Version 1

step 1: setup basic server application

```
//app.js
const express = require('express');
const app = express();
const path = require('path');



app.set('view engine' , 'ejs');
app.set('views' , path.join(__dirname,'views'));
// now for public folder
app.use(express.static(path.join(__dirname,'public')));



const port = 5000;
app.listen(port,()=>{
    console.log(`server connected at port : ${port} `);
})
```

step 2: ab tum database ko connect krloo and uske saath interact krlo.

```
//app.js
const express = require('express');
const app = express();
const path = require('path');
const mongoose = require('mongoose');

//adding here
mongoose.connect('mongodb://127.0.0.1:27017/shopping-sam-app')
.then(()=>{console.log("DB connected")})
.catch((err)=>{console.log(err)})


app.set('view engine' , 'ejs');
app.set('views' , path.join(__dirname,'views'));
// now for public folder
app.use(express.static(path.join(__dirname,'public')));


const port = 5000;
app.listen(port,()=>{
    console.log(`server connected at port : ${port} `);
})
```

step 3: ab models ke folder ke andar jaakr product ka schema bana do

```
//models --> product.js
const mongoose = require('mongoose');

const productSchema = new mongoose.Schema({
    name:{
        type:String,
        trim:true,
        required:true
    },
    img:{
        type:String,
        trim:true,
        default:'/images/product.jpg'
```

```
    },
    price:{
        type:Number,
        min:0,
        default:"missing",
        required:true
    },
    desc:{
        type:String,
        trim:true
    }
})
let Product = mongoose.model('Product' , productSchema);
module.exports = Product;
```

step 4: seed ki file banao jisme array ho and insertMany() kro so that you can add the product there.

```
//seed.js
const mongoose = require('mongoose');
//seeding ke liye model/collection
const Product = require('./models/product');

const products = [
    {
        name:"iphone 14pro",
        img:'https://images.unsplash.com/photo-1491933382434-500287f9b54b?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MTV8fGFwcGxlfGVufDB8f
        price: 140000,
        desc: "bohat mahenga"
    },
    {
        name:"macbook m2",
        img:'https://images.unsplash.com/photo-1491933382434-500287f9b54b?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MTV8fGFwcGxlfGVufDB8f
        price: 250000,
        desc: "aukaat ke bahar"
    },
    {
        name:"iwatch",
        img:'https://images.unsplash.com/photo-1491933382434-500287f9b54b?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MTV8fGFwcGxlfGVufDB8f
        price: 70000,
        desc: "useless product"
    },
    {
        name:"ipad",
        img:'https://images.unsplash.com/photo-1491933382434-500287f9b54b?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MTV8fGFwcGxlfGVufDB8f
        price: 88000,
        desc: "badiya cheez"
    },
    {
        name:"airpods",
        img:'https://images.unsplash.com/photo-1491933382434-500287f9b54b?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MTV8fGFwcGxlfGVufDB8f
        price: 27000,
        desc: "vahiyaad thuuu radddi"
    }
]
//seeding
async function seedDB(){
    await Product.insertMany(products);
    console.log("data seeded successfully")
}

module.exports = seedDB;
```

task 5: export krke seedDB ko require kro in app.js and then check your database.

```
//app.js
const express = require('express');
const app = express();
const path = require('path');
const mongoose = require('mongoose');
const seedDB = require('./seed'); //added
```

```
mongoose.connect('mongodb://127.0.0.1:27017/shopping-sam-app')
.then(()=>{console.log("DB connected")})
.catch((err)=>{console.log(err)})


app.set('view engine' , 'ejs');
app.set('views' , path.join(__dirname,'views'));
// now for public folder
app.use(express.static(path.join(__dirname,'public')));


// seeding dummy data added
// seedDB();

const port = 5000;
app.listen(port,()=>{
    console.log(`server connected at port : ${port} `);
})
```

task 6: ab hume routes folder banana hai for **productRoutes.js**

```
//routes --> productRoutes
const express =  require('express');
const Product = require('../models/product');
const router = express.Router();

router.get('/products' , async(req,res)=>{
    let products = await Product.find({});
    res.render('products/index' , {products});
})


module.exports = router;
```

task 7: views ke folder ke andar ek aur folder banao **products** jisme saare product ke template honge.

```
//views --> products --> index.ejs //done according to the boilerplate.js file
<% layout('layouts/boilerplate') %>
<ul>
    <% for(let item of products){ %>
        <li>
            <img src="<%=item.img%>">
            <h3><%= item.name %></h3>
            <h5><%= item.price %></h5>
            <p><%= item.desc %></p>
        </li>
    <% } %>
</ul>
```

you can use this way as well lekin what we can do is since hum ejs use kr rhe hai

ab partials ke folder ki jagah aap **ejs mate** ka use kar sakte ho (google).

ejs —> templating language hai

ejs-mate —> templating engine

(express ke andar default engine exist krta hai jo is ejs ko read kar sakta hai. if you want you can change it)

```
//terminal
npm i ejs-mate
```

or

task 8: views ke folder ke andar aap partials naam ka folde create kar sakte ho and **header**, **footer**, **navbar** daal sakte ho usme

```
//views --> partials --> header.ejs || footer.ejs || navbar.ejs
ignore if using ejs-mate
```

```
//app.js
const express = require('express');
const app = express();
const path = require('path');
const mongoose = require('mongoose');
const seedDB = require('./seed');
const productRoutes = require("./routes/productRoutes");
const ejsMate = require('ejs-mate'); //adding

mongoose.set('strictQuery', false);
mongoose.connect('mongodb://127.0.0.1:27017/shopping-sam-app')
.then(()=>{console.log("DB connected")})
.catch((err)=>{console.log(err)})


app.engine('ejs' , ejsMate); //adding
app.set('view engine' , 'ejs');
app.set('views' , path.join(__dirname,'views'));
// now for public folder
app.use(express.static(path.join(__dirname,'public')));
```

task 9: ab aap **views** ke andar ek folder banao **layouts** ka jiske andar aap boiler plate laga sakte ho.

```
//views --> layouts --> boilerplate.ejs
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shopping Cart</title>
</head>
<body>
    <h1>All Products</h1>

    <%- body -%>

    <h4>Footer</h4>

</body>
</html>
```

task 10: ab hume bootstrap use krna hai to we will copy the CSS and JS inside **boilderplate.js**

```
//views --> layouts --> boilerplate.ejs
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- adding bootstrap css-->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-GLhlTQ8i
    <title>Shopping Cart</title>
</head>
<body>
    <h1>All Products</h1>
```

```
    <%- body -%>

    <h4>Footer</h4>


    <!-- adding bootstrap js -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-w76AqPfDkMBDXo30jS1
</body>
</html>
```

```
//views-- > products --> index.ejs
<% layout('layouts/boilerplate') %>
<div class="row">

    <% for(let item of products){ %>

        <div class="col-lg-4">
            <div class="card mt-5 mx-auto" style="width: 18rem;">
                <img src="<%=item.img%>" class="card-img-top" alt="...">
                <div class="card-body">
                  <h3 class="card-title"> <%= item.name %> </h3>
                  <h5 class="card-title"> Rs: <%= item.price %> </h5>
                  <p class="card-text"> <%= item.desc.substring(0,100) %> </p>
                  <a href="#" class="btn btn-primary">View Product</a>
                </div>
            </div>
        </div>

    <% } %>
</div>
```

task 11: ab navbar add kro from the bootstrap

```
// views --> partials --> navabr.ejs
//copied from bootstrap (only changing the backgound-color)
<nav class="navbar fixed-top navbar-expand-lg" style="background-color: #0077b6;">
    <div class="container">
      <a class="navbar-brand" href="/products">Shopping App</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarScroll" aria-controls="navbarScroll" ar
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarScroll">
        <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 100px;">
          <li class="nav-item">
            <a class="nav-link" aria-current="page" href="#">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/products/new">New</a>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
              Link
            </a>
            <ul class="dropdown-menu">
              <li><a class="dropdown-item" href="#">Action</a></li>
              <li><a class="dropdown-item" href="#">Another action</a></li>
              <li><hr class="dropdown-divider"></li>
              <li><a class="dropdown-item" href="#">Something else here</a></li>
            </ul>
          </li>
        </ul>
        <form class="d-flex" role="search">
          <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
          <button class="btn btn-outline-success" type="submit">Search</button>
        </form>
      </div>
    </div>
</nav>
```

task 12: add css to the boilerplate

```
//public --> css --> app.css
//to add margin-top and the shadow from bootstrap

//app.css
main{
    margin-top: 5.5rem;
}


//index.ejs
<div class="card shadow mt-3 mx-auto" style="width: 18rem;"> //added shadow
               <img src="<%=item.img%>" class="card-img-top" alt="...">
```

task 13: ab hume new product add krna hai to pehle uske liye form display krvana hai.

```
//views --> products --> new.ejs

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


<div class="row">
    <div class="col-6 mx-auto">
        <form action="/products" method="POST">
            <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">
            </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">
            </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" placeholder="Price of Product">
                </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"></textarea>
            </div>
            <button type="submit" class="btn btn-sm btn-success">Add</button>
        </form>
    </div>
</div>
```

task 14: route bhi banana hai for adding new form.

```
//routes --> product --> productRoutes

...code...

// adding a fomr for  anew product
router.get('/products/new' , (req,res)=>{
    res.render('products/new');
})
```

task 15: ab **post request** jaa chuki hai aapki form ke through to route for adding a product actually in the db.

```
//routes --> product --> productRoutes
...code...

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

💡 here merko **req.body** ko parse karna hai to i need to have a **middleware** i.e **bodyparser in app.js.**

app.use(express.static(path.join(__dirname,'public')));
app.use(express.urlencoded({extended:true})); //added this

task 16: css ko changing i.e color of btn and align-text-center

```
// public --> css --> app.css

//app.css
:root{
    --main-color: #0077b6;
}
main{
    margin-top: 5.5rem;
}

.img{
    height: 300px;
    width: 100%;

}

.product-card .btn{
    background-color: var(--main-color) ;
}



//index.ejs
...code...
<div class="col-lg-4 product-card"> //added class and added text-center
            <div class="card text-center shadow mt-3 mx-auto" style="width: 18rem;">
```

task 17: now ek baar click krne par you need to show the details of that product

```
//routes --> product --> productRoutes

// route for shwoing the deatails of thre products
router.get('/products/:id' , async(req,res)=>{
    let {id} = req.params;
    let foundProduct = await Product.findById(id);
    res.render('products/show' , {foundProduct});

})
```

task 18: show.ejs banao and vahan par dikhao aapka foundProduct

```
/views --> product --> show.ejs
<% layout('layouts/boilerplate') %>
```

```
<div class="row">
    <div class="col-lg-6 product-card">
            <div class="card text-center shadow mt-3 mx-auto" style="width: 18rem;">
                <img class="img" src="<%=foundProduct.img%>" class="card-img-top" alt="...">
                <div class="card-body">
                  <h3 class="card-title"> <%= foundProduct.name %> </h3>
                  <h5 class="card-title"> Rs: <%= foundProduct.price %> </h5>
                  <p class="card-text"> <%= foundProduct.desc %> </p>
                  <a href="#" class="btn btn-primary">Buy Product</a>
                </div>
            </div>
    </div>

    <div class="col-lg-6">
        <h2 class="display-5">Leave your review</h2>
    </div>

</div>
```

💡 **index.ejs** par jaakr show product ka href change krdo i.e
<a href="/products/<%=item._id%>" class="btn btn-primary">View Product

task 19: ab edit route banao taaki pehle form mile and asli mei reflect krne ke liye we need method-override.

edit.ejs file bhi banao (same as new.ejs bas **value attribute** add krdena)

```
//views --> product --> edit.ejs
<% layout('layouts/boilerplate') %>


<div class="row">
    <div class="col-6 mx-auto">
        <form action="/products" method="POST">
            <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" value="<%=foundProduct.name%>">
            </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" value="<%=foundProduct.img%
            </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" placeholder="Price of Product" value="<%=foundProduct
                </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"><%=foundProduct.desc%></
            </div>
            <button type="submit" class="btn btn-sm btn-success">Save Changes</button>
        </form>
    </div>
</div>
```

```
//routes --> product --> productRoutes

// route for editing the product so we need form for it
router.get('/products/:id/edit' , async(req,res)=>{
    let {id} = req.params;
    let foundProduct = await Product.findById(id);
    res.render('edit' , {foundProduct});
})
```

💡 **edit btn** mei href dedo in **show.ejs** page so that click krne se hit krjaae route.

```
//views --> product --> show.ejs
...code...
<div class="row">
    <div class="col-lg-6 product-card mt-5">
            <div class="card shadow mt-3 mx-auto" style="width: 22rem;">
                <img class="img" src="<%=foundProduct.img%>" class="card-img-top" alt="...">
                <div class="card-body">
                  <h3 class="card-title text-center"> <%= foundProduct.name %> </h3>
                  <h5 class="card-title"> Rs: <%= foundProduct.price %> </h5>
                  <p class="card-text"> <%= foundProduct.desc %> </p>
                  <a href="#" class="btn btn-success">Buy</a>
                  <a href="#" class="btn btn-secondary">Add to Cart</a>
                  <a href="/products/<%=foundProduct._id%>/edit" class="btn btn-info">Edit</a>
                  <form class="d-inline-block" action="/products/<%=foundProduct._id%>?_method=DELETE" method="POST">
                      <button class="btn btn-danger btn-sm">Delete</button>
                   </form>
                </div>
            </div>
    </div>
```

task 20: Ab hume database mei karvaane hai changes jo edit form mei hue hai to vahan changes ke liye make route and use patch with method-override.

```
/routes --> products --> productRoutes
// changing the original edits in the database made in the editform
router.patch('/products/:id' , async(req,res)=>{
    let {id} = req.params;
    let {name,img,price,desc} = req.body;
    await Product.findByIdAndUpdate(id , {name,img,price,desc});
    res.redirect(`/products/${id}`)
})
```

ab hume request bhejna hai to edit.ejs mei bhi edit krna hoga about method and methodoverride.

```
//terminal
npm i method-override
//app.js
let methodOverride = require('method-override'); //added
...
app.use(express.urlencoded({extended:true}));
app.use(methodOverride('_method')); //added
```

hume edit.ejs mei form ka action bhi badalna hai.

```
//views --> products --> edit.ejs
 //just edit it
<form action="/products/<%=foundProduct._id%>?_method=PATCH" method="POST">
```

task 21: ab hume delete krna hai ek particular product

```
//rotes --> product --> productRoutes
//delete a route
router.delete('/products/:id' , async(req,res)=>{
    let {id} = req.params;
    await Product.findByIdAndDelete(id);
```

```
    res.redirect('/products');
})
```

show.ejs mei form banao so that form ke through delete kr paae **method-override** kre.

```
//views --> product --> show.ejs
<a href="/products/<%=foundProduct._id%>/edit" class="btn btn-info">Edit</a>
                  <form class="d-inline-block" action="/products/<%=foundProduct._id%>?_method=DELETE" method="POST">
                      <button class="btn btn-danger btn-sm">Delete</button>
                   </form>
```

Version 2

task 22: baar baar seed hone se bachne ke liye we will deleteMany() pehle.

```
//seed.js

async function seedDB(){
    await Product.deleteMany({}); //added here
    await Product.insertMany(products);
    console.log("data seeded successfully")
}
```

task 23: ab price is in number and humne text bheja hua hai so have to change the type taaki hum usse string mei bhi la paae and decimal values store kr paae**.**

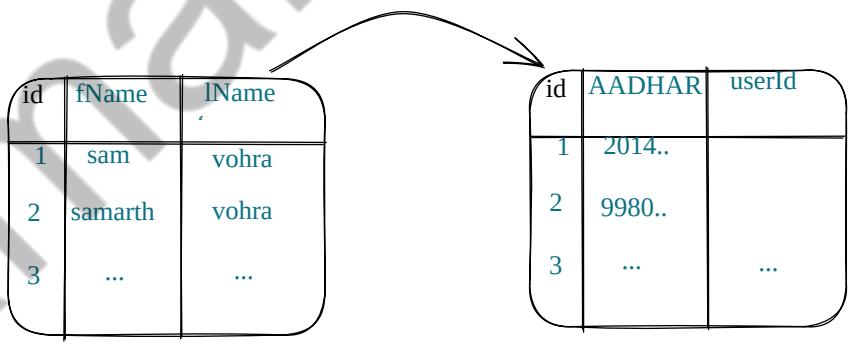
```
//views --> products --> new.ejs
<div class="input-group mb-3">
                    <span class="input-group-text" id="basic-addon1">Rs. </span>
                    <input class="form-control" step="any" type="number" name="price" id="paisa" placeholder="Price of Product">
                </div>
```

aapne aap typecasting hojaaegi need nhi aaegi aapko.

task 24: pehle hum thoda mogoose relationship samajh lete hai and then we will proceed.

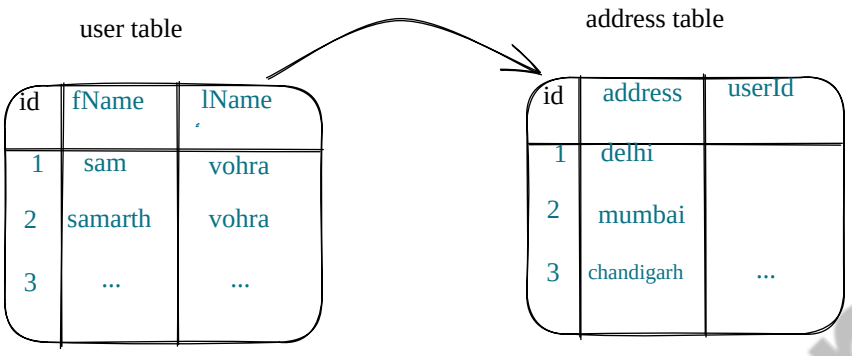
we have different type of relationship:

1. one to one (1:1)



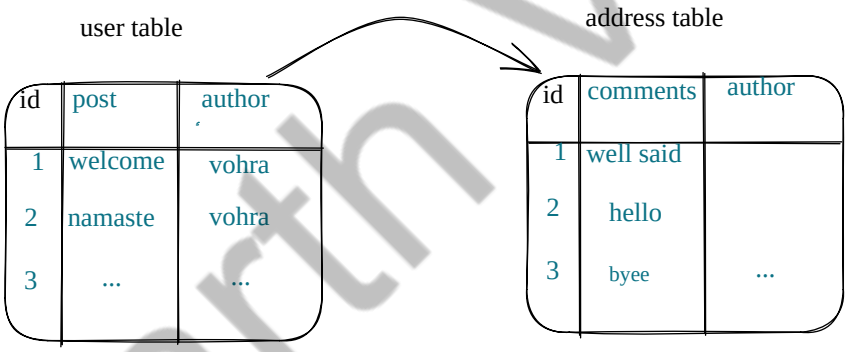
here we cannot have multiple addhar for one person
so it simply signifies one-one relationship

2. one to few (here few means less i.e 3-4 se zyada address nhi hote, so one to few)



here one user can have many address so it is one to few realtionship in this case.

3. one to many (1:N)



here one post can have many comments on it so it is one to many realtionship in this case.

4. m many to one
5. many to many

(majorly you will get one to few or one to many relationship)

task 25: (ignore krdo baccho)

adding user's address as array by creating schema.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://127.0.0.1:27017/relationDB')
.then(()=>{console.log("Db connected")})
.catch((err)=>{console.log(err)})
```

```
const userSchema = new mongoose.Schema({
    name:String ,
    age:Number ,
    addresses:[
        {
            _id:(id:false),
            lane: String ,
            city: String ,
            state: String ,
            country : String
        }
    ]
});

const User = mongoose.model('User' , userSchema );

const makeUser = async()=>{
   const user = new User({
        name:"samarth Vohra" ,
        age:27
    })
    await user.save();
    console.log("user successfully saved")
    console.log(user);
}

// makeUser();

const addAdress = async(id)=>{
    const user = await User.findById(id);
    user.addresses.push({
        lane: "G-157" ,
        city: "new delhi" ,
        state: "delhi" ,
        country : 'india'
    })

    await user.save();
    console.log(user)

}

addAdress('640e330a2e7f40db7d70c77e');
```

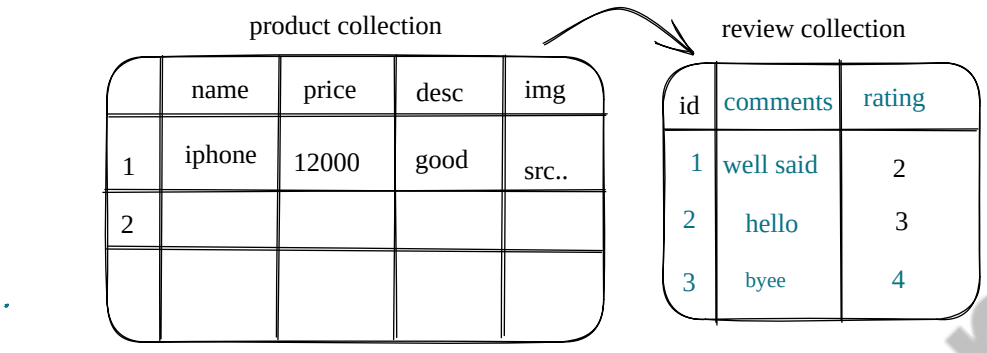
task 26:

Ab hume review add krna hai and review is also our 1:many relationship.

1. **make review.js**. file in model ka folder.

ab agar mai product ke model ke andar junga and vahan add krunga to that will be a case of one to few relationship lekin ab i want ki reviews to 1:many relationship hai to we will make reviews ka model alag se.

ab 2 table banenege



Ab hume in 2 collections ko apas mei kuch relation se jodhna hai taaki pta chal jaae kis product par kaunsa review aaya hai.

```
//review.js

const mongoose =  require('mongoose');

const reviewSchema = new mongoose.Schema({
    rating:{
        type:Number,
        min:0,
        max:5

    },
    comment:{
        type:String,
        trim:true
    }
})
let Review = mongoose.model('Review' , reviewSchema);
module.exports = Review;
```

Task 27: ab mai kya krunga mai product ke schema ke andar ek **reviews ki array** banaunga jiska kaam hoga ki us particular product ke upar jo jo review kia gya hai us review ko mai review ki id ki madad se inside the array store krdunga taaki jo jo review us product par hai vo hume using id mil jaae.

```
//product.js

desc:{
        type:String,
        trim:true
     },
reviews:[   //adding this
    {
        type:mongoose.Schema.Types.ObjectId,
        ref:'Review'
    }
]
```

task 28:

ab hume routes banane hai for the review to hume **review.js** krke file banani hai inside routes folder.

```
//routes --> review.js

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

router.post('/products/:id/review' , (req,res)=>{
    console.log(req.body);
    res.send("review route")
})


module.exports = router;
```

task 29: hume show.ejs mei bhi to input dene hai to get the values of rating.

```
//view --> product --> show.ejs

//adding here
<div class="col-lg-4 mt-5">
        <h2 class="display-5">Leave your review</h2>
        <form action="/products/<%= foundProduct._id %>/review" method="POST">
            <div class="mb-3">
                <label class="form-label" for="stars">Rating: </label>
                <input class="form-control" type="range" min="0" max="5" id="stars" name="rating">
            </div>
            <div class="mb-3">
                <label class="form-label" for="comment">Comment: </label>
                <textarea class="form-control" name="comment" id="comment" rows="3"></textarea>
            </div>
            <button class="btn btn-sm btn-success">Submit</button>
        </form>
    </div>
```

task 30: ab hume review route ko bhi app.js ke andar as a review hi to dekhna hai.

```
//app.js

const methodOverride  = require('method-override');
const reviewRoutes = require("./routes/review"); //added


// Routes
app.use(productRoutes);
app.use(reviewRoutes); //added
```

ab simply **review model** ko bhi require kro and start working on it.

ab jo id nikaal rha hu i.e product ki id and review ki id us product ki id mei store honi chahiye.

```
//review.js

const express =  require('express');
const Product = require('../models/product');
const Review = require('../models/review');

const router = express.Router();

router.post('/products/:id/review' , async(req,res)=>{
    let {productId} = req.params;
    let {rating , comment} = req.body;
```

```
    const product = await Product.findById(productId);
    // creating a new review
    let 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();
})


module.exports = router;
```

—> **Now see ki review push hua ya nhi hua in the array**

```
//terminal
yahan par dono colletions check kro you will be able to see
```

task 31:

ab hume simply show krna hai humare rendered reviews iske liye hum populate ka use krenge.

hum simply jahan par ek particular product display hora tha **(show.ejs)** ussi jagah par we need to populate this review.

> 💡 **populate** kya krega simply aapne jo object id store kari hai usko populate krega and us id ki jagah saara ka ssara comment and rating show krdega in place of that objectId.

```
//productRoutes.js
//go to show waala route


// route for shwoing the deatails of thre products
router.get('/products/:id' , async(req,res)=>{
    let {id} = req.params;
    // let foundProduct = await Product.findById(id);
    let foundProduct = await Product.findById(id).populate('reviews'); //changed
    console.log(foundProduct); //added
    res.render('products/show' , {foundProduct});

})
```

above way se we can join these 2 collections.

task 32:

Ab mere paas **foundProduct** to hai hi, to us **foundProduct** ke andar se reviews ke array use krke simply **show.ejs** ke andar jaakr mai simply show kar sakta hu.

```
//show.ejs

<button class="btn btn-sm btn-success">Submit</button>
        </form>

// added form here for showing reviews
        <div class="my-3">
```

```
           <% for(let review of foundProduct.reviews){ %>

            <div class="card mb-3">
                <div class="card-header">Rating: <%=review.rating%></div>
                <div class="card-body">
                  <!-- <h5 class="card-title">Special title treatment</h5> -->
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <button class="btn btn-danger">Delete</button>
                </div>
              </div>

           <% } %>
        </div>
```

ab hume redirect bhi krna hai to see the show.ejs page.

```
//review.js (routes)

router.post('/products/:productId/review' , async(req,res)=>{

        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});

        // adding review id to product array
        product.reviews.push(review);
        await review.save();
        await product.save();
        res.redirect(`/products/${productId}`) //added here

})
```

task 33:

Ab hum idhar star rating add krenge using a **libary starability.css**

```
//show.ejs

<div class="col-lg-4 mt-5">
        <h2 class="display-5">Leave your review</h2>
        <form action="/products/<%= foundProduct._id %>/review" method="POST">
            <div class="mb-3">
                <label class="form-label" for="stars">Rating: </label>
                <fieldset class="starability-basic">
                    <!-- <legend>First rating:</legend> -->below adding from library
                    <input type="radio" id="no-rate" class="input-no-rate" name="rating" value="0" checked aria-label="No rating." />
                    <input type="radio" id="first-rate1" name="rating" value="1" />
                    <label for="first-rate1" title="Terrible">1 star</label>
                    <input type="radio" id="first-rate2" name="rating" value="2" />
                    <label for="first-rate2" title="Not good">2 stars</label>
                    <input type="radio" id="first-rate3" name="rating" value="3" />
                    <label for="first-rate3" title="Average">3 stars</label>
                    <input type="radio" id="first-rate4" name="rating" value="4" />
                    <label for="first-rate4" title="Very good">4 stars</label>
                    <input type="radio" id="first-rate5" name="rating" value="5" />
                    <label for="first-rate5" title="Amazing">5 stars</label>
                  </fieldset>
                <!-- <input class="form-control" type="range" min="0" max="5" id="stars" name="rating"> -->
            </div>
            <div class="mb-3">
                <label class="form-label" for="comment">Comment: </label>
                <textarea class="form-control" name="comment" id="comment" rows="3"></textarea>
            </div>
            <button class="btn btn-sm btn-success">Submit</button>
        </form>

        <div class="my-3">
           <% for(let review of foundProduct.reviews){ %>

            <div class="card mb-3">
```

```
                <div class="card-header">Rating: <%=review.rating%></div>
                <div class="card-body">
                  <!-- <h5 class="card-title">Special title treatment</h5> -->
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <button class="btn btn-danger">Delete</button>
                </div>
              </div>

            <% } %>
          </div>


    </div>
```

task 34: ab hume css bhi add krni hai to we can do one thing css folder ke andar **star.css** file banalo

```
//css --> star.css

.starability-result {
  position: relative;
  width: 150px;
  height: 30px;
  background-image: url("
  font-size: 0.1em;
  color: transparent;
}

.starability-result:after {
  content: ' ';
  position: absolute;
  left: 0;
  height: 30px;
  background-image: url("
  background-position: 0 -30px;
}

.starability-result[data-rating="5"]::after {
  width: 150px;
}

.starability-result[data-rating="4"]::after {
  width: 120px;
}

.starability-result[data-rating="3"]::after {
  width: 90px;
}

.starability-result[data-rating="2"]::after {
  width: 60px;
}

.starability-result[data-rating="1"]::after {
  width: 30px;
}

@media screen and (-webkit-min-device-pixel-ratio: 2), screen and (min-resolution: 192dpi) {
  .starability-result {
    background-image: url("
    background-size: 30px auto;
  }
  .starability-result:after {
    background-image: url("
    background-size: 30px auto;
  }
}

.starability-basic {
  display: block;
  position: relative;
  width: 150px;
  min-height: 60px;
  padding: 0;
  border: none;
}
```

```
.starability-basic > input {
  position: absolute;
  margin-right: -100%;
  opacity: 0;
}

.starability-basic > input:checked ~ label,
.starability-basic > input:focus ~ label {
  background-position: 0 0;
}

.starability-basic > input:checked + label,
.starability-basic > input:focus + label {
  background-position: 0 -30px;
}

.starability-basic > input[disabled]:hover + label {
  cursor: default;
}

.starability-basic > input:not([disabled]):hover ~ label {
  background-position: 0 0;
}

.starability-basic > input:not([disabled]):hover + label {
  background-position: 0 -30px;
}

.starability-basic > input:not([disabled]):hover + label::before {
  opacity: 1;
}

.starability-basic > input:focus + label {
  outline: 1px dotted #999;
}

.starability-basic .starability-focus-ring {
  position: absolute;
  left: 0;
  width: 100%;
  height: 30px;
  outline: 2px dotted #999;
  pointer-events: none;
  opacity: 0;
}

.starability-basic > .input-no-rate:focus ~ .starability-focus-ring {
  opacity: 1;
}

.starability-basic > label {
  position: relative;
  display: inline-block;
  float: left;
  width: 30px;
  height: 30px;
  font-size: 0.1em;
  color: transparent;
  cursor: pointer;
  background-image: url("
  background-repeat: no-repeat;
  background-position: 0 -30px;
}

.starability-basic > label::before {
  content: '';
  position: absolute;
  display: block;
  height: 30px;
  background-image: url("
  background-position: 0 30px;
  pointer-events: none;
  opacity: 0;
}

.starability-basic > label:nth-of-type(5)::before {
  width: 120px;
  left: -120px;
}

.starability-basic > label:nth-of-type(4)::before {
```

```
  width: 90px;
  left: -90px;
}

.starability-basic > label:nth-of-type(3)::before {
  width: 60px;
  left: -60px;
}

.starability-basic > label:nth-of-type(2)::before {
  width: 30px;
  left: -30px;
}

.starability-basic > label:nth-of-type(1)::before {
  width: 0px;
  left: 0px;
}

@media screen and (-webkit-min-device-pixel-ratio: 2), screen and (min-resolution: 192dpi) {
  .starability-basic > label {
    background-image: url("
    background-size: 30px auto;
  }
}

@media screen and (-ms-high-contrast: active) {
  .starability-basic {
    width: auto;
  }
  .starability-basic > input {
    position: static;
    margin-right: 0;
    opacity: 1;
  }
  .starability-basic .input-no-rate {
    display: none;
  }
  .starability-basic > label {
    display: inline;
    float: none;
    width: auto;
    height: auto;
    font-size: 1em;
    color: inherit;
    background: none;
  }
  .starability-basic > label::before, .starability-basic > label::after {
    display: none;
  }
}
```

ab boiler plate par bhi to add kro is css ka link.

```
//layouts --> boilerplate.ejs

<link rel="stylesheet" href="/css/app.css">
<link rel="stylesheet" href="/css/star.css"> //added
```

task 35:

ab task mei hume star hi to show krne hai rather than showing rating as number so simply use the same library anc scroll down. (showing the static result)

```
//show.ejs
<div class="my-3">
          <% for(let review of foundProduct.reviews){ %>

           <div class="card mb-3"> //changed below
               <!-- <div class="card-header">Rating: <%=review.rating%></div> -->
               <div class="card-body">
```

```
                    <p class="starability-result" data-rating="<%=review.rating%>">
                        Rated: <%=review.rating%> stars //changed here
                     </p>
                  <!-- <h5 class="card-title">Special title treatment</h5> -->
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <button class="btn btn-danger">Delete</button>
                </div>
              </div>

            <% } %>
 </div>
```

task36: ab hume time of comment bhi add krna hai to hum simply **review ke model** ke andar ek object bhejdenge **timestamps** ka jiski madad se we can conclude the values **updatedAt** adn **createdAt.**

```
//review.js (schema)

comment:{
        type:String,
        trim:true
    }
} , {timestamps:true}) //adding here
```

now show at the databse that are they showing the value

ab merko is timestamp ko months and years mei krna hai change so uske liye ill make use of our stackoverflow. **(timestamp to date js) —>** toDateString();

```
//show.ejs
<div class="card mb-3">
                <!-- <div class="card-header">Rating: <%=review.rating%></div> -->
                <div class="card-body">
                    <p class="starability-result" data-rating="<%=review.rating%>">
                        Rated: <%=review.rating%> stars
                     </p>
                  <!-- <h5 class="card-title">Special title treatment</h5> -->

                  //adding code below
                  <p class="card-text">Comment: <%=review.comment%></p>
                  <%if(review.createdAt){%>
                    <p> <%=review.createdAt.toDateString()%></p>
                  <%}%>
                  <button class="btn btn-danger">Delete</button>
                </div>
</div>
```

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;
```

Version 4

cookies and session

cookie is a general term jo **client side storage** ka part hai (client side storage = browser)

cookie are **key-value** pair jo server aapke browser mei bhejta hai

ye 3 cheezo ke kaam aata hai. **(search http cookie mdn)**

1. personalisation
2. session management
3. tracking

(go to application and show **cookies**)

humara cookie **stateful** hota hai i.e it is dependant on previous requests i.e aapne cookie store krli and ab ab request bhejre ho i.e req ke saath saath cookie jaara hai , to humare liye it is stateful and on the other hand baaki routes are **stateless** for us.

to aap directly nhi dekh sakte aapke request ke andar ki cookie uske liye you need to use **cookie-parser** middleware from npm.

task 43:

```
//terminal
npm i cookie-parser
```

```
//app.js

const express = require('express');
const app =  express();
const cookieParser = require('cookie-parser');


app.use(cookieParser());

app.get('/' , (req,res)=>{
    res.send('connected');
})

app.get('/setcookie' , (req,res)=>{
    res.cookie('mode' , 'light');
    res.cookie('location' , 'delhi');
    res.cookie('username' , 'samarth');
    res.send('ent you a cookie successfully');
})

app.get('/greet' , (req,res)=>{
    let {username} = req.cookies;
    // console.log(req.cookies);
    res.send(`hi bro ${username} hope you r doing good`);
})
```

```
app.listen(3000 ,(req,res)=>{
    console.log("server running at 3000");
})
```

task 44: signed cookie

ab ek boht zyada useful statement hai jisse we say signed cookie (v.v.imp)

if you want to work with signed cookie you need to work with the middleware simply go to **npm cookie-parser.**

```
//app.js

const cookieParser = require('cookie-parser');


app.use(cookieParser('youneedabettersecret')); //added


app.get('/' , (req,res)=>{
    console.log(req.cookies) //added
    res.send(req.cookies); //chnaged
})

app.get('/getsignedcookie' , (req,res)=>{
    res.cookie('earthquake' , 'aaya' , {signed:true}); //added
    res.send('cookie sent successfully')
})
```

ab jo aapko signed cookie mili hai vo aapko res.send ke andar nhi milega baaki sab mil jaega.

> 💡 express ka middleware hai to use docs mei **cookie** dekh lo.

ab point to remeber is aapne tamper krdia apne **cookie ko jo signed** thi to jab aap usse dekhna chahte ho, aap usse directly access nhi kr paoge aapko **express.js ke andar** jaana hai and vahan **request ke andar** you will see **req.signedCookies.** Ab jab aap usse res.send krvaoge to you will not get the RESPONSE as it is tampered *you will get* **false.**

```
//app.js

app.get('/' , (req,res)=>{
    // console.log(req.cookies);
    // res.send(req.cookies);
```

```
    res.send(req.signedCookies); //changed
})
```

cookies ki kuch limitations hai, cookies aapke browser ke andar store hoti hai to it is **client side storage,** ab cookie ke andar jo cheeze hai that is because of your browser agar aap broser change kroge to aapki cookie lost hojaegi.

but cookie ke andar aap kabhi koi crusial information store nhi krte so agar info chali bhi jaae to aapko dikkat nhi hoti. and cookie ki bhi khudki ek limit hoti hai.

so ab i want ki meri information stored rahe and hatt na jaae. so we use **session.**

session storage (browser ke andar storage), lekin **session** is server side storage.

for eg: choti moti information hai usse hum session mei store kar sakte hai like **user** ki info.

Session:

task 45:

1. stores key-value pair

aapne server ko request bheji and now server aapko ek response dega ab server aapko response ke saath saath cookie bhi bhejta hai us cookie ke andar aapki session ki id hogi and session ke andar kuch jagah allocate hojaati hai to store your session ke andar stored cheeze like username.

and agar ab aap request bhi bhejoge to request apne andar cookie bhjeti hai harr request ke saath to information stored rahegi.

```
//terminal
mkdir express-session-demo
npm init -y
npm i express nodemon
npm i express-session
touch app.js
```

ab humne bataya tha ki cookie ke andar session id saath jaati hai to ye store hogi us secret ke behalf se.

and we have a different http and https methods, so we will removethe 4th argument we will handle **s**(security at the time of deployment) that happens in the case of deployment.

```
//app.js
//go to session npm package and copy
```

```
const express = require('express');
const app =  express();
const session = require('express-session');

app.use(session({
    secret: 'keyboard cat',
    resave: false,
    saveUninitialized: true,
    // cookie: { secure: true } //https case
}))


app.get('/' , (req,res)=>{
    res.send('welcome to session');
})



app.listen(3000 ,(req,res)=>{
    console.log("server running at 3000");
})
```

ab hume mere cookie ke andar session ki id nazar arhi hai jiski hum baat kr rhe they.

> 💡 agar server restart hota hai to saari cheeze firse store krni hogi automatically ni hoga.

task 46:

```
//app.js

const express = require('express');
const app =  express();
const session = require('express-session');

app.use(session({
    secret: 'keyboard cat',
    resave: false,
    saveUninitialized: true,
    // cookie: { secure: true } //https
}))


app.get('/' , (req,res)=>{
    res.send('welcome to session');
})

//added
app.get('/viewcount' , (req,res)=>{
    if(req.session.count){
        req.session.count+=1;
    }
    else{
        req.session.count=1;
    }
    res.send(`You visited counter ${req.session.count} times`);
})
```

```
//added
app.get('/setname' , (req,res)=>{
    req.session.username = "samarth vohra";
    res.redirect('/greet');
})
//added
app.get('/greet' , (req,res)=>{
    let {username="anonymous"} = req.session;
    res.send(`hi from ${username}`)

})

app.listen(3000 ,(req,res)=>{
    console.log("server running at 3000");
})
```

ab agar aap session storage ko change krke hit kroge to kya hoga anonymous ajaaega instead of samarth vohra.

UI Change kia hai thoda sa so simply jaakr dekho ki how can you make it & humne average reviews waali cheez bhi add krdi hai. (github)

task 48: ab mujhe kuch message provide krna hai to ill use flash msg waali cheezo ko display krna hai to how can we do it.

using a **flash** npm package.

```
//terminal
npm i connect-flash express-session
```

require kro express session ko and simply use it.

```
//app.js
const reviewRoutes = require("./routes/review");
const session = require('express-session'); //added from here
const flash = require('connect-flash');

...code...
let configSession = {
    secret: 'keyboard cat',
    resave: false,
    saveUninitialized: true
}

app.use(session(configSession));
app.use(flash());  //added till here

// Routes
app.use(productRoutes);
app.use(reviewRoutes);
```

```
//review.js
await product.save();
req.flash('msg' , 'Review added successfully');
res.redirect(`/products/${productId}`)
```

ab review redirect kahan kar rha hai vo simply show a particular product par kr rha hai so we will go to **productRoutes.js**

```
//productRoutes.js
// 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 , msg:req.flash('msg')}); //added
    }

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

})
```

task 49: ab mai show template mei jaaunga and vahan simply card ke upar msg flash krvaado.

```
//show.ejs
<div class="col-lg-6 product-card mt-5">
//added below
        <div class="mb-3">
            <%if(msg && msg.length){%>
                <%=msg%>
            <%}%>
        </div>
```

task 50: ab edit ke liye bhi flash jalao and is case mei show par hi redirect kar rhe hai to humara setup of msg already hai vahan par to dikkat nhi aaegi.

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

        let {id} = req.params;
```

```
        let {name,img,price,desc} = req.body;
        await Product.findByIdAndUpdate(id , {name,img,price,desc});
        req.flash('msg' , 'Product edited successfully'); //added
        res.redirect(`/products/${id}`)
    }

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

ab bootstrap se sundar kar sakte ho aap usse simply alert —> dismiss

```
<%if(msg && msg.length){%>
                <div class="alert alert-warning alert-dismissible fade show" role="alert">
                    <strong><%=msg%></strong>
                    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
                </div>

<%}%>
//added above
<div class="row">
    <div class="col-lg-6 product-card mt-5">
```

task 51: ab merko harr file mei jakr uske andar aaise req.flash bhejna pdhega to usse bachne ke liye we have something called **locals**.

ab mujhe harr template par mere ek flash dena hai to i can make use of **locals**.

ek middleware banao and jo bhi aapset kroge vo harr jagah milegi.

```
//app.js
app.use(session(configSession));
app.use(flash());

app.use((req,res,next)=>{
    res.locals.success = req.flash('success');
    res.locals.error = req.flash('error');
    next();
})
```

ek flash.ejs file banao jiske andar success and error ke alert stored ho taaki hum baar baar alag alag jagah jaakr store na kr rhe ho.

```
//flash.ejs

<%if(success && success.length){%>
    <div class="alert alert-warning alert-dismissible fade show" role="alert">
        <strong><%=success%></strong>
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
      </div>
<%}%>
```

```
<%if(error && error.length){%>
    <div class="alert alert-warning alert-dismissible fade show" role="alert">
        <strong><%=error%></strong>
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
      </div>
<%}%>
```

ab merko show.ejs mei jaakr change krna hai.

```
//show.ejs

<%-include('../partials/flash')%>
<div class="row">

//index.ejs

<% layout('layouts/boilerplate') %>
<%-include('../partials/flash')%>
```

and review.js ke andar se bhi merko success ke naam se bhejna hai flash.

```
//review.js
await product.save();
req.flash('success' , 'Review added successfully'); //added
res.redirect(`/products/${productId}`)
```

💡 aapne review create kia and now review ke andar ka flash success ke corressponding aapne msg store krdia and jab aap redirect hore ho to aap redirect kr rhe ho so here aap harr incoming se pehle middleware run kr rhe ho jiske andar locals ka storage hai.
ab success ke andar success waala msg store hogya hai

```
//productsRoutes.js
...code...

// actually adding a product in a DB
router.post('/products' , validateProduct , async (req,res)=>{
    // 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});
        req.flash('success' , 'Product added successfully'); //added
        res.redirect('/products');
    // }

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


...code...


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

        let {id} = req.params;
        let {name,img,price,desc} = req.body;
        await Product.findByIdAndUpdate(id , {name,img,price,desc});
        req.flash('success' , 'Product edited successfully'); //added
        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);
        req.flash('success' , 'Product deleted successfully'); //added
        res.redirect('/products');
    }

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

Version 5

authDemo

Version 5 mei original changes

So now hum user ko picture mei lekar aaenge using a tool **Passport. (for authentication)**

hum bcrypt naam ka hashing fucntion use krenge.

go to **passport.js** (only for nodejs)

show **documentation** —> **authenticate**

1. hum **local strategy** use krne vaale hai (strategy)
2. **passport** is a tool (using tool)

strategies mei jaao index ke andar and make them see the content.\

task 58:

Ab create a schema and yaad rakho schema ke andar password directly store nhi ho sakte we need to hash the password pehle using a package namely **passport-local-mongoose** hashing function.

> 💡 ab **passport-local-mongoose** ka kaam hai jab aap signup krte ho and aapki detials like username and password store krte ho DB mei to ye **PLM** usse encrypt krke store krvaega bcrypt bhi krdeta hai same lekin **PLM** easily krdega for us i.e seedha hash krdega but one thing to keep in mind is ki ye username aur password kaun add krega in my schema to ye **PLM** krdega so hume iske liye schema banane ki need nhi hai.

task 59: hume **PLM** ko install krke plugin bhi add krne hote hai (documentation in npm)

taaki **PLM** kisaari power ko use kr sake.

```
//terminal
npm i passport-local-mongoose



//models --> user.js
const mongoose = require('mongoose');
const passportLocalMongoose = require('passport-local-mongoose'); //add later
```

```
const userSchema = new mongoose.Schema({
    email: {
        type: String,
        trim: true,
        required: true
    }
});

userSchema.plugin(passportLocalMongoose); //add later


const User = mongoose.model('User', userSchema);

module.exports = User;
```

task 60: ab new route banana hai for user i.e **auth.js** inside routes folder.

and use **app.js mei require krlo**.

```
//routes--> auth.js
const express = require('express');
const router = express.Router();


module.exports = router;




//app.js file

const productRoutes = require('./routes/product');
const reviewRoutes = require('./routes/review');
const authRoutes = require('./routes/auth');       //added


app.use(productRoutes);
app.use(reviewRoutes);
app.use(authRoutes);       //added
```

task 61:

ab documentation mei **static methods** dikhao and **register** dikhao.

static methods directly aapke schema par apply hojaae hai chahe unhe aap banao ya koi aur.

```
//auth.js
//isme hum password alag se dedenge not variable ke saath mei

const express = require('express');
const router = express.Router();
const User = require('../models/user');


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

```
    const user = {
        email: 'samarth@gmail.com',
        username:'samarth'
    }
    const newUser = await User.register(user, 'sam123');

    res.send(newUser);
});



module.exports = router;
```

—> Ab hum db mei dekh sakte hai humare db mei ki user aaya hai ya nhi jo aapka PLM automatically kr rha hai sab and 2 fields bhi khud se dera hai.

IMPORTANT: COMMENT KRDIA HAI BACCHO FAKEUSER KO TO NOTES AACHE SE DEKHO

task 62:

ab register() to **PLM** ne krdia ab jab hum login form bharenge to humara **passport** kya krega seedha user jo register ho rakhe hai unke andar se authenticate() method ki madad se behind the scene match krva kr **login** krva dega.

passport.js ke docs mei jaakr **config** dikhao and batao ki it accepts a local strategy ka function and simply authenticate the user.

—> hum khud ka bhi function add krsakte hai just like in passport ka scene but PLM hume derha hai to we wil simply use the method **authentcate().**

```
//terminal
npm i passport  passport-local



//app.js
const passport =  require('passport'); //added all
const LocalStrategy =  require('passport-local');
const User = require('./models/user');

...code...

app.use(session(sessionConfig));
app.use(flash());


passport.use(new LocalStrategy( //added from here

));
```

ab hume middleware set krna hai after **authentcate (jo passport ko bola gya hai to use)**

ab we want persistant login to uske liye we have **passport.initialize()** middleware. and to store it we have **passport.session().**

```
//app.js
app.use(flash());

app.use(passport.initialize()); //added
app.use(passport.session());  //added
```

task 62: ab hume session ka kaam krna hai to for that we need to make use of **serialize** the user and **deserialize** the user.

```
//app.js

app.use(session(sessionConfig));
app.use(flash());

app.use(passport.initialize());
app.use(passport.session());
passport.serializeUser(User.serializeUser()); //added
passport.deserializeUser(User.deserializeUser()); //added



passport.use(new LocalStrategy(User.authenticate()));
```

ab humara kaam sirf pages banane ka hai and unke routes banane hai.

task 63:

ek route hit kro to get the **signup** form.

ek folder banao **auth** and simply uske andar file banao signup.js

```
//auth.js

router.get('/register' , (req,res)=>{
    res.render('auth/signup');
})



//views --> auth --> signup.js
<% layout('layouts/boilerplate') -%>


<div class="row">
    <div class="col-md-6 mx-auto">
        <%- include('../partials/flash')  %>
```

```
        <h1 class="display-6">Sign Up</h1>
        <form action="/register" method="post">
            <div class="mb-3">
                <label class="form-label" for="username">Username</label>
                <input class="form-control" type="text" name="username" id="username" placeholder="Username">
            </div>
            <div class="mb-3">
                <label class="form-label" for="email">Email</label>
                <input class="form-control" type="email" name="email" id="email" placeholder="Email">
            </div>
            <div class="mb-3">
                <label class="form-label" for="password">Password</label>
                <input class="form-control" type="password" name="password" id="password" placeholder="Password">
            </div>
            <button type="submit" class="btn btn-success btn-small">SignUp</button>
        </form>
    </div>
</div>
```

task 64: ab hume signup krna hai actually i.e data ko macth krvana hai to post request bhejunga.

taaki us data ke through mai user bana aku to simply i would need to make use of a class to make that user

—> form ka methods **POST** and route **/register**.

```
//auth.js

//form adding
router.get('/register' , (req,res)=>{
    res.render('auth/signup');
})


//actually adding
router.post('/register' , async(req,res)=>{
    let {email,username,password} = req.body;
    const user = new User({email,username});
    const newUser = await User.register(user,password);
    res.send(newUser);
})
```

Ab hum check kr sakte hai ki hum user create kar paa rhe hai.

task 65:

ab hume login form banana hai and simply usse match krvana hai data.

make a file login.ejs in auth folder

```
//views --> auth --> login.ejs

//same as signup bas email hta do
```

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


<div class="row">
    <div class="col-md-5 mx-auto">
        <%- include('../partials/flash')  %>
        <div class="card p-3 shadow-sm m-2">
            <h1 class="display-6">Login</h1>
            <form action="/login" method="post">
                <div class="mb-3">
                    <label class="form-label" for="username">Username</label>
                    <input class="form-control" type="text" name="username" id="username" placeholder="Username">
                </div>
                <div class="mb-3">
                    <label class="form-label" for="password">Password</label>
                    <input class="form-control" type="password" name="password" id="password" placeholder="Password">
                </div>
                <button type="submit" class="btn btn-success btn-block btn-small">Login</button>
                <p class="fw-light mt-2">Don't have an account yet ? <a href="/register">SignUp</a> </p>
            </form>
        </div>
    </div>
</div>
```

```
//auth.js

//route to get login form
router.get('/login' , (req,res)=>{
    res.render('auth/login');
})
```

task 66: ab sabse important kaam ki user ko login krvana hai.

ab jab post request jaegi mera passport app.js file ke andar user.authenticate() naam ka middleware chalaega and aap usme simply authenticate krva dega aapke user ko.

```
//auth.js

router.get('/login' , (req,res)=>{
    res.render('auth/login');
})

//to check in the authentication waala system when ur authenticate
//will go to app.js and hit it.
//passport.js se copy
router.post('/login',
    passport.authenticate('local', {
       failureRedirect: '/login',
       failureMessage: true ,
    }),
    function(req, res) {
        req.flash('success' , 'welcome back')
        res.redirect('/products');
    }
);
```

```
//passport ko auth.js mei bhi require kro
const User = require('../models/user');
const passport = require('passport');      //added
```

Ab hum login kar paa rhe hai we can see it by simply redirecting it to /products and see all the products.

task 67: navbar ke andar login and logout mei path dedo.

```
//navbar.ejs
<div class="navbar-nav ml-auto">
          <a href="/login" class="nav-link">Login</a>
          <a href="/logout" class="nav-link"><i class="fas fa-shopping-cart"></i>
            <sup><span class="badge bg-danger">2</span></sup>
          </a>
          <a href="#" class="nav-link">Logout</a>
 </div>
```

task 68: ab we need to logout the user.

```
//auth.js

//logout session
router.get('/logout',(req,res)=>{
    ()=>{
        req.logout();
    }
    req.flash('success' , 'goodbye friend');
    res.redirect('/products');
});
```

jab bhi aap authenticate hoge to ek property apne aap automatically aapke req ke andar user add krdega i.e **req.user.**

to aap console mei dekh sakte ho and simply uske naam se print kr sakte ho.

```
//auth.js

router.post('/login',
    passport.authenticate('local', {
       failureRedirect: '/login',
       failureMessage: true ,
    }),
    function(req, res) {
        // console.log(req.user); //adding
        req.flash('success' , `welcome back ${req.user.username}`). //adding
        res.redirect('/products');
```

```
    }
);
```

task 69: ab we want ki jo ye user hai req.user() ke andar, to usme hum access kar paae to hum usse simply bhej dete hai in middleware in app.js

```
//app.js
app.use((req, res, next) => {
    res.locals.currentUser = req.user;      //added new
    res.locals.success = req.flash('success');
    res.locals.error = req.flash('error');
    next();
})
```

ab mai simply check lga sakta hu ki agar user login hai to usse signup na dikhau & agar nhi hai to login dikhau.

```
//navbar.ejs

<div class="navbar-nav ml-auto">

        <%if(!currentUser){%>      //added from here
          <a href="/login" class="nav-link">Login</a>
        <%}else{%>
          <a href="#" class="nav-link"><i class="fas fa-shopping-cart"></i>
            <sup><span class="badge bg-danger">2</span></sup>
          </a>
          <a href="/logout" class="nav-link">Logout</a>
        <%}%>


</div>
```

task 70: ab hum simply add krna chahate hai ek middleware kyu? taaki mai simply uske andar ek check lga saku ki kya band aloggedin hai agar loggedin nahi hai to hum simply login kre bina koi page open nhi krne denge and agar login hua to hum usme saare pages access krne denge,

```
//middleware.js

const isLoggedIn = (req,res,next)=>{
    if(!req.isAuthenticated()){
        req.flash('error' , 'you need to login first');
        return res.redirect('/login');
    }
    next();
}
...

module.exports = {validateProduct ,validateReview , isLoggedIn} ; //changed
```

ab hume isse routes mei use bhi krna hai to simply isse aap product.js vaale mei use kro.

```
//product.js

const { validateProduct , isLoggedIn } = require('../middleware'); //chnged

//ab sab routes mei add krdo middleware
```

task 71: ab i want ki login functionality mera signup form fill krne ke baad chl jaae.

ab req ke andar login ka access bhi rehta hai so for that we can use the following function which is in passport.

```
//auth.js
router.post('/register' , async(req,res)=>{
    try{
        let {email,username,password} = req.body;
        const user = new User({email,username});
        const newUser = await User.register(user,password);
        // res.send(newUser);
        req.login(newUser, function(err) {        //addig below
            if (err) { return next(err); }
            req.flash('success' , 'welcome, you are registered successfully')
            return res.redirect('/products');
        });
    }
    catch(e){
        req.flash('error' , e.message);
        return res.redirect('/products');
    }
})
```

task 72: ab session ko end krne ke liye cookie ko hum humare **expressSession** vaale middleware mei use kr sakte hai.

💡 basically session end krne ke liye you can see in application part to vahan nhi hota session mei end to aap khud se kr rhe ho end.

```
//app.js

const sessionConfig = {
    secret: 'weneedsomebettersecret',
    resave: false,
    saveUninitialized: true,
    cookie:{            //added from here
        httpOnly:true,
        expires:Date.now() + 1000*60*60*24*7,
        maxAge: 1000*60*60*24*7
    }
}
```

Version 6

task 73: ab i want ki agar mai review add krna chahta hu to usse add krte hue jab mai login page par firse hit kru to us tym par mujhe rather then showing things from the starting it should show me from the last step itself to uske liye i have an object namely **req.originalUrl**. jiski madad se mai simply use kr sakta hu last route.

```
//middleware.js

const isLoggedIn = (req,res,next)=>{
    console.log(req.originalUrl); //added
    if(!req.isAuthenticated()){
        req.flash('error' , 'you need to login first');
        return res.redirect('/login');
    }
    next();
}
```

ab mai vapas ussi page par jaana chahta hu after login so, hum login button hit waali functionality kaise kar paa rhe hai because of our route i.e jahan hum products ko redirect krva rhe hai vahan simply hum previous route i.e **originalUrl** dedenge. vo simply mai session ke andar store krke le sakta hu.

task 74: ab hum kuch autherisation krenge as we are done with authentication.

—> jaise ki jo user hai kya vo **seller** hai ya **buyer.**

to uske liye aap signup.ejs form mei add kroge and simply

```
//signup.ejs
<% layout('layouts/boilerplate') -%>


<div class="row">
    <div class="col-md-6 mx-auto">
        <%- include('../partials/flash')  %>
        <h1 class="display-6">Sign Up</h1>
        <form action="/register" method="post">
            <div class="mb-3">
                <label class="form-label" for="username">Username</label>
                <input class="form-control" type="text" name="username" id="username" placeholder="Username">
            </div>
            <div class="mb-3">
                <label class="form-label" for="email">Email</label>
                <input class="form-control" type="email" name="email" id="email" placeholder="Email">
            </div>
            <div class="mb-3">
                <p class="mb-2">Want to register as ? </p>
                <div class="mx-2">
                    <label class="form-label" for="seller">Seller</label>
                    <input class="form-check-input mt-0" type="radio" name="role" value="seller" id="seller" aria-label="Radio button for f
                </div>
                <div class="mx-2">
                    <label class="form-label" for="buyer">Buyer</label>
                    <input class="form-check-input mt-0" type="radio" name="role" value="buyer"  id="buyer" aria-label="Radio button for fo
                </div>

            </div>
            <div class="mb-3">
                <label class="form-label" for="password">Password</label>
                <input class="form-control" type="password" name="password" id="password" placeholder="Password">
            </div>
            <button type="submit" class="btn btn-success btn-small">SignUp</button>
        </form>
    </div>
</div>
```

task 75: ab aapke paas ek naya field add hua hai to aap schema mei bhi change kroge.

```
//models --> user.js

const userSchema = new mongoose.Schema({
    email: {
        type: String,
        trim: true,
        required: true
    },
    role:{
        type:String,
        default:'buyer',

    }

});
```

and ab jab aap ek new user create kr rhe ho i.e auth.js mei, to simply you will add the property.

```
//auth.js
router.post('/register' , async(req,res)=>{
    try{
        let {email,username,password,role} = req.body; //added here
        const user = new User({email,username,role}); //added here
        const newUser = await User.register(user,password);
        // res.send(newUser);
        req.login(newUser, function(err) {
            if (err) { return next(err); }
            req.flash('success' , 'welcome, you are registered successfully')
            return res.redirect('/products');
        });
    }
    catch(e){
        req.flash('error' , e.message);
        return res.redirect('/products');
    }
})
```

task 75: ab is role ke basis par mai kuch hide kr skta hu i.e ki user hai ya buyer us basis par cheeze hide krdunga. i.e agar user hai to product add nhi krne dunga lekin agar buyer hai to user ko review and buy krne dunga.

> 💡 ab mai poora user ka collection drop krke simply buyer ka ek banda and user ka ek banda bana dunga so that i can see both functionalities.

ab navbar mei name bhi nazar ajaae to mai usse navbar.ejs mei edit krdeta hu.

```
//navbar.ejs

<%if(!currentUser){%>
          <a href="/login" class="nav-link">Login</a>
        <%} else{%>
          <a href="#" class="nav-link text-capitalize">Hello <%=currentUser.username%></i>  //adding this
            <a href="#" class="nav-link"><i class="fas fa-shopping-cart"></i>
            <sup><span class="badge bg-danger">2</span></sup>
          </a>
          <a href="/logout" class="nav-link">Logout</a>
 <%}%>
```

task 76: ab mai yehi chahunga ki jis bande ne us particular product ko banaya tha vahi isse delete krpae.

iske liye aapko product ke schema mei kuch value add krni hai i.e **author** of that product.

```
//models --> product.js
avgRating: {
        type: Number,
        default:0
    },
    author:{  ////added form this to below
        type:mongoose.Schema.Types.ObjectId,
        ref:'User'
    },
```

ab my task is ki jab bhi mai product ko create kr rha hu mai vahan us product ke saath author bhi add kru.

```
//routes --> product.js
router.post('/products',isLoggedIn,validateProduct,async (req, res) => {

    try {
        const { name, img, desc, price } = req.body;  //added below 1 line
        await Product.create({ name, img, price: parseFloat(price), desc,author:req.user._id });
        req.flash('success', 'Successfully added a new product!');
        res.redirect('/products');
    }
    catch (e) {
        res.status(500).render('error', { err: e.message })
    }
});
```

💡 Ab author ki id store hogyi hogi is db mei check krlo.

task 77: ab simply add krchuke hai new product ko and vahi author ab delete kr sakta hai us product ko so simply ek middleware banalo for the same.

```
//middleware.js

//added poora
const isSeller = (req,res,next)=>{
    if(!req.user.role){
        req.flash('error' , 'you donot have the permission to do that');
        return res.redirect('/products');
    }
    else if(req.user.role !== 'seller'){
        req.flash('error' , 'you donot have the permission to do that');
        return res.redirect('/products');
    }
    next();
}
//changed below
module.exports = {validateProduct ,validateReview , isLoggedIn , isSeller} ;
```

Ab tum isse check krlo ki ye seller hai ya nhi to product.js mei jaakr grab that middleware and check it.

```
//product.js        //changed below
const { validateProduct , isLoggedIn  , isSeller} = require('../middleware');

//added middleware
router.post('/products',isLoggedIn,isSeller,validateProduct,async (req, res) => {

    try {
        const { name, img, desc, price } = req.body;
        await Product.create({ name, img, price: parseFloat(price), desc,author:req.user._id });
        req.flash('success', 'Successfully added a new product!');
        res.redirect('/products');
    }
    catch (e) {
```

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

task 78: ab agar user seller nhi hai to we can remove the new link ka access from the navbar.

```
//navbar.ejs

<%if(currentUser && currentUser.role === 'seller'){%>
        <li class="nav-item">
          <a class="nav-link" href="/products/new" tabindex="-1" aria-disabled="true">New</a>
        </li>
 <%}%>
```

same goes for buy and delete agar vo seller nhi hai to hata dete hai access.

```
//show.ejs
<%if(currentUser && currentUser.role ==='seller'){%>
                    <a href="/products/<%=product._id%>/edit" class="btn btn-info btn-sm">Edit</a>
                    <form class="d-inline-block" action="/products/<%=product._id%>?_method=DELETE" method="POST">
                        <button class="btn btn-sm btn-danger">Delete</button>
                    </form>
<%}%>
```

task 79: abhi bhi we have to make 1 more check ki kya ye current user humara us product ka auther hai agar nhi hai to we will simply not let him delete or edit.

iske liye again we will use a middleware.

```
//middleware.js
//added poora
const isProductAuthor = async(req,res,next)=>{
//to get the id of thatparticular product
    let {id} = req.params;
    const product = await Product.findById(id);
    console.log(product.author);
    console.log(req.user);
    if(!product.author.equals(req.user._id)){
        req.flash('error' , 'you donot have the permission to do that');
        return res.redirect(`/products/${id}`);
    }
    next();
}

//chnaged below
module.exports = {validateProduct ,validateReview , isLoggedIn , isSeller , isProductRouter} ;
```

go to product.js and add the middleware

```
//product.js

//delete mei added middleware
router.delete('/products/:id',isLoggedIn,isProductAuthor, async (req, res) => {

    try {
        const { id } = req.params;
        await Product.findByIdAndDelete(id);
        res.redirect('/products');
    }
    catch (e) {
        res.status(500).render('error',{err:e.message})
    }
});
```

Version 7

task 80:

humne **home.ejs** banaya and usse root route i.e **'/'** par render kiya.

```
//home.ejs
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shopping Cart</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/css/bootstrap.min.css"
        integrity="sha384-r4NyP46KrjDleawBgD5tp8Y7UzmLA05oM1iAEQ17CSuDqnUK2+k9luXQOfXJCJ4I" crossorigin="anonymous">
    <link rel="stylesheet" href="/css/home.css">
</head>

<body class="d-flex text-center text-white bg-dark">
    <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column">
        <header class="mb-auto">
            <div>
                <h3 class="float-md-left mb-0">Shopping Cart</h3>
                <nav class="nav nav-masthead justify-content-center float-md-right">
                    <a class="nav-link active" aria-current="page" href="#">Home</a>
                    <a class="nav-link" href="/products">Products</a>
                    <% if(!currentUser) { %>
                    <a class="nav-link" href="/login">Login</a>
                    <a class="nav-link" href="/register">Register</a>
                    <% } else { %>
                    <a class="nav-link" href="/logout">Logout</a>
                    <% } %>
                </nav>
            </div>
        </header>
        <main class="px-3">
            <h1>Shopping Cart</h1>
            <p class="lead"> Welcome to Shopping Cart! <br> Jump right in and explore our many products. <br>
                Feel free to add some of your own and comment on others!</p>
            <a href="/products" class="btn btn-sm btn-secondary font-weight-bold border-white bg-white ">View
                Products</a>
        </main>

        <footer class="mt-auto text-white-50">
            <p>All Right Reserved &copy; 2023 </p>
        </footer>


    </div>


    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/js/bootstrap.min.js"
        integrity="sha384-oesi62hOLfzrys4LxRF63OJCXdXDipiYWBnvTl9Y9/TRlw5xlKIEHpNyvvDShgf/"
        crossorigin="anonymous"></script>
</body>

</html>
```

```
//home.css
body {
    height: 100vh;
    background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)),
      /*url("https://images.unsplash.com/photo-1559521783-1d1599583485?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1950&
        url("https://images.unsplash.com/photo-1561715276-a2d087060f1d?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2.1&aut
    background-size: cover;
    background-position: center;
    text-shadow: 0 0.05rem 0.1rem rgba(0, 0, 0, 0.5);
    box-shadow: inset 0 0 5rem rgba(0, 0, 0, 0.5);
  }
  .cover-container {
```

```
    max-width: 60vw;
  }

  .nav-link {
    padding: 0.25rem 0;
    font-weight: 700;
    color: rgba(255, 255, 255, 0.5);
    margin-left: 1rem;
    border-bottom: 0.25rem solid transparent;
  }

  .nav-link:hover {
    color: rgba(255, 255, 255, 0.5);
    border-bottom-color: rgba(255, 255, 255, 0.5);
  }

  .nav-link.active {
    color: white;
    border-bottom-color: white;
  }

  .btn-secondary,
  .btn-secondary:hover {
    color: #333;
    text-shadow: none;
  }
```

```
//app.js
const authRoutes = require('./routes/auth');

app.get('/' , (req,res)=>{ //added from this
    res.render('home');
})
```

task 81: Like functionality

(khud try krlena baccho notes mei nhi likhra)

task 82: adding to cart

ab cart kiski hogi user ki hogi ofcourse to it will be an array only so hum schema mei change krenge pehle jiske andar hum products add krege.

```
//product.js

role:{
        type:String,
        default:'buyer',

    },
    cart:[ //adding here
        {
            type: mongoose.Schema.Types.ObjectId,
            ref:'Product'
        }
    ]
```

task 83: ab mera kaam kya hai mai simply jab bhi **add to cart button** ko hit kru to simply mai usme route lagau and simply navbar ka **UI** change kru.

make **cart.js** inside routes

```
//routes --> cart.js

const express = require('express');
const router = express.Router();



module.exports = router;
```

now use it in **app.js**

```
//app.js

const productRoutes = require('./routes/product');
const reviewRoutes = require('./routes/review');
const authRoutes = require('./routes/auth');
const cartRoutes = require('./routes/cart');   //added


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

// middle express
app.use(productRoutes);
app.use(reviewRoutes);
app.use(authRoutes);
app.use(cartRoutes);         //added
```

ab i have to add the product ki id in user ki array so we have to make sure ki user **loggedIn** ho.

ab simply mai user ko find krke via his id , simply uske andar cart ki array mei add krunga product ki id.

```
//cart.js
const {isLoggedIn} = require('../middleware'); //adding
const Product = require('../models/product');   //adding
const User = require('../models/user');    //adding

router.post('/user/:productId/add' , isLoggedIn , async(req,res)=>{
    let {productId} = req.params;
    let userId = req.user._id;
    let product = await Product.findById(productId);
    let user = await User.findById(userId);
    user.cart.push(product);
    res.redirect('/user/cart');
})
```

ab simply you have to hit the get route for redirecting.

and Now,

vahan you have to hit the route jiske andar aapko simply apne us user ko bhejna hai, taaki us user ko hum find krpaae and simply and vahan products ko show krdo.

aap idhar simply send krdo **cart.ejs** ka file and usme saare cart vaale cheezo ko display kro.

```
//cart.js

router.get('/user/cart' , isLoggedIn , async(req,res)=>{
    let user = await User.findById(req.user._id)
    res.render('cart/cart' ,{user});
})
```

task 84: ab simply views ke andar **cart** naam ka folder banao jiske andar **cart.ejs** naam ki file and jiske andar aap simply.

```
//views --> cart --> cart.ejs
<% layout('layouts/boilerplate') -%>

<ul>
    for(let item of user.cart){
        <li><%=item.name%> --> <%=item.price%>  </li>
    }
</ul>
```

ab aap navbar mei change kro to hit the route.

task 85:

```
//navbar.ejs

<%} else{%>
          <a href="#" class="nav-link text-capitalize">Hello <%=currentUser.username%></i>
//added here <a href="/user/cart" class="nav-link"><i class="fas fa-shopping-cart"></i>
//chnage  <sup><span class="badge bg-danger"><%=currentUser.cart.length%></span></sup>
          </a>
          <a href="/logout" class="nav-link">Logout</a>
<%}%>
```

ab aapko add to cart ko **show.ejs** se change krna hai so simply go to show.ejs and change.

lekin ye merko as post request bhenjni hai so merko isse **button** banana pdhega.

```
//show.ejs

//added button instead of anchor
<form class="d-inline-block" action="/user/<%=product._id%>/add">
                    <button>Add To Cart</button>
</form>
<%if(currentUser && currentUser.role === 'seller'){%>
```

```
//cart.js
 //you need to populate as well

const user = await User.findById(req.user._id).populate('cart'); //added
```