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});
    }
})
```