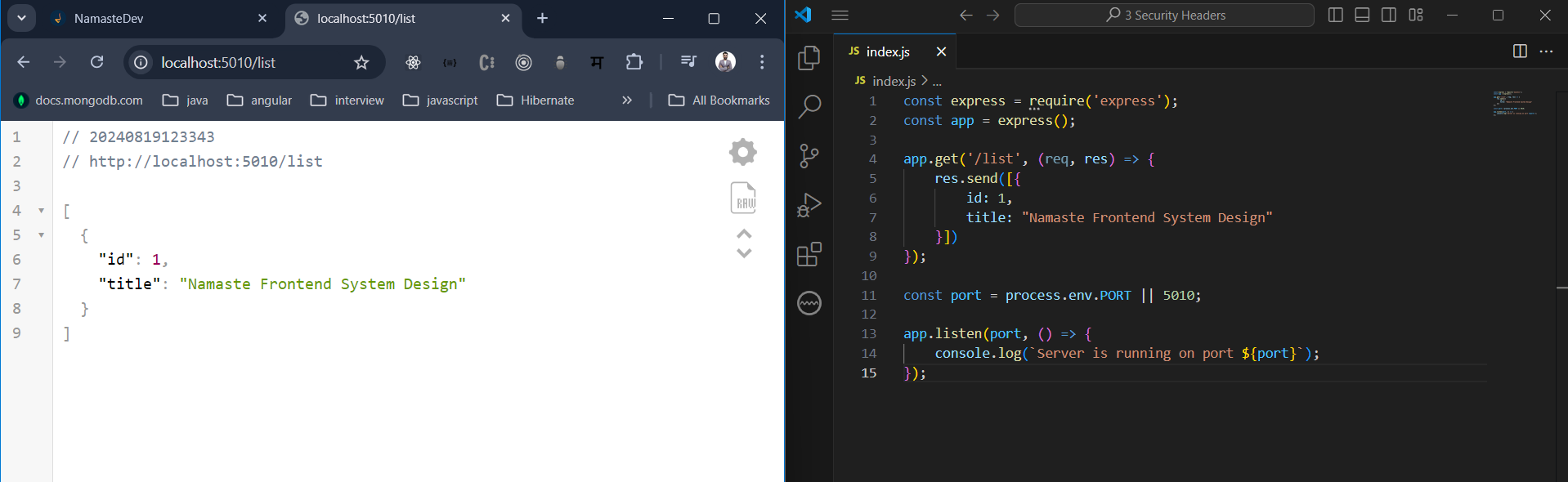
Security Headers –

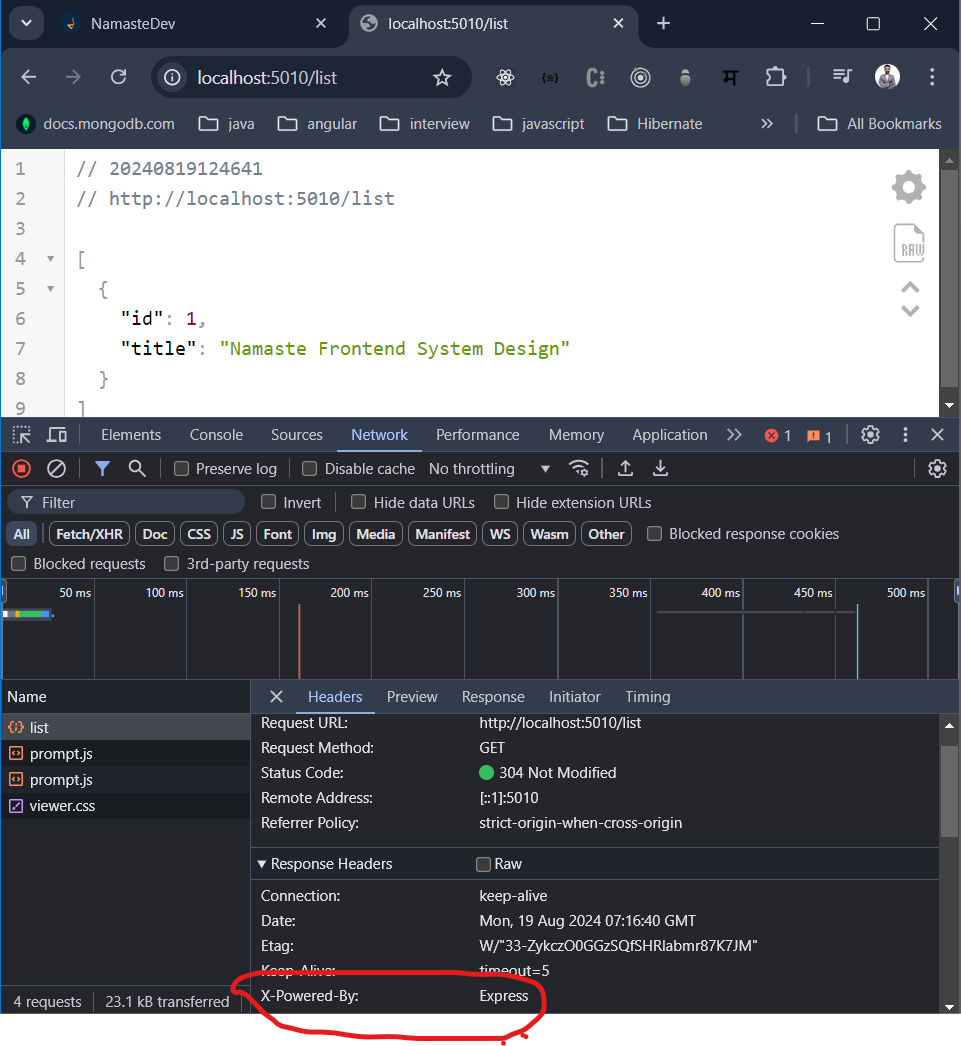
1. X-Powered-By – this header is basically telling you your application is built in using what kind of server.

Example –



This is a normal express server it gives the list of courses with the title of namaste frontend system design.

Lets run and check what we get it in the response, consider I am using some google or github api’s now do you know which server tech google uses? Do you ever know what are the server serving by facebook? You don’t know until and unless if they tell you what technology they are using in the architectural design.

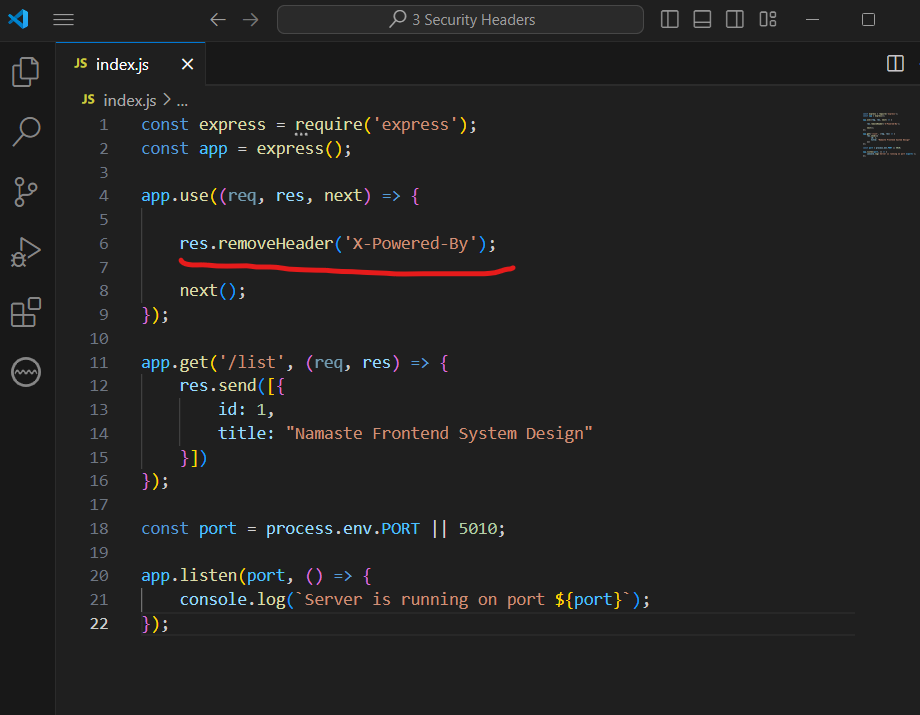


But lets see in our case it is same or different, so this is the first request which is the list so we can go and see what we get in the response, so in the response we can see something

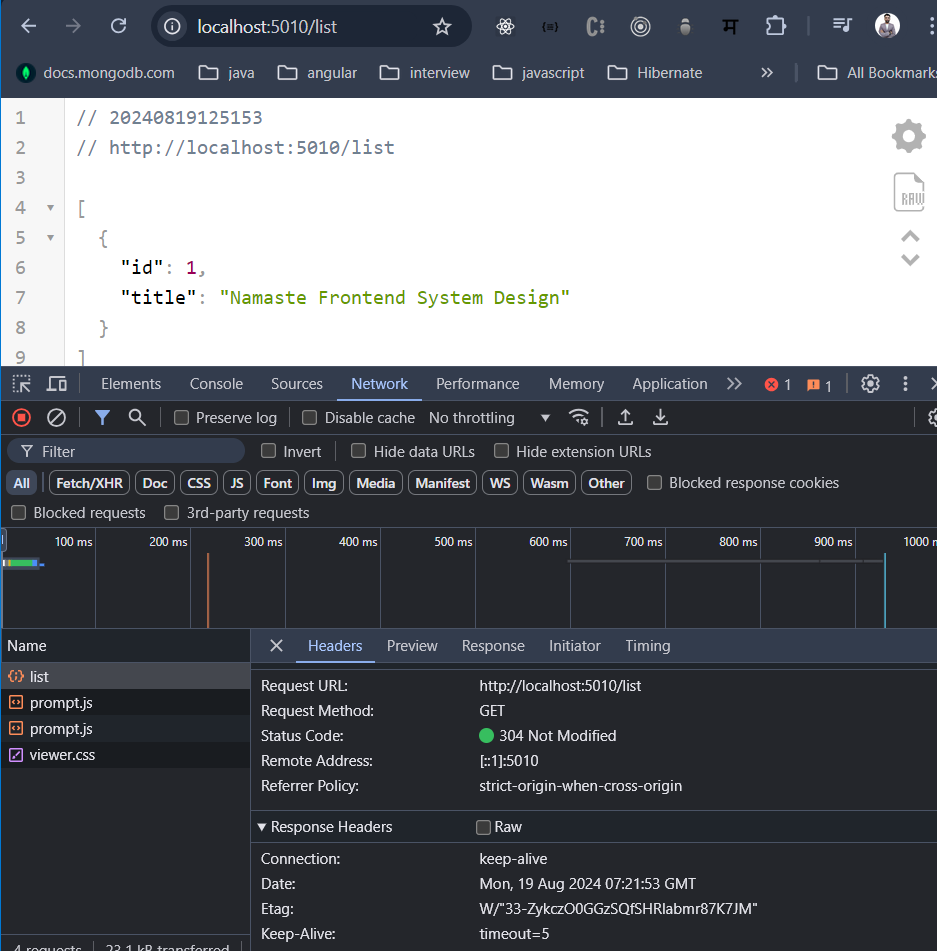
X-Powered-By is express server. How did it get to know that I am using express server.

This information is leaked, now consider there are some security vulnerabilities which are going into express server and people are expert on that then they can try to exploit it.

So that is why we need to remove this header from the response, and for that we use one middleware function in the server side index.js file.



Output –



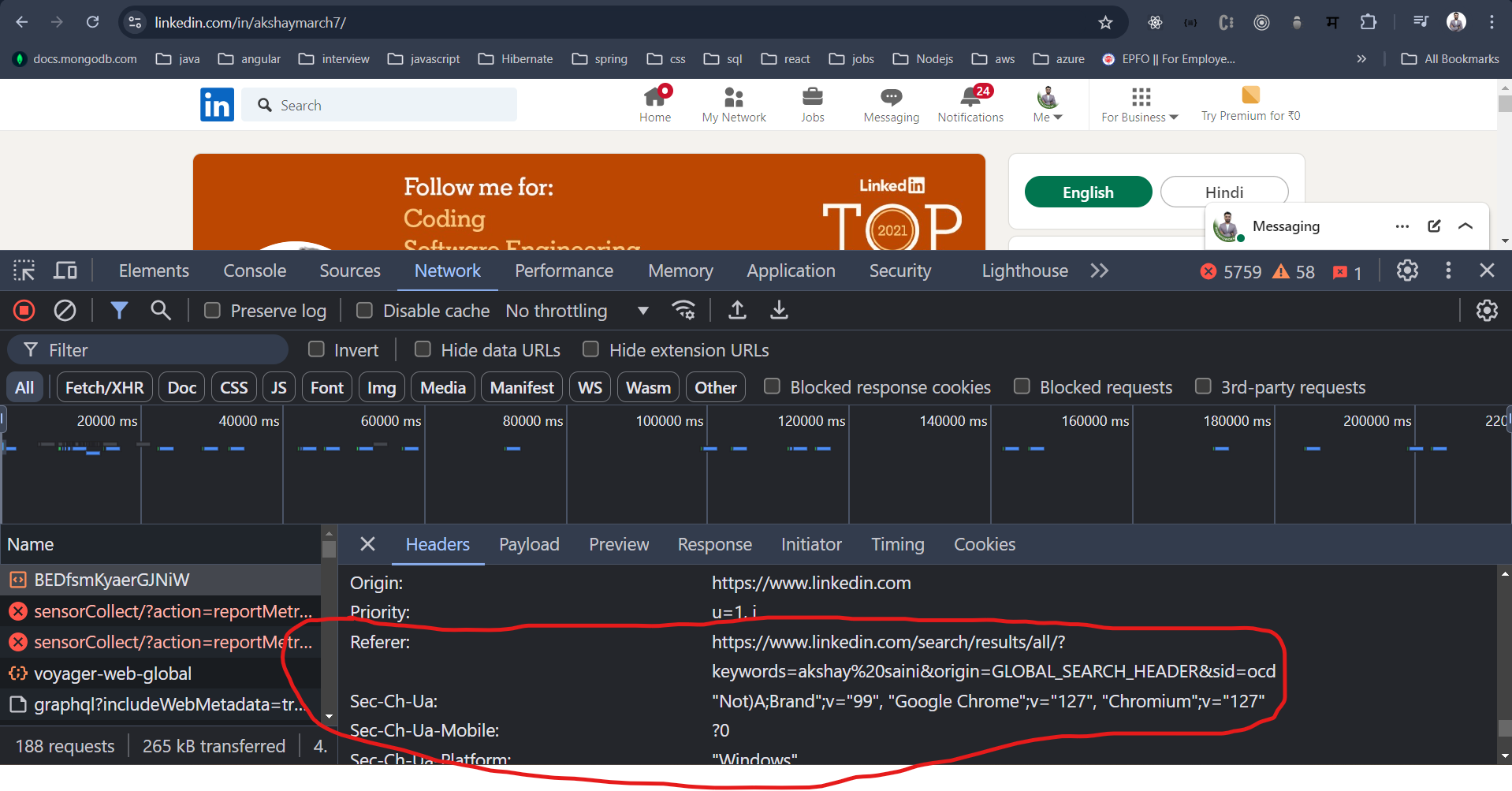
So please do not expose this information which is X-Poewered-By which is very very important.

1. Referrer-policy –

When you share your YouTube video or a LinkedIn post from one platform to the other platform, so end of the day in your telemetry in your matrix you are able to see from which sources these traffics are coming, reason being whenever we do click whenever we do navigate whenever we load certain things all of this redirection basically brings you something called from where this traffic is coming.

Example you are on page which is related to Facebook and you are seeing some link you clicked on link and you went to linkedin.com now on the facebook you are on some private path or some route when you go to the linkedin what do you wanted to track most of the time from where the traffic is coming either it is coming from facebook or you are coming from what such cases can be handled by just configuring the right set of your header policies.

Which is your referrer-policy is something we can tweak and say that I don’t wanted to tell all the information, I just wanted to send may be just from origin.



If you see referrer is /search/result/all and there is some and some other information we got redirected from the previous page where the previous page was is link of referrer, it contain some information it may be sensitive or it may be overwhelming information, why do we wanted to do similar kind of thing where we are doing cross domain, so we wanted to ensure that such thing never ever happen.

So, there are many referrer policies –

**Referrer-Policy:** **no-referrer** (when I do not want to send any of the referrer)

: **no-referrer-when-downgrade** (I do not want to send referrer when I am downgrading, I am on https but when you are redirecting on the http probably I do not want to do)

: **origin** (I just wanted to send the origin)

: **origin-when-cross-origin** (within the linkedin I am okey to send the entire url no problem when from going to linkedin to whatsapp or linkedin to facebook so in cross domain I wanted to ensure only origin happens.)

So, this is one the powerful thing that many people do not take it seriously basically, and they ended up into getting those problem.

So, referrer policies its seems like it basically you end up reviling lot of interesting information of your own application if you do not set up right set of referrer policy, you are licking your entire information when you are navigating from one page to another page, this is something you have to take care off.

1. X-Content-Type-Options –

Consider example you have a client which is your browser and you have a server, client is basically expecting give me a page which is .jpg, what happened someone tempered this request, what happened instead of serving the data a person comes in between and it actually temper the data of it, it actually injected a html code inside this or anything which is executable, now in this case browser used to do Is called sniffing what it meant back when this response basically goes back to client browser just check two thing which is the content type of the response that you are getting and the 2nd thing is what is the content I am getting sometime depending on your content browser understand oh you actually wanted to send the html, let me upgrade that and consider it as an html but the original file was requested was jpg.

Now what happen say suppose someone actually man in a middle injected some harmful script those are going to be executed and your system is going to be down. You did not expect that to happen because your browser actually changes the sniffing type change the content type and consider the content type based on the content instead of the content type which was in a response.

To avoid such things what we do we can set the content-type-option that you can set over here.

   res.setHeader('X-Content-Type-Options', 'nosniff');

1. X-XSS-Protection –

Now you say that I wanted to disable this at a browser level, now browsers are very smart browsers are say that if you set this header I am going to not allow anything which is non xss attack, anyone basically try to inject the malware script I am not going to allow that.

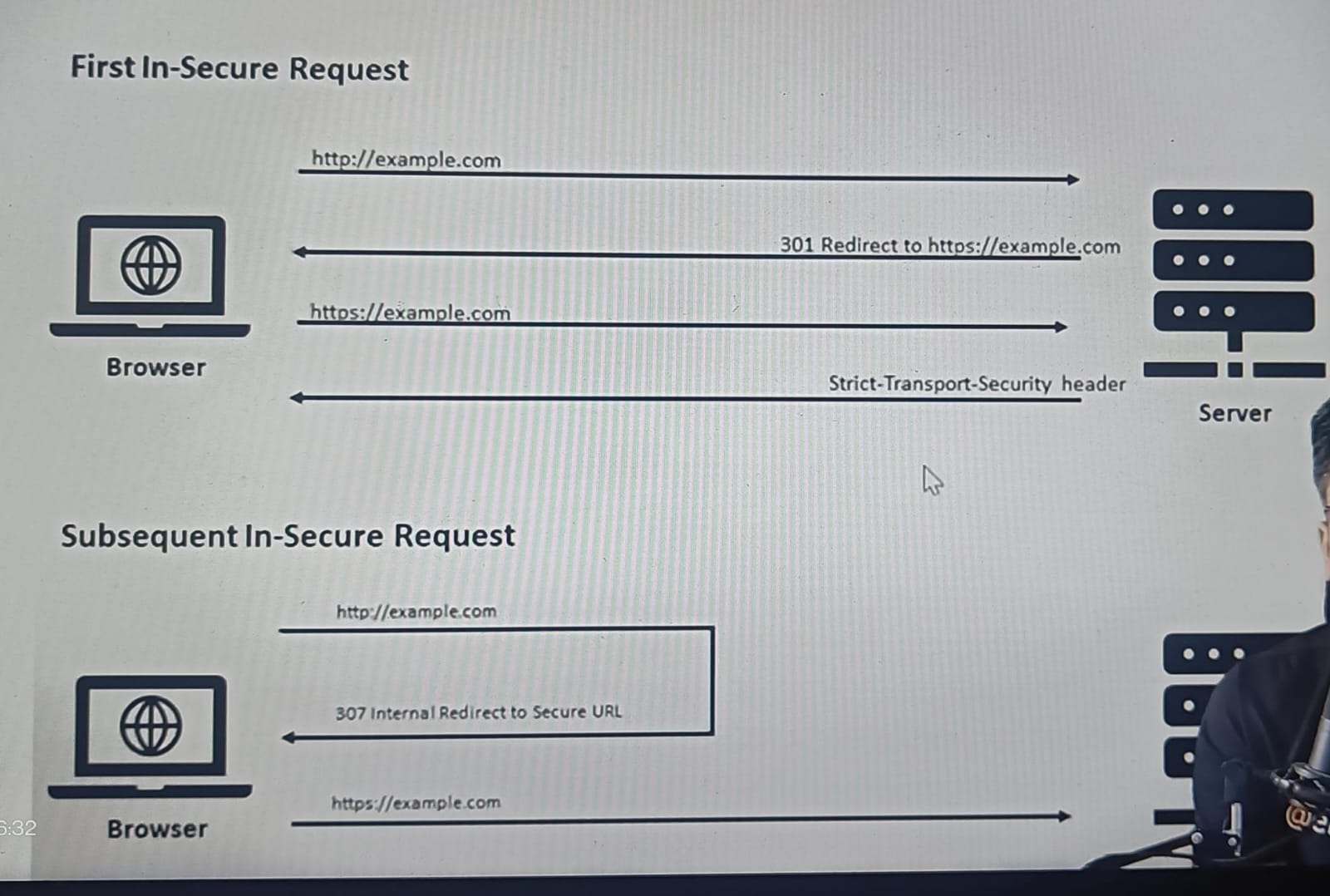
X-XSS-Protection : 0 (I do not want)

: 1 (I want to have)

: 1; mode=block (if I find anyone doing xss I will block)

: 1; report=<reporting-uri> (for report)

1. HSTS (Strict Transport Security Header) –



This header says that consider example, you have a application which is running on https and you have something which is on http also now what you want any request which is coming from http you want these request to be automatically redirected to https so that anyone is not able access the http platform.

In that case we have 2 ways, if someone tries to access http it goes to the server now on the server you have to write the logic which says that if the request comes for the first time as an http please redirect to https, now https will be basically go to the server and server will set strict transport http header.

What it meant by next time if you will come I wont let it go from the browser itself, any subsequent request browser itself says that ohh there is some header which already set with respect to that I am not going to allow, I know internal redirection with the 307 and myself basically take care of converting from http to https and then basically I make the request to the https only.

    res.setHeader('Strict-Transport-Security', 'max-age=31536000; includeSubDomains; preload');