Version 3

task 37: Authentication:

Ab hum padhne wale hai ***middlewares*** jaise **express.static** and **express.urlencoded** ye sab joo hum use krte hai all these are middlewares.

💡 middleware function ke andar we have access of 3 things **req,res,next** iska matlab hota hai ki hum middleware function ke andar hi **req** ko **res** ko manipulate kr sakte hai.
and iske andar is next ka matlab hota hai ki ab jab aapka saara manipulation ya jo bhi kaam tha vo hogya to hum aage kya krenge?

```
//middlewarefolder --> index.js

let express = require('express');
let app = express();


app.use( (req,res,next)=>{
    // res.send('namaste ji ab kaunsa route dhoondh rhe ho');
    console.log("jo krne aae they kro aur fir code ko aage jane do");
    next();
})

const verify = (req,res,next)=>{
    let {password} = req.query;
    if(password !== "orange"){
        return res.send('invalid password');
    }
    else{
        return next();
    }
}

app.get('/' , (req,res)=>{
    res.send('home route');
})
app.get('/secret' , verify , (req,res)=>{
    res.send('mai kabhi batla tha nhi par teri parwah krta hu mai maa');
})

app.listen(2323 , ()=>{
    console.log("server connected at port 2323")
})
```

above is the example of kab middleware chalega and kab uska next chalega.

💡 **next()** chalne ke baad agar koi line likhi hai to bhi we can see the code. agar isse bachna hai to simply **return next().**

task 38: ab agar mai ek product delete krta hu to i want ki uske corresponding saare details delete hojaae i.e reviews as well.

uska matlab ki jab mai product ko delete kar rha hu usko delete krne se just pehle mujhe uske andar for loop lagake saare reviews ko hatana hoga.

```
//productRoutes.js
```

```
//delete a route
router.delete('/products/:id' , async(req,res)=>{
    let {id} = req.params;
    const product = await Product.findById(id); //added

    for(let id of product.reviews){      //added full loop
        await Review.findByIdAndDelete(id);
    }

    await Product.findByIdAndDelete(id); //added
    res.redirect('/products');
})
```

this is also a way par ye ideal way nhi hai hum isse aur better bana sakte hai by using **middlewares**.

💡 Ab tk jo humne middleware dekhe vo express ke middleware they lekin jo ab hum dekhenge vo mongoose ke middleware honge jo bohat different hai from express.

acording to the documentation we have 2 types of middleware:

1. pre
2. post

task 39: Ab hum ye middleware lagate hai schema par

```
//product.js (schema)

productSchema.pre('findOneAndDelete' , async function(data){
    console.log("pre middleware");
    console.log(data);
})

productSchema.post('findOneAndDelete' , async function(data){
    console.log("post middleware");
    console.log(data);
})


let Product = mongoose.model('Product' , productSchema);
module.exports = Product;
```

ab hume krna hai ki productSchema.post middleware method mei simply hum reviews ki array ko delete kre.

```
//product.js (schema)
//pre ka koi kaam nhi filhaal
const Review = require('./review'); //adding

productSchema.post('findOneAndDelete' , async function(product){ //changing
    if(product.reviews.length > 0){
        await Review.deleteMany({_id:{$in:product.reviews}})
    }
})
```

task40:

ab humara kaam hai ki validation kre and validation keliye we have 2 things:

1. client side validation
2. server side validation

client side pr validate krne ke liye i can use an attribue **requied.** in new.ejs.

also hum **bootstrap** ka use kar sakte hai to do same validation.

(bootstrap —> form —> validation)

1. **required** sabme likho.
2. ab required likhne se bootstrap ka native validation hota hai jo yellow krke aarha hai na and hum nhi chahahte **native/default validation,** to hum **novalidate attribute** use krenge**.**

```
//new.ejs
<form action="/products" method="POST" novalidate> //added here
```

ab hume iske logic ke liye script chahiye to hum isse add kr sakte hai neeche hi.

```
//NEW.EJS

<% layout('layouts/boilerplate') %>


<div class="row">
    <div class="col-6 mx-auto"> //ADDING CLASS AND ATTRIBUTE
        <form action="/products" method="POST" class="needs-validation" novalidate>
            <div class="mb-3">
                <label for="naam" class="form-label">Name: </label>
                <input type="text" class="form-control" name="name" id="naam" placeholder="Name of Product" required>
//ADDED THESE 2
<div class="valid-feedback">
    Looks good!
</div>
<div id="validationServer03Feedback" class="invalid-feedback">
    Please provide a valid city.
</div>
            </div>
            <div class="mb-3">
                <label for="imge" class="form-label">Image Url: </label>
                <input type="text" class="form-control" name="img" id="imge" placeholder="Image URL of Product" required>
                    <div class="valid-feedback">
                        Looks good!
                    </div>
            </div>
            <div class="mb-3">
                <label for="paisa" class="form-label">Price: </label>
                <div class="input-group mb-3">
                    <span class="input-group-text" id="basic-addon1">Rs. </span>
                    <input class="form-control" type="number" name="price" id="paisa" step="any" placeholder="Price of Product" required>
                        <div class="valid-feedback">
                            Looks good!
                        </div>
                </div>
            </div>

            <div class="mb-3">
                <label for="des" class="form-label">Description: </label>
                <textarea class="form-control"  name="desc" id="des" rows="5" placeholder="Description of Product" required></textarea>
                    <div class="valid-feedback">
                        Looks good!
                    </div>
            </div>
            <button type="submit" class="btn btn-sm btn-success">Add</button>
        </form>
    </div>
</div>


//ADDED JS
<script>
    (() => {
  'use strict'

  // Fetch all the forms we want to apply custom Bootstrap validation styles to
  const forms = document.querySelectorAll('.needs-validation')

  // Loop over them and prevent submission
  Array.from(forms).forEach(form => {
    form.addEventListener('submit', event => {
      if (!form.checkValidity()) {
        event.preventDefault()
        event.stopPropagation()
      }

      form.classList.add('was-validated')
    }, false)
  })
})()

</script>
```

💡 The above function is called **IIFE (Immidiately invoked function expression)**

ye to hogya client side validation ab we will do server side validation.

aap ye saari cheeze **edit.ejs** mei bhi krdena.

task 41: ab hum **error handling** ke liye cheezo ko **try and catch** mei likh rhe hai.

jisme mai apna error.ejs banaunga just under **error.ejs**

```
//productRoutes.js

const express =  require('express');
const Product = require('../models/product');
const Joi = require('joi'); //added
const router = express.Router();

// displaying all the products
router.get('/products' , async(req,res)=>{
    try{
        let products = await Product.find({});
        res.render('products/index' , {products});
    }
    catch(e){
        res.status(500).render('error' , {err:e.message});
    }

})


// adding a fomr for  anew product
router.get('/products/new' , (req,res)=>{
    try{
        res.render('products/new');
    }
    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})
```

```
// actually adding a product in a DB
router.post('/products' , async (req,res)=>{
    try{
        let {name,img,price,desc} = req.body;
        await Product.create({name,img,price,desc});
        res.redirect('/products');
    }

    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})

// route for shwoing the deatails of thre products
router.get('/products/:id' , async(req,res)=>{
    try{

        let {id} = req.params;
        // let foundProduct = await Product.findById(id);
        let foundProduct = await Product.findById(id).populate('reviews');
        // console.log(foundProduct);
        res.render('products/show' , {foundProduct});
    }

    catch(e){
        res.status(500).render('error' , {err:e.message});
    }

})

// route for editing the product so we need form for it
router.get('/products/:id/edit' , async(req,res)=>{
    try{

        let {id} = req.params;
        let foundProduct = await Product.findById(id);
        res.render('products/edit' , {foundProduct});

    }
    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})

// changing the original edits in the database made in the editform
router.patch('/products/:id' , async(req,res)=>{
    try{

        let {id} = req.params;
        let {name,img,price,desc} = req.body;
        await Product.findByIdAndUpdate(id , {name,img,price,desc});
        res.redirect(`/products/${id}`)
    }

    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})

//delete a route
router.delete('/products/:id' , async(req,res)=>{
    try{

        let {id} = req.params;
        // const product = await Product.findById(id);

        // for(let id of product.reviews){
        //     await Review.findByIdAndDelete(id);
        // }

        await Product.findByIdAndDelete(id);
        res.redirect('/products');
    }

    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})
```

```
module.exports = router;
```

```
//views --> error.ejs

<% layout('layouts/boilerplate') %>

<div class="alert alert-danger" role="alert">
    <%=err%>
</div>
```

task 42: server side validation.

here we will use ek package named as **JOI** it is the most powerful **schema description language** and **data validator for javascript**.

—> Ab isme **2 steps** hote hai pehle defining a schema in the place where use krna hai and doosra hai validating that schema.

validation ko hum alag se krlenge just like seed.js i.e **schmea.js** so that code bada na hojaae.

and usko middleware ke andar daal denge **(file: middleware.js)** taaki product creation route waale function ko run krne se pehle we can use the validation of **schema joi** middleware.

—>ab jahan aapne product ko validate krvane ke baad waala middleware ko use krna hai i.e productRoutes ke andar vahan require krke simply middleware use krlo

```
//terminal
npm i joi


//schema.js
const Joi = require("joi");

const productSchema = Joi.object({
    name: Joi.string().required(),
    img: Joi.string().required(),
    price: Joi.number().min(0).required(),
    desc: Joi.string().required()
});

const reviewSchema = Joi.object({
    rating:Joi.number().min(0).max(5),
    Comment: Joi.string().required()
})

module.exports = { productSchema , reviewSchema } ;
```

same goes for reviews aap **validateReview** bana sakte ho and then usse middleware mei bhejkar simply usse review.js jahan aapne routes likhe hai vahan bhej sakte ho and vahan use kr sakte ho apne middleware ko to evaluate.

```
//middleware.js

const { productSchema } = require("./schema");
const { reviewSchema } = require("./schema");

const validateProduct = (req,res,next)=>{
    const {name, img, price , desc} = req.body;
    const {error} = productSchema.validate({name,img,price,desc});

    if(error){
        const msg = error.details.map((err)=>err.message).join(',');
        return res.render('error' , {err:msg});
    }
    next();
}

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

    const {rating, comment} = req.body;
    const {error} = reviewSchema.validate({rating,comment});

    if(error){
        const msg = error.details.map((err)=>err.message).join(',');
        return res.render('error' , {err:msg});
    }
    next();
}

module.exports = {validateProduct ,validateReview} ;
```

Ab routes mei change krna hai

```
//productRoutes.js

const express =  require('express');
// const Joi = require('joi');
const Product = require('../models/product');
const router = express.Router();
const {validateProduct} =  require('../middleware');  //added here

// displaying all the products
router.get('/products' , async(req,res)=>{
    try{
        let products = await Product.find({});
        res.render('products/index' , {products});
    }
    catch(e){
        res.status(500).render('error' , {err:e.message});
    }

})


// adding a fomr for  anew product
router.get('/products/new' , (req,res)=>{
    try{
        res.render('products/new');
    }
    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})

// actually adding a product in a DB
router.post('/products' , validateProduct , async (req,res)=>{  //added here
    // try{
        let {name,img,price,desc} = req.body;

        // server side validation switched to schema.js
        // const productSchema = Joi.object({
        //     name: Joi.string().required(),
        //     img: Joi.string().required(),
        //     price: Joi.number().min(0).required(),
        //     desc: Joi.string().required()
        // });
        // const {error} = productSchema.validate({name,img,price,desc});
        // console.log(error);

        await Product.create({name,img,price,desc});
        res.redirect('/products');
    // }

    // catch(e){
    //     res.status(500).render('error' , {err:e.message});
    // }
})

// route for shwoing the deatails of thre products
router.get('/products/:id' , async(req,res)=>{
    try{

        let {id} = req.params;
        // let foundProduct = await Product.findById(id);
        let foundProduct = await Product.findById(id).populate('reviews');
        // console.log(foundProduct);
        res.render('products/show' , {foundProduct});
    }

    catch(e){
        res.status(500).render('error' , {err:e.message});
    }

})

// route for editing the product so we need form for it
router.get('/products/:id/edit' , async(req,res)=>{
    try{

        let {id} = req.params;
        let foundProduct = await Product.findById(id);
        res.render('products/edit' , {foundProduct});

    }
    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})

// changing the original edits in the database made in the editform
router.patch('/products/:id', validateProduct, async(req,res)=>{ //added here
    try{

        let {id} = req.params;
        let {name,img,price,desc} = req.body;
        await Product.findByIdAndUpdate(id , {name,img,price,desc});
        res.redirect(`/products/${id}`)
    }

    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})

//delete a route
router.delete('/products/:id' , async(req,res)=>{
    try{

        let {id} = req.params;
        // const product = await Product.findById(id);

        // for(let id of product.reviews){
        //     await Review.findByIdAndDelete(id);
        // }

        await Product.findByIdAndDelete(id);
        res.redirect('/products');
    }

    catch(e){
        res.status(500).render('error' , {err:e.message});
    }
})


module.exports = router;
```

```
//review.js

const express =  require('express');
const Product = require('../models/product');
const Review = require('../models/review');
const {validateReview} = require('../middleware'); //added here

const router = express.Router();
```

```
router.post('/products/:productId/review',validateReview,async(req,res)=>{ //added here
        try{
                let {productId} = req.params;
                let {rating , comment} = req.body;
                const product = await Product.findById(productId);
                // console.log(product);
                // creating a new review
                const review  = new Review({rating , comment}); // let review  = new Review({...req.body})

                // adding review id to product array
                product.reviews.push(review); //mongodb internally isme se id nikaal kr usse push krdega.

                await review.save();
                await product.save();
                res.redirect(`/products/${productId}`)
        }
        catch(e){
                res.status(500).render('error' ,{err:e.message})
        }

})


module.exports = router;
```