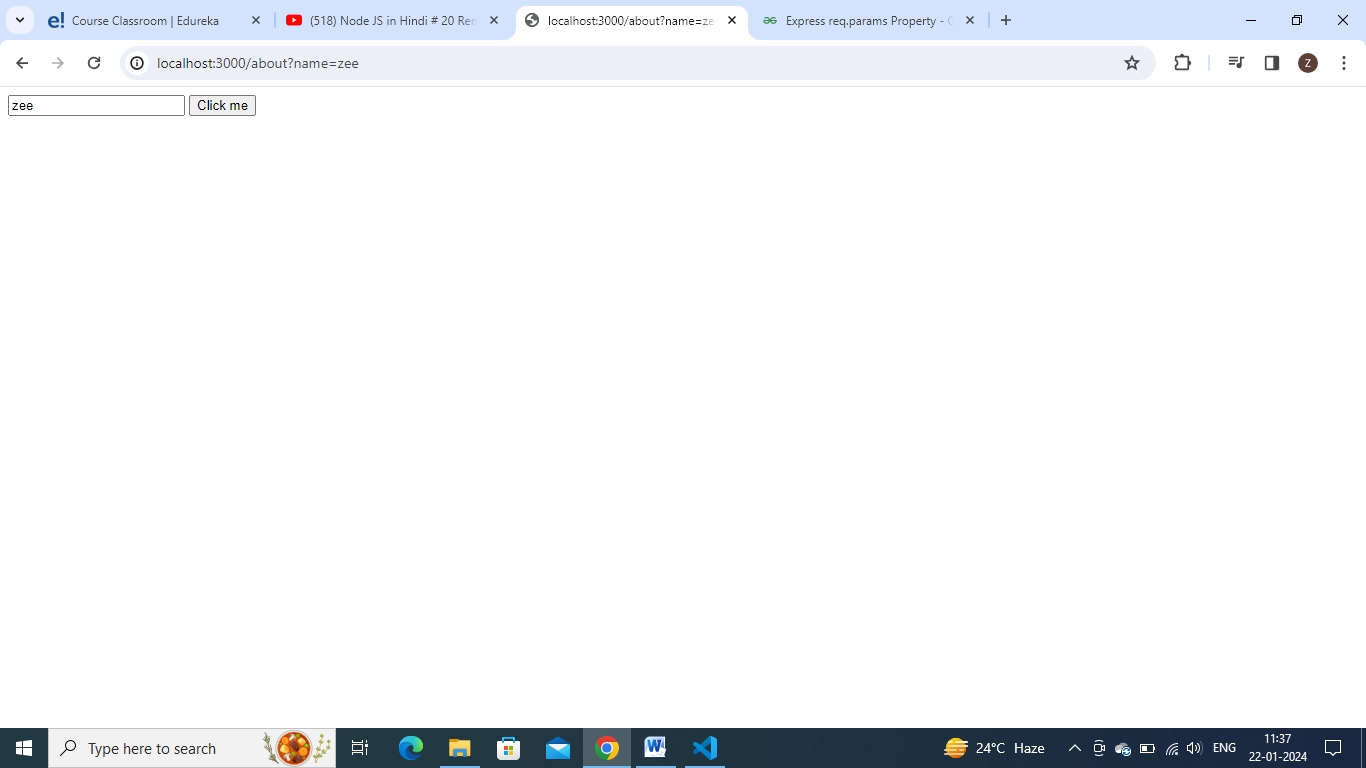
    res.send(`

    <input type="text" placeholder="username" value="${req.query.name}" />

    <button >Click me</button>

    `)

})



**//make html pages refer express 2 folder**

**To**tal html page creating like this in string is not feasible

Make separate html page and render

**Index.js**

const express =require('express');

const app=express();

app.listen(3000,()=>{

    console.log('server running')

})

Create a public folder, and create home,about.html pages inside it.

To load this public folder, set path for public folder

Console.log(\_\_dirname) it gives present folder name

To add /public into \_\_dirname – use path module

const path=require('path')

const publicPath=path.join(\_\_dirname,'public');

console.log(publicPath)

express.static middleware will load static files from public folder

localhost:3000/ - will automatically render index.html (default page)

localhost:3000/about.html – will render about.html page

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

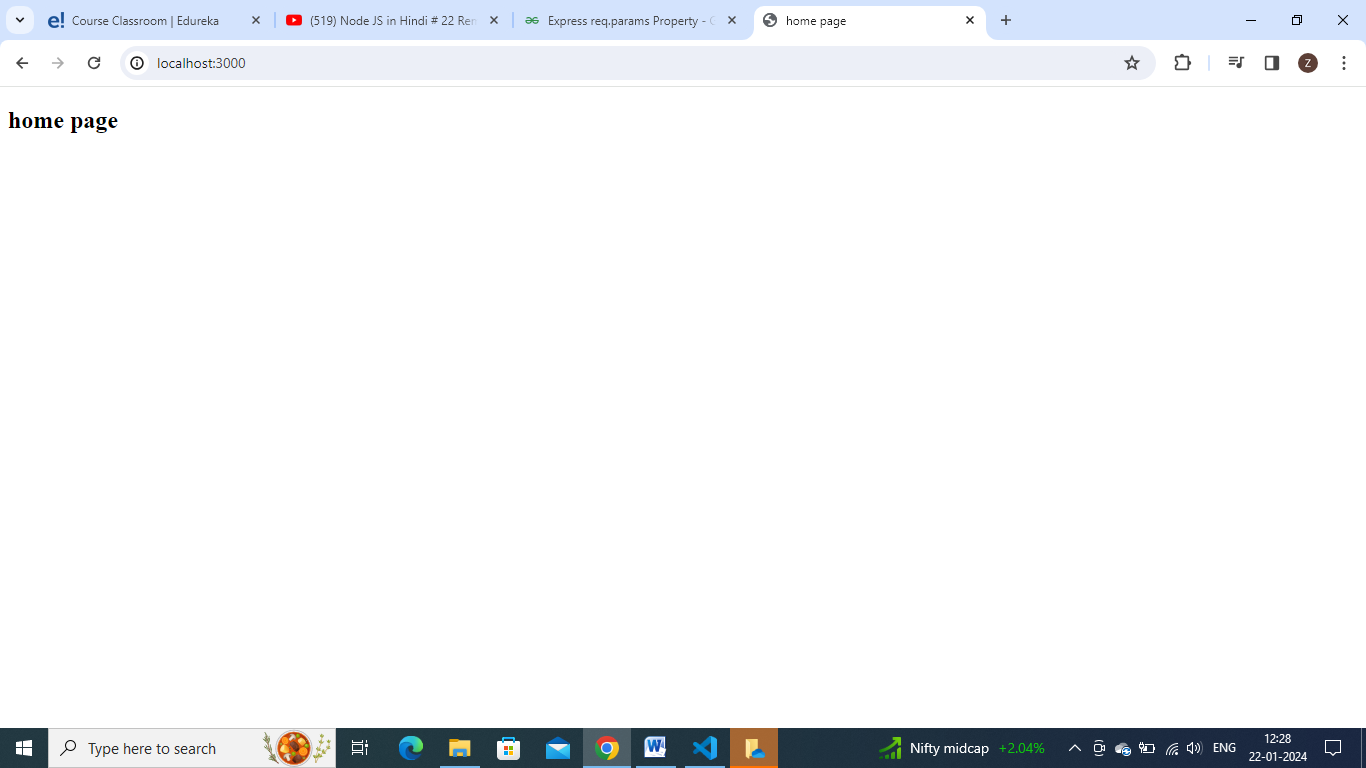
    res.sendFile(`${publicPath}/index.html`)

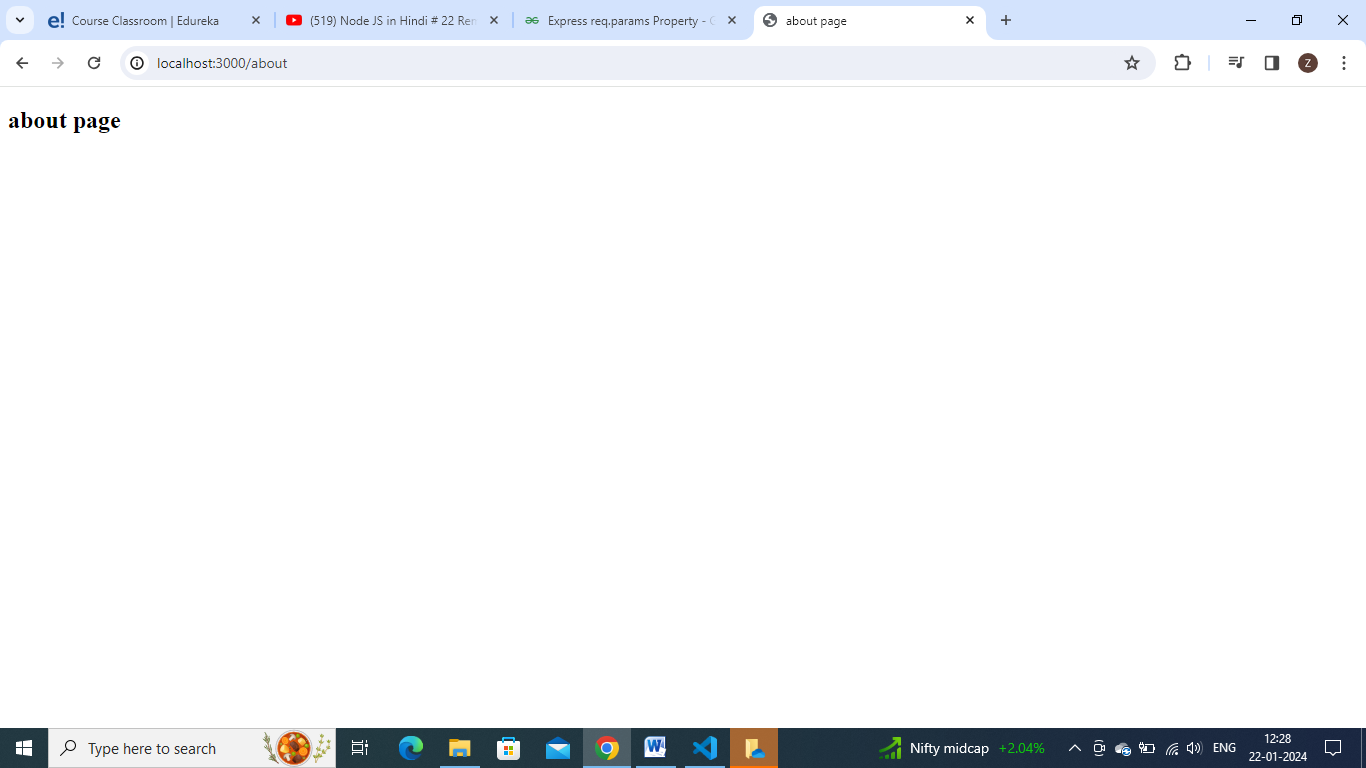
})

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

    res.sendFile(`${publicPath}/about.html`)

})





**So now no need of /about.html extension**

app.get('\*’,(req,res)=>{

    res.sendFile(`${publicPath}/err.html`)

})

**Template engine: - pug, ejs,handlebars(hbs)**

**Refer ejs express 3 folder:**

**Name ,img and all -**  getting from db and displaying in our browser => dynamic page

As it is which we wrote in our page is displaying in browser => static page

**Ejs**:

Npm init

Npm I express ejs

Create basic express file

const express=require('express');

const app=express();

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

    res.send('hello world')

})

app.listen(3000,()=>{

    console.log('server is started')

})

Telling our express app that we are using ejs in it => it is known as setting viewengine as ejs

const express=require('express');

const path=require('path');

const app=express();

app.set('view engine',”ejs”)

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

    res.send('hello world')

})

app.listen(3000,()=>{

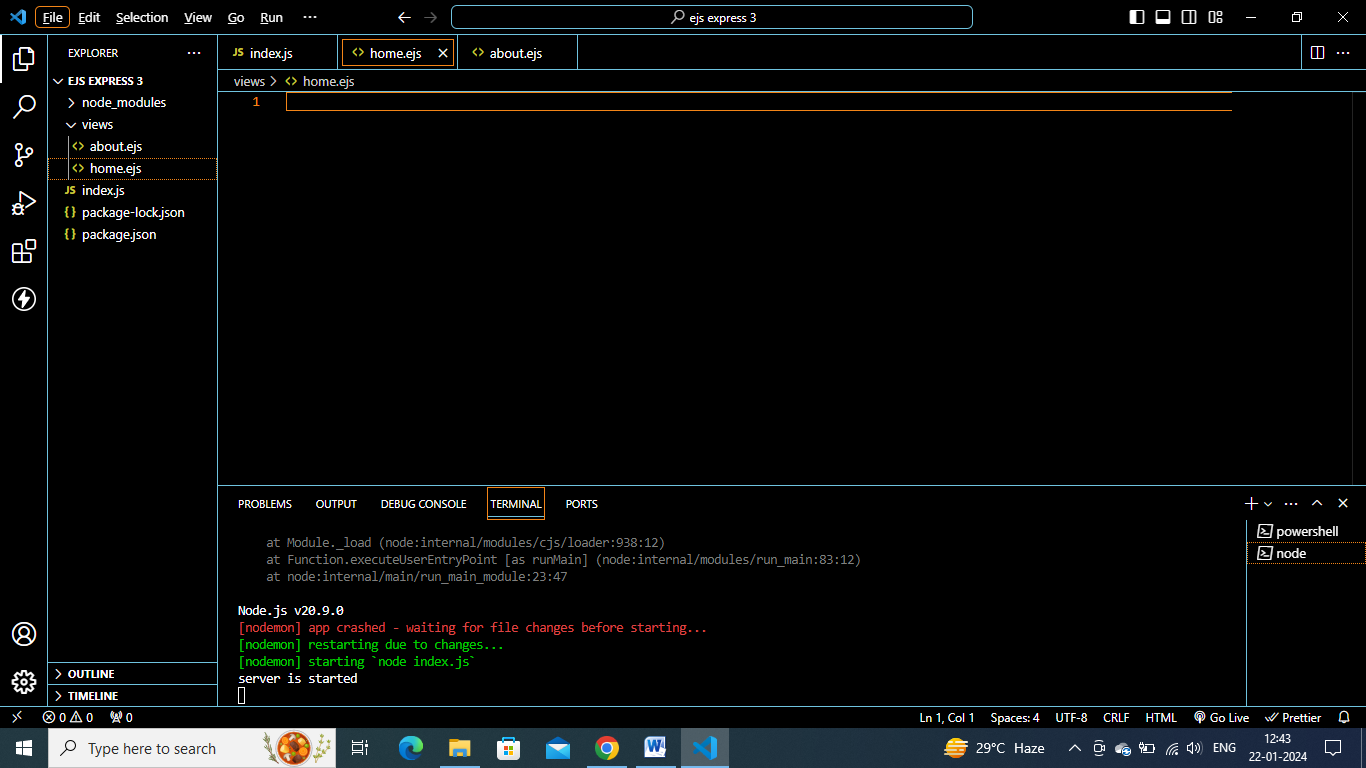
    console.log('server is started')

})

Added app.set(‘view engine’, ‘ejs’) line

Create folder views => our all ejs files must be kept in views folder

By default our express file search for ejs file s inside views folder only



**Home.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>home page</title>

</head>

<body>

    <h2>homepage ejs</h2>

</body>

</html>

**Index.js**

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

    res.render(‘home’)

})

const express=require('express');

const path=require('path');

const app=express();

app.set('view engine','ejs')

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

    const name='zee';

    res.render('home',{name: name})

})

app.listen(3000,()=>{

    console.log('server is started')

})

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>home page</title>

</head>

<body>

    <h2>homepage ejs <%= name%></h2>

</body>

</html>

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

    const user={

        name:'myname',

        class:'123',

        area:'hyd'

    }

    res.render('home',{user:user})

})

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>home page</title>

</head>

<body>

    <h2>homepage ejs <%= locals.name%></h2>

    <p>page user <%= locals.user.class %></p>

</body>

</html>

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

    const user={

        name:'myname',

        class:'123',

        area:'hyd',

        skills:['php','js','jhdjs']

    }

    res.render('home',{user:user})

})

To iterate it

**Home.ejs**

<body>

    <h2>homepage ejs <%= locals.name%></h2>

    <p>page user <%= locals.user.class %></p>

    <ul>

        <% user.skills.forEach((item)=>{ %>

            <li><%= item %></li>

        <% })%>

        })

    </ul>

</body>

Including navabar.ejs in all files

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>navbar</title>

</head>

<body>

    <h2>

        header home about page

    </h2>

</body>

</html>

Create navbar.ejs inside views>common>navbar.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>home page</title>

</head>

<body>

    <%= include('common/navbar'); %>

    <h2>homepage ejs <%= locals.name%></h2>

    <p>page user <%= locals.user.class %></p>

    <ul>

        <% user.skills.forEach((item)=>{ %>

            <li><%= item %></li>

        <% })%>

    </ul>

</body>

</html>

    <%= include('common/navbar'); %>

Instead of = give – then html get rendered

**Middlewares:**

Middlewares are funcs used along with routes to modify req and res

Sometimes , we can check user authenticated ornot , check age before access

Can write once and use in any no of routes when required

**Create index.js express app:**

const express=require('express');

const app=express();

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

    res.send('home page')

})

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

    res.send('about page')

})

app.listen(3000,(req,res)=>{

    console.log('server started')

})

**Creating middleware**

const express=require('express');

const app=express();

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

    console.log('im a middleware')

}

app.use(middle);

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

    res.send('home page')

})

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

    res.send('about page')

})

app.listen(3000,(req,res)=>{

    console.log('server started')

})

**Check output- it keeps on loading but in console – u get im a middleware**

**So middleware got executed, but cntrl did not went to next line**

**Call next() in middleware then only it go to next.**

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

    console.log('im a middleware')

    next()

}

**Here middleware will be applicable to all routes.**

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

    // console.log('im a middleware')

    if(!req.query.age){

        res.send('please provide age')

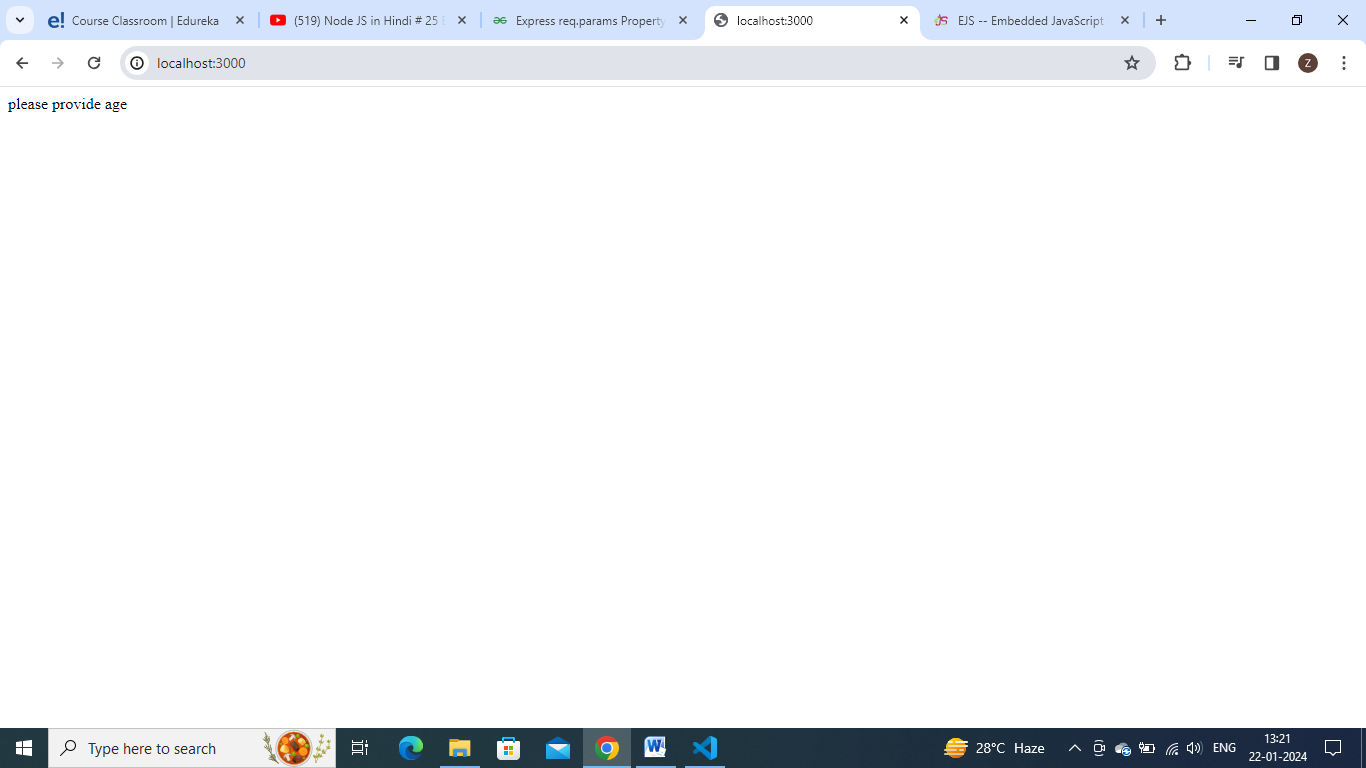
    }

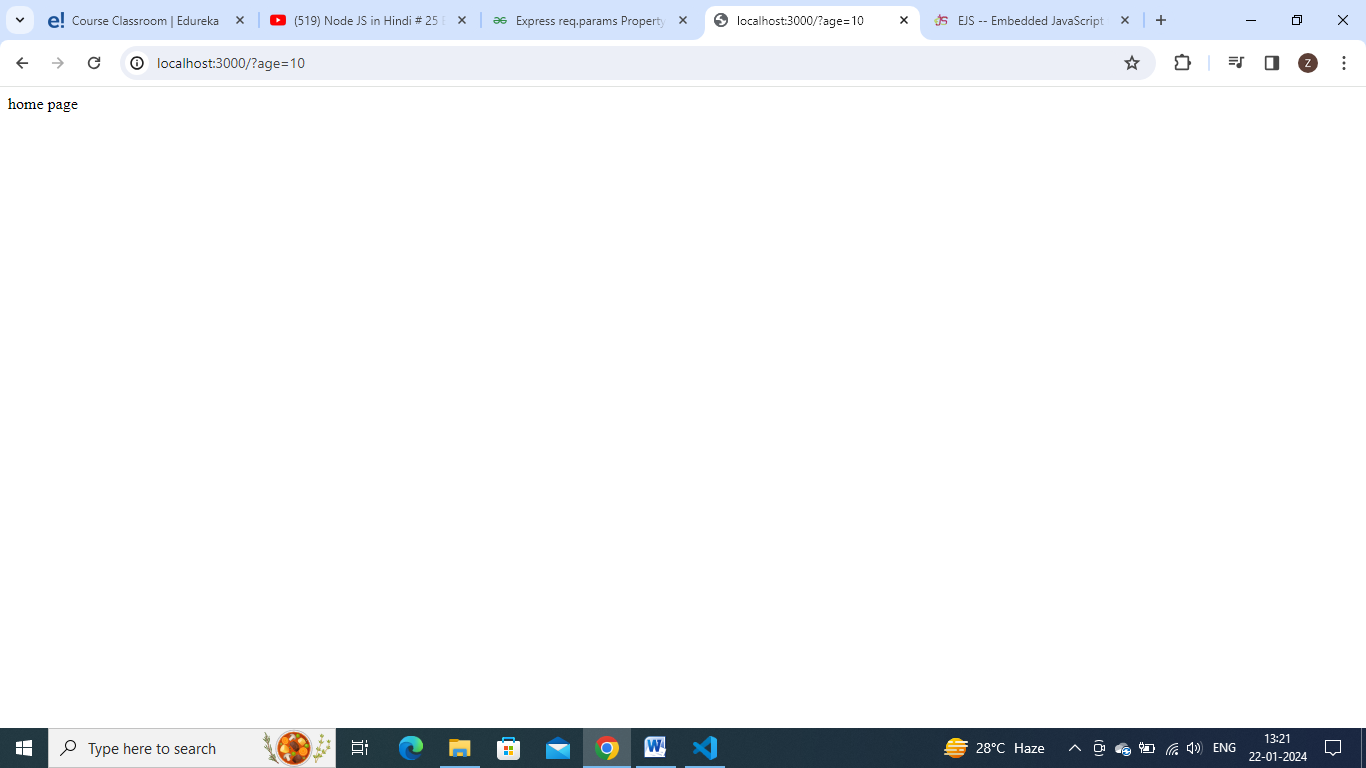
    else{

    next()

    }

}





const express=require('express');

const app=express();

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

    // console.log('im a middleware')

    if(!req.query.age){

        res.send('please provide age')

    }

    else if(req.query.age<18 ){

        res.end(' u cannot access  this page')

    }

    else{

        next()

    }

}

app.use(middle);

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

    res.send('home page')

})

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

    res.send('about page')

})

app.listen(3000,(req,res)=>{

    console.log('server started')

})

This is applicationlevel middleware

The other type is routerlevel middleware

Buildin middlewares

Thirdparty middlewares

**Route level middleware:**

**To add middlware to only particular route**

Comment this line

// app.use(middle);

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

    res.send('about page')

})

Write liekthis for a particular route to which u needed

**To keep middleware in separate file**

**Middleware.js**

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

    if(!req.query.age){

        res.send('please provide your age')

    }

    else if( req.query.age < 18){

        res.send('u are under  aged')

    }

    else{

        next();

    }

}

**Add the below line in index.js**

const middle=require('./middleware')

**middlewares can be any number of in our project – so separate file will be best practice**

**Routes:**

**To call this middleware in 2 , 3 places at 2,3 endpoints.**

**Index.js**

**Add this line**

const middle=require('./middleware')

const route=express.Router()

route.use(middle)

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

    res.send('home page')

})

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

    res.send('about page')

})

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

    res.send('about page')

})

app.use('/',route)

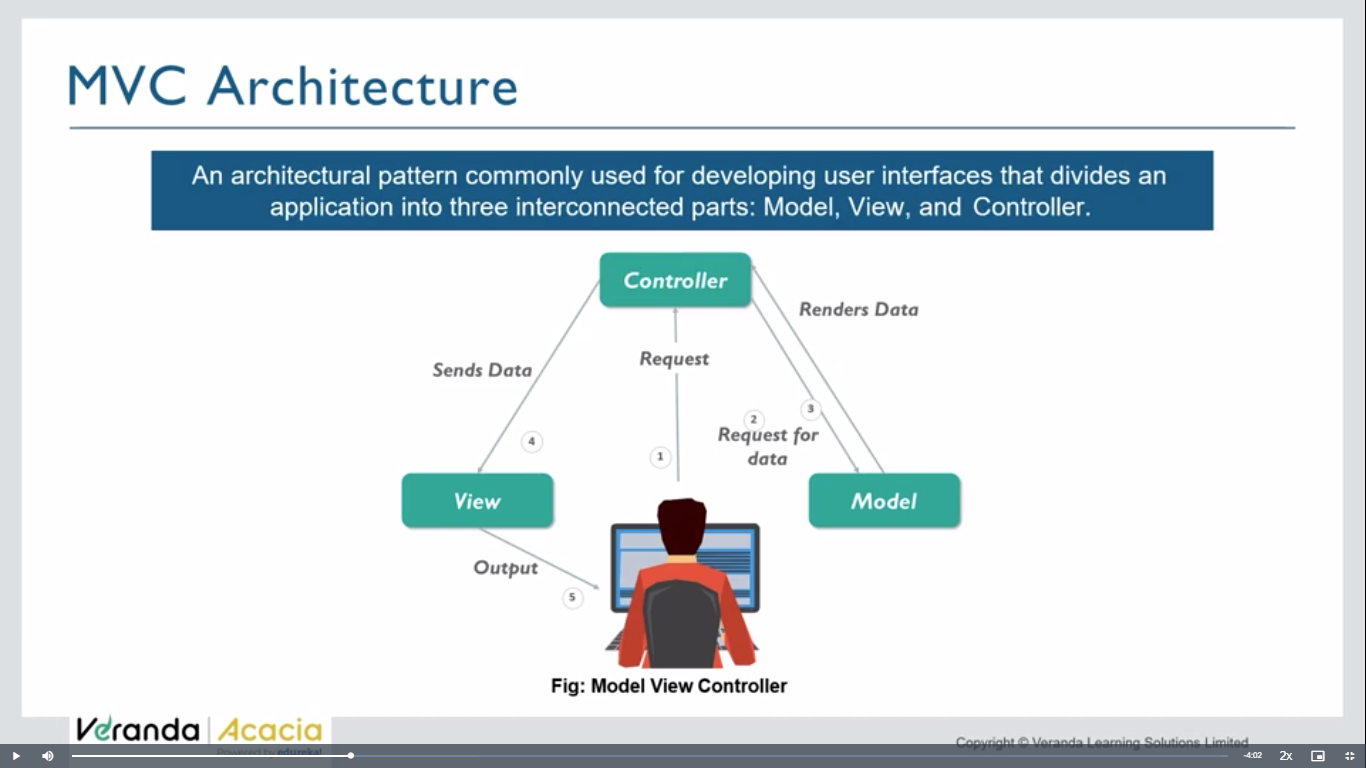
app.listen(3000,(req,res)=>{

    console.log('server started')

})

**Suppose I wants to access middleware on only about and contact page, then I can use route.get on those places.**

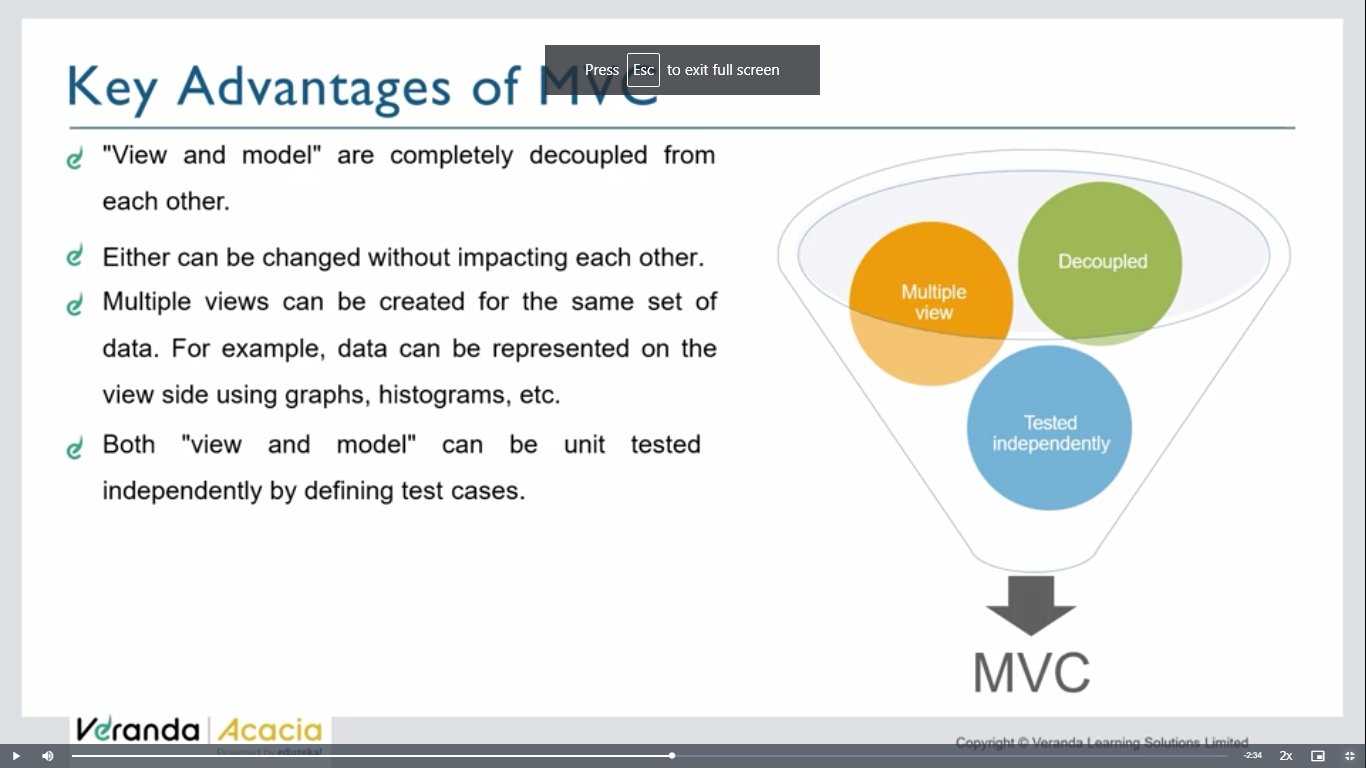
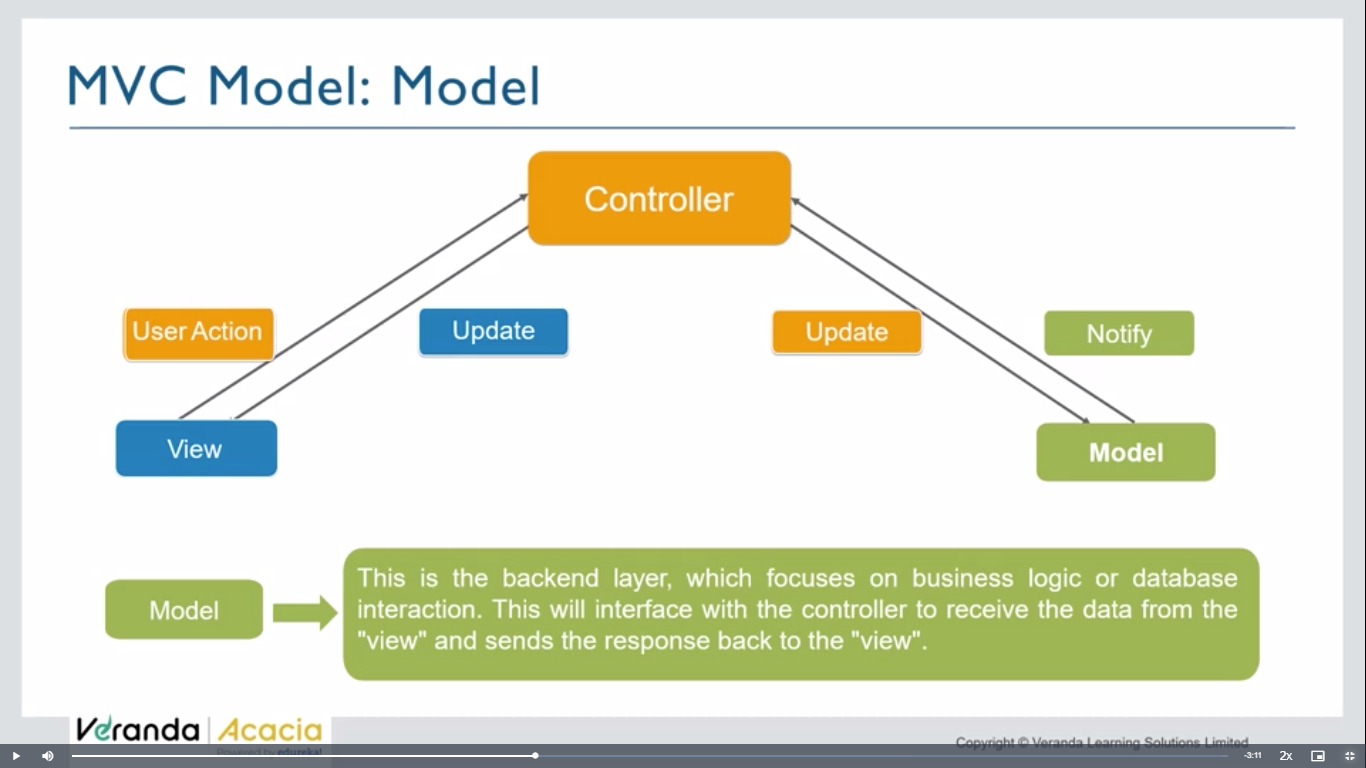
**MVC architecture:**



**View layer- presentation layer- which user sees**

**Controller- takes req from user, see the appropriate handler method , execute it and sends back response to the client**

**Model- used to model / structure the data present in database. Store and interact db in structure way**



**In mern stack – we use react to create view layer**

**What is MVC?**

The [**Model-View-Controller (MVC)**](https://www.geeksforgeeks.org/mvc-design-pattern/) framework is an architectural/design pattern that separates an application into three main logical components **Model**, **View**, and **Controller**. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. It was traditionally used for desktop **graphical user interfaces (GUIs)**. Nowadays, MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also used for designing mobile apps.

MVC was created by **Trygve Reenskaug**. The main goal of this design pattern was to solve the problem of users controlling a large and complex data set by splitting a large application into specific sections that all have their own purpose.

### ****Controller:****

The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn’t have to worry about handling data logic, it just tells the model what to do. It processes all the business logic and incoming requests, manipulates data using the **Model**component, and interact with the **View**to render the final output.

### ****View:****

The **View**component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren’t taken directly but through the controller. It only interacts with the controller.

### ****Model:****

The **Model**component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller’s request because the controller can’t interact with the database by itself. The model interacts with the database and gives the required data back to the controller.

**Index.js**

const express=require('express');

const app=express();

const rest = require('./models/restaurant.json');

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

    res.send('hello world')

})

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

    res.json(rest)

})

app.listen(3000,()=>{

    console.log('server started')

})

**Index.js**

const express=require('express');

const app=express();

const rest = require('./models/restaurant.json');

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

//     res.send('hello world')

// })

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

//     res.json(rest)

// })

app.listen(3000,()=>{

    console.log('server started')

})

**Routes>user.js**

const express=require('express');

const app=express();

const router=express.Router();

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

    res.send('router req')

})

module.exports=router

**index.js**

const userRoute=require('./routes/user');

app.use('/user',userRoute)

**models>restaurant.json**

**add fake json data**

**index.js**

const rest = require('./models/restaurant.json');

const express=require('express');

const app=express();

const rest = require('./models/restaurant.json');

const userRoute=require('./routes/user');

app.use('/user',userRoute)

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

//     res.send('hello world')

// })

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

    res.json(rest)

})

app.listen(3000,()=>{

    console.log('server started')

})

**At ‘/’ endpoint get json data**