HTTP is a stateless protocol.  
  
This means every request is independent.  
  
The web application server can’t tell if 2 requests came from the same browser or user.  
  
But the users aren’t stateless.  
  
No one wants to log in to your application every time they make a request.  
  
So - how do you help them?  
  
👉 One solution is to use cookies.  
  
Yes, cookies! But not the one you eat when you’re hungry.  
  
A cookie is basically a key-value pair that’s stored on the browser.  
  
How do they work?  
  
- The user logs in to your frontend application.  
  
- The frontend sends the request to the backend server  
The backend server generates a cookie  
  
- It sets the cookie on the browser via the Set-Cookie response header.  
  
- The user makes a new request to view a different page.  
  
- The front end sends the request to the backend and includes the Cookie as part of the header.  
  
- The server checks the cookie for the user and responds with the required data.  
  
Sounds good, doesn’t it?  
  
But there’s a major issue with using cookies.  
  
Cookies are accessible via the browser. You can modify the cookie information.  
  
That’s why it’s not a good idea to use cookies for storing sensitive data about the users.  
  
This is where sessions come into the picture.  
  
👉 The session contains a unique set of characters to identify the user.  
  
It works as follows:  
  
- The user makes a login request  
  
- The frontend sends the request to the backend server  
  
- The backend creates a session using a secret key and stores it in some sort of session storage (database or cache)  
  
- Next, the server sends a cookie back to the client  
  
- However, the cookie contains the unique identifier for the session  
  
- The user makes a new request to view another page.  
  
- The browser sends the session ID as part of the cookie.  
  
This time only the server can validate whether the session is valid.  
  
🧰 Few important points to mention over here:  
  
✅ Cookies can have a “Secure” flag indicating that it should only be sent over HTTPS. This is good for security reasons.  
  
✅ Also, “HttpOnly” cookies restrict the cookie’s access to JavaScript reducing the risk of XSS attacks.  
  
✅ Cookies (especially 3rd party cookies) raise a bunch of privacy concerns because they can be used to track user behavior.  
  
✅ While cookies can be made secure, server-side sessions provide additional layers of security against CSRF attacks and handling sensitive information  
  
✅ Also, server-side sessions can be centrally managed. This means you can invalidate sessions, expire or revoke them if needed.  
  
===  
  
That's all for now!  
  
If you enjoyed this deep dive into Cookies and Sessions, don’t forget to:  
  
- Give the LIKE button a cookie  
- REPOST so that everyone gets the right session ID in the cookie  
- Follow me for more posts like this  
  
Also, subscribe to my free newsletter to join 1700+ subscribers receiving practical software design tips:  
  
<https://lnkd.in/dHXxbr4a>

Activate to view larger image,

