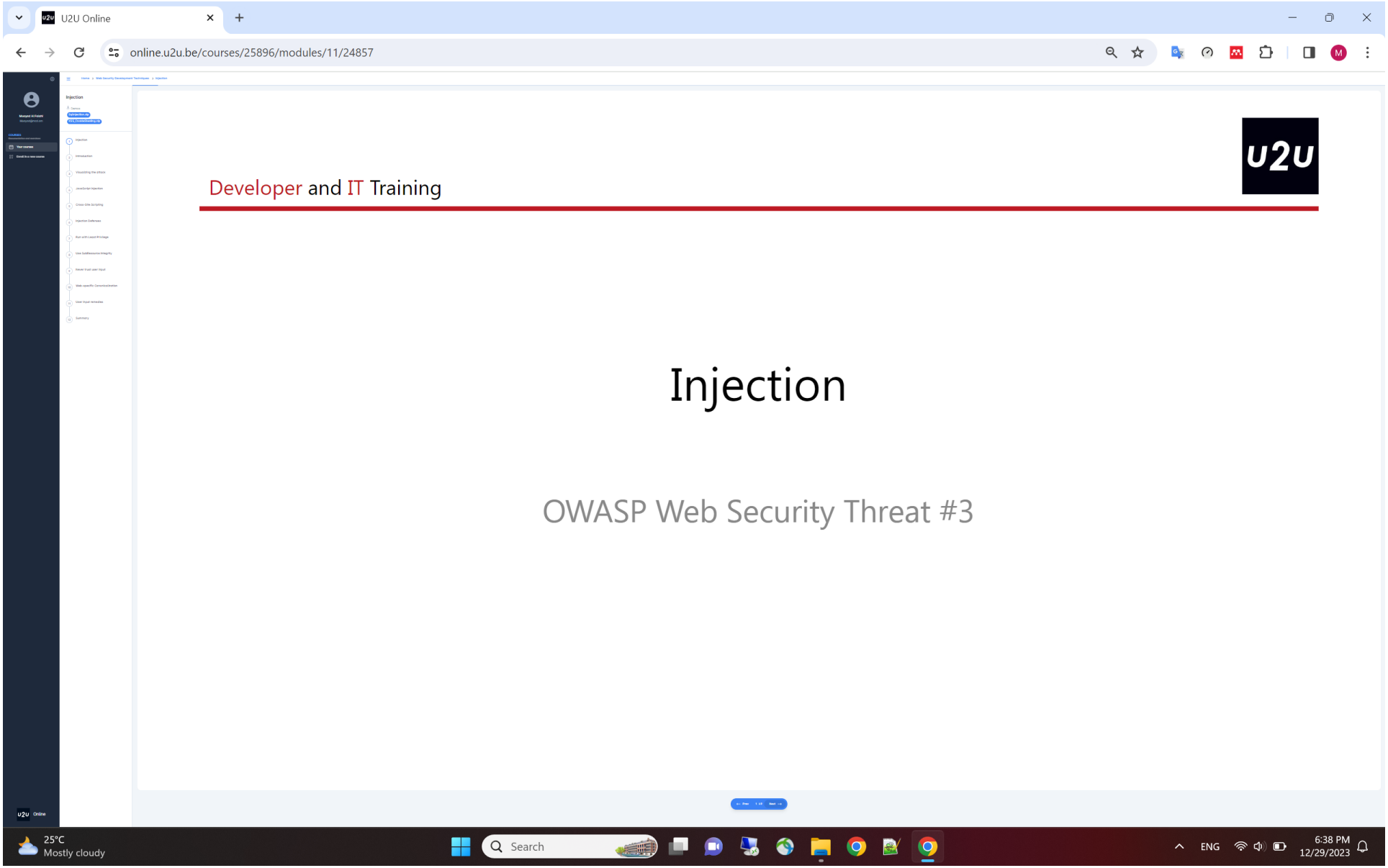
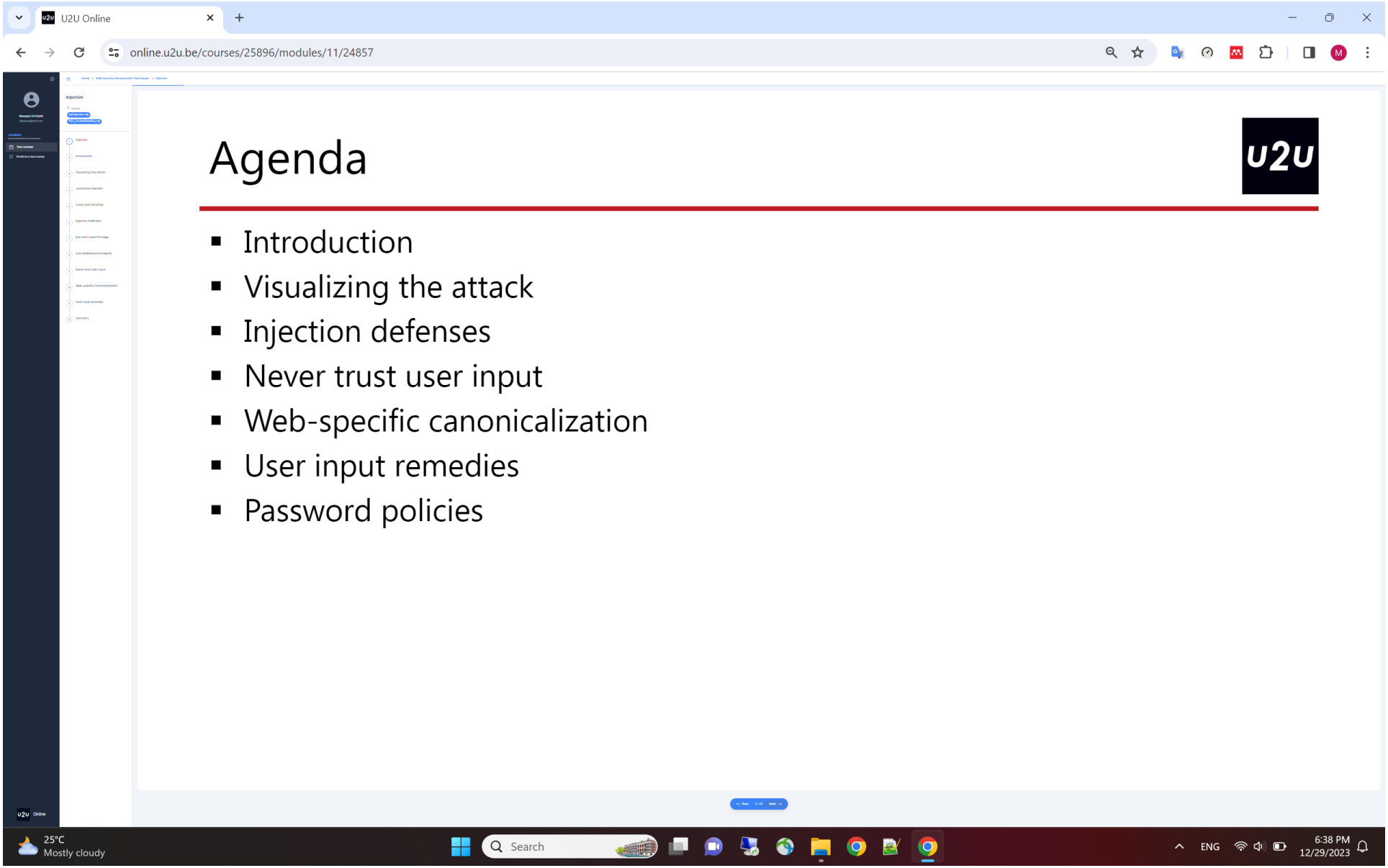
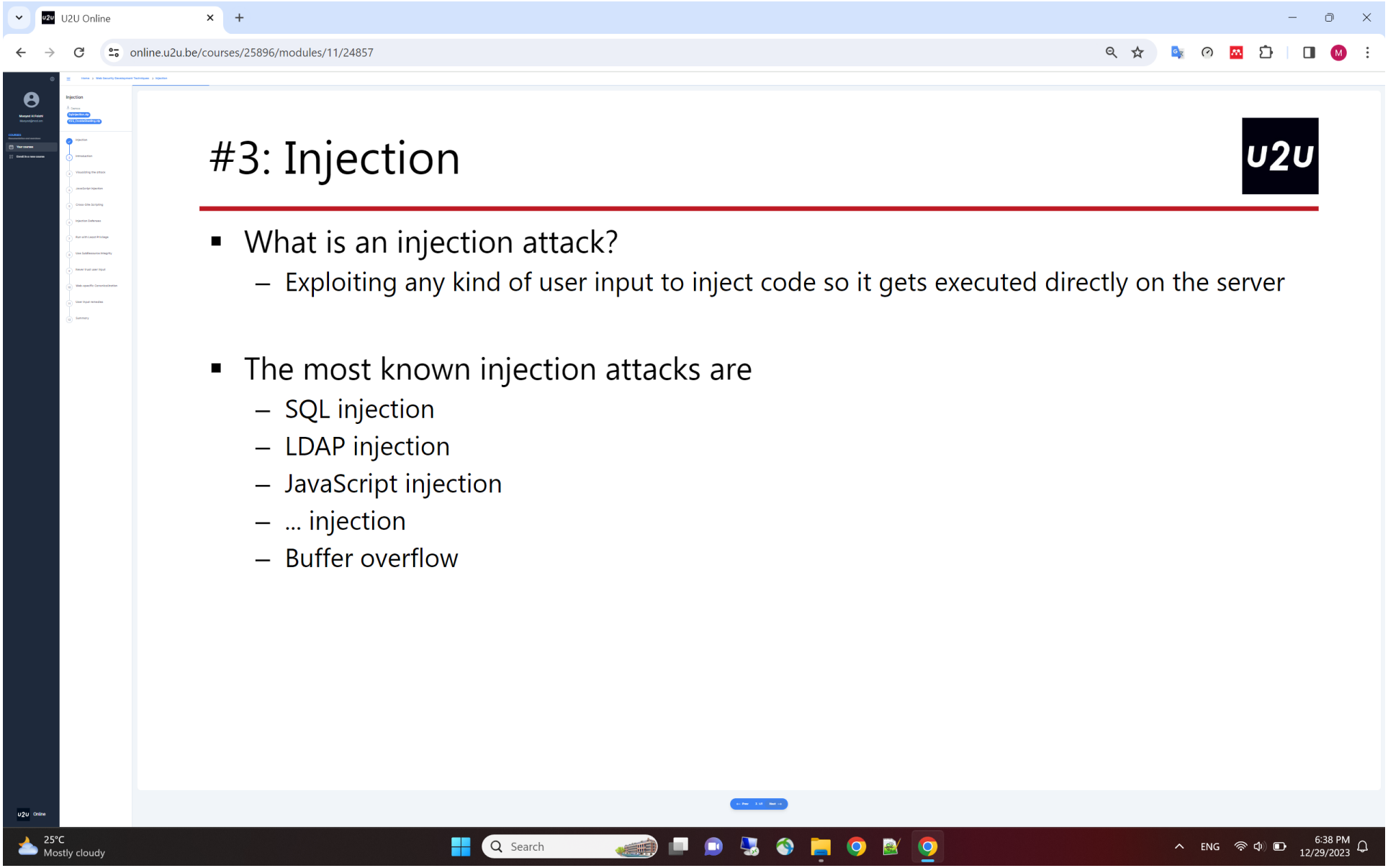
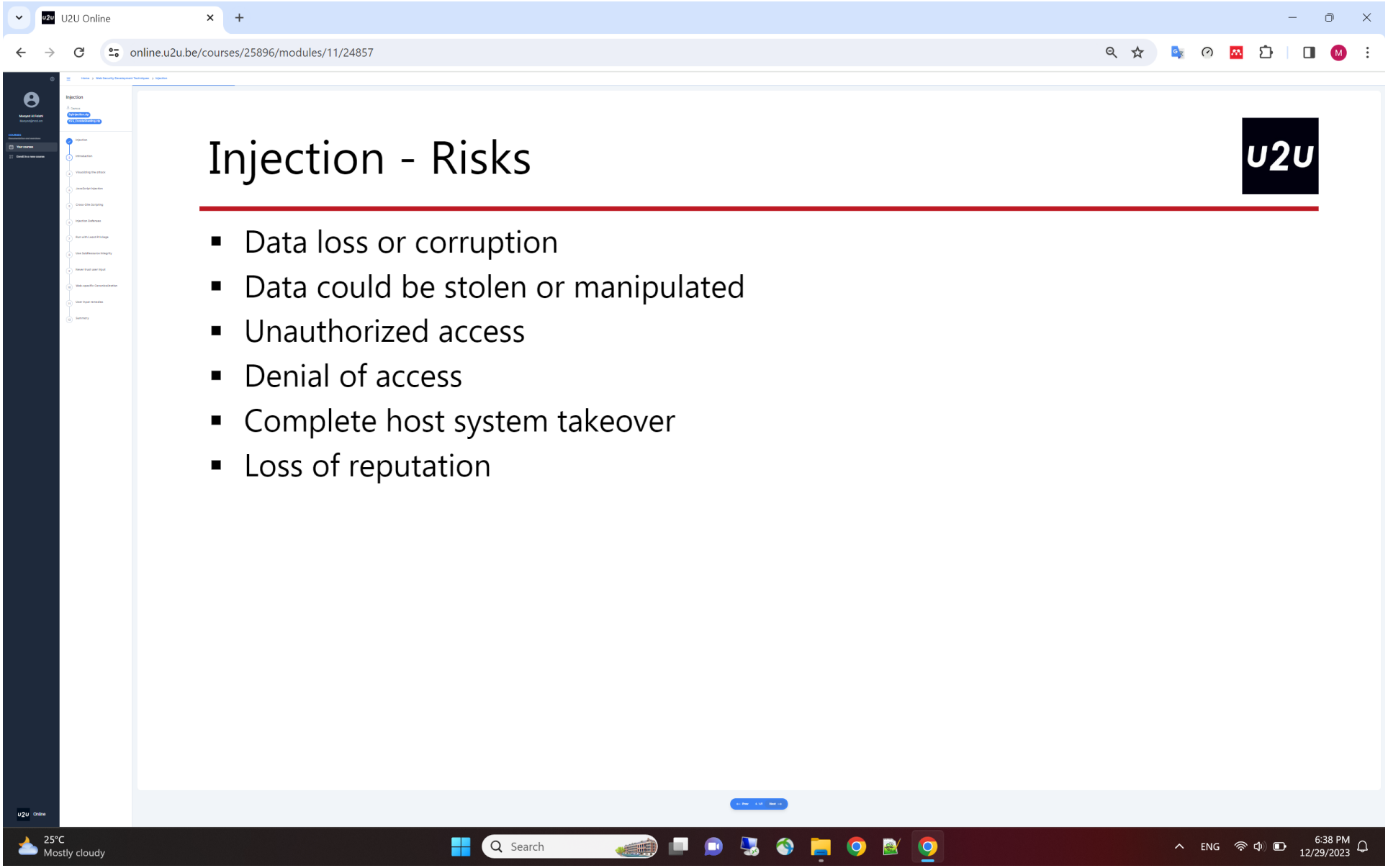
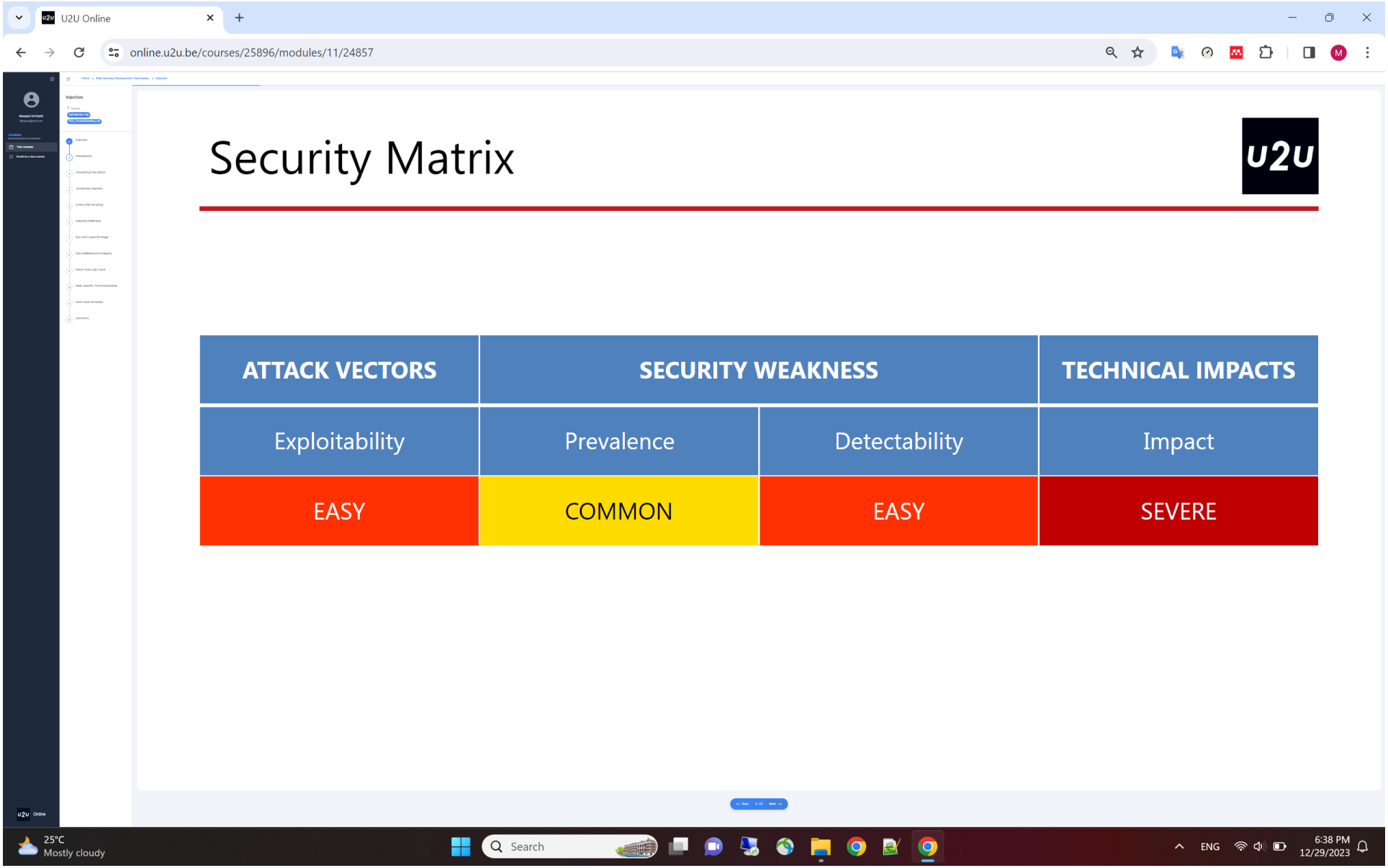

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Developer and IT Training
u2u
Injection
OWASP Web Security Threat #3
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Agenda
u2u
Introduction
Visualizing the attack
Injection defenses
Never trust user input
Web-specific canonicalization
User input remedies
Password policies
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

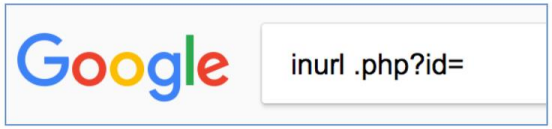

U2U Online
online.u2u.be/courses/25896/modules/11/24857
#3: Injection
U2U
What is an injection attack?
Exploiting any kind of user input to inject code so it gets executed directly on the server
The most known injection attacks are
SQL injection
LDAP injection
JavaScript injection
... injection
Buffer overflow
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Injection - Risks
U2U
Data loss or corruption
Data could be stolen or manipulated
Unauthorized access
Denial of access
Complete host system takeover
Loss of reputation
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

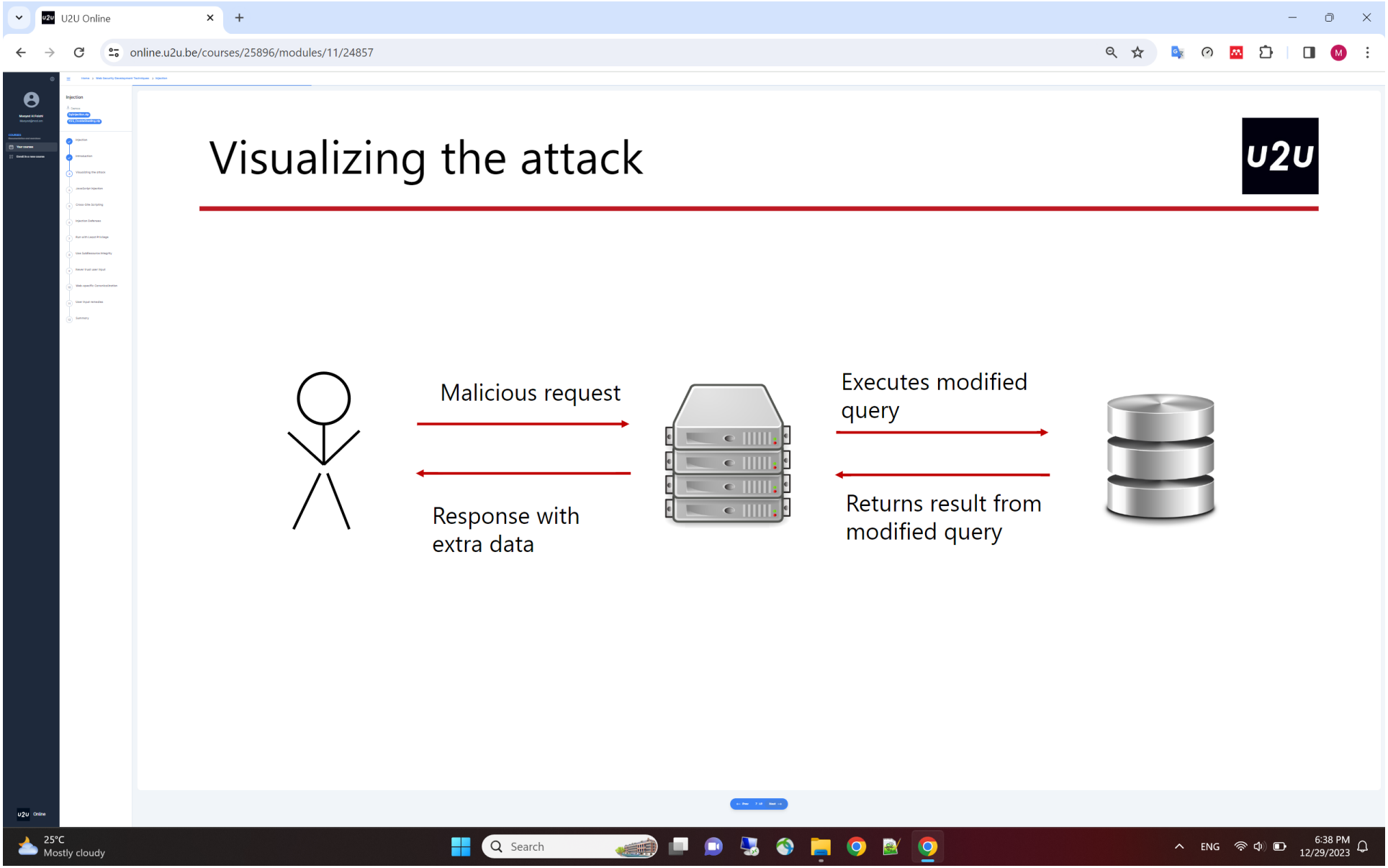

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Security Matrix
u2u
ATTACK VECTORS
SECURITY WEAKNESS
TECHNICAL IMPACTS
Exploitability
Prevalence
Detectability
Impact
EASY
COMMON
EASY
SEVERE
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

Prevalence: COMMON

- Try following google query:



- Pick a site, and modify id= in url


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Visualizing the attack
u2u
Malicious request
Response with extra data
Executes modified query
Returns result from modified query
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

SQL Injection

https://www.youtube.com/watch?v=_jKylhJtPmI

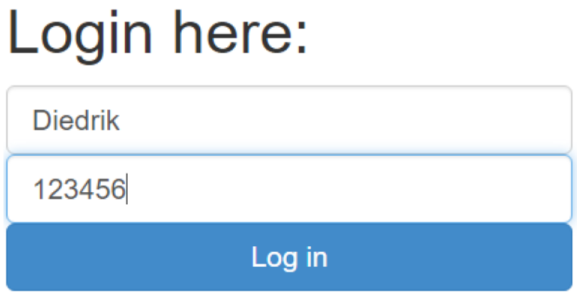
- Let's consider the following SQL command

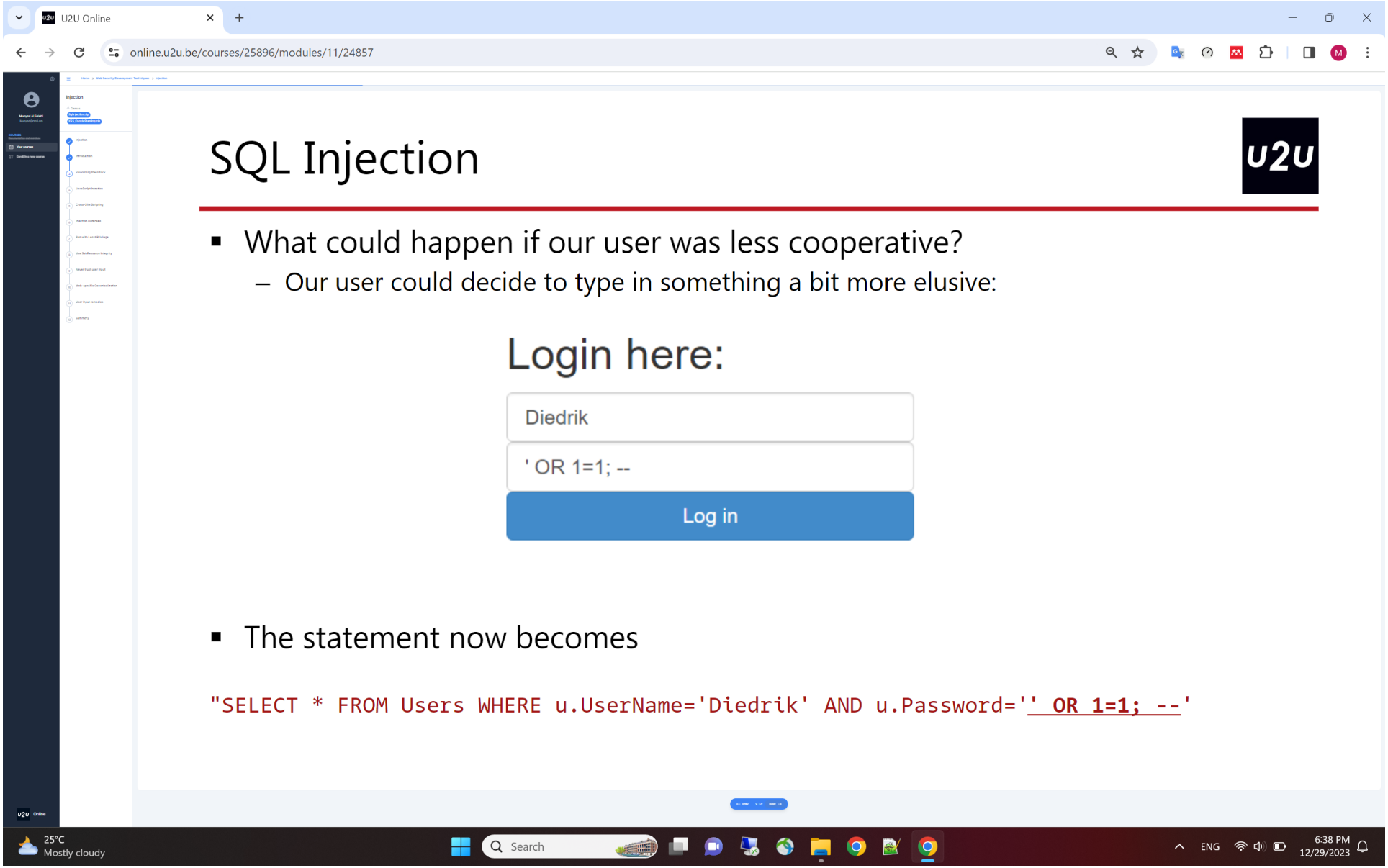
```
"SELECT * FROM Users WHERE u.UserName='" + username + "' AND u.Password='" + pw + "'"
```

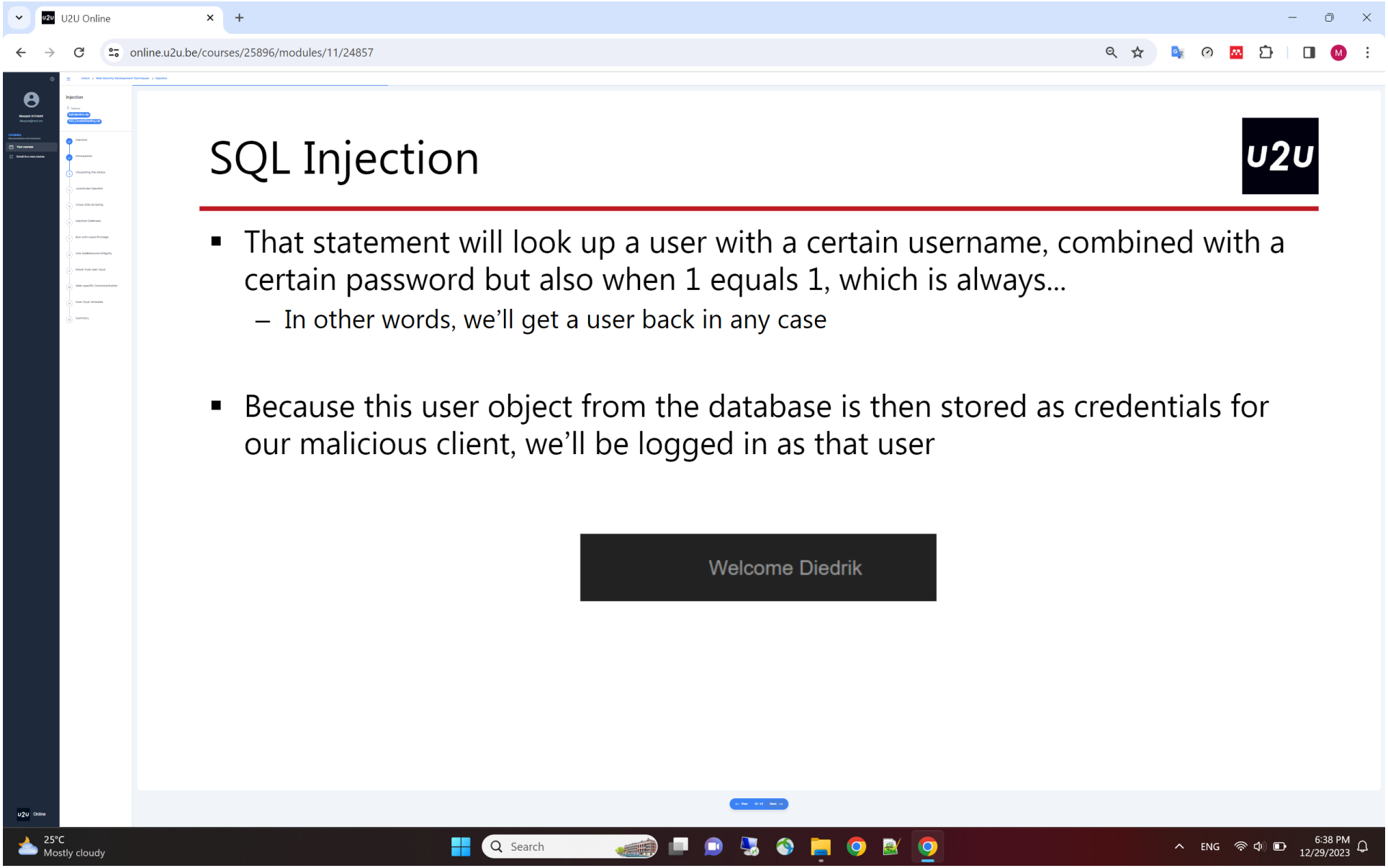
- The idea is to select a user entry matching a username
  - So a command could be turned into

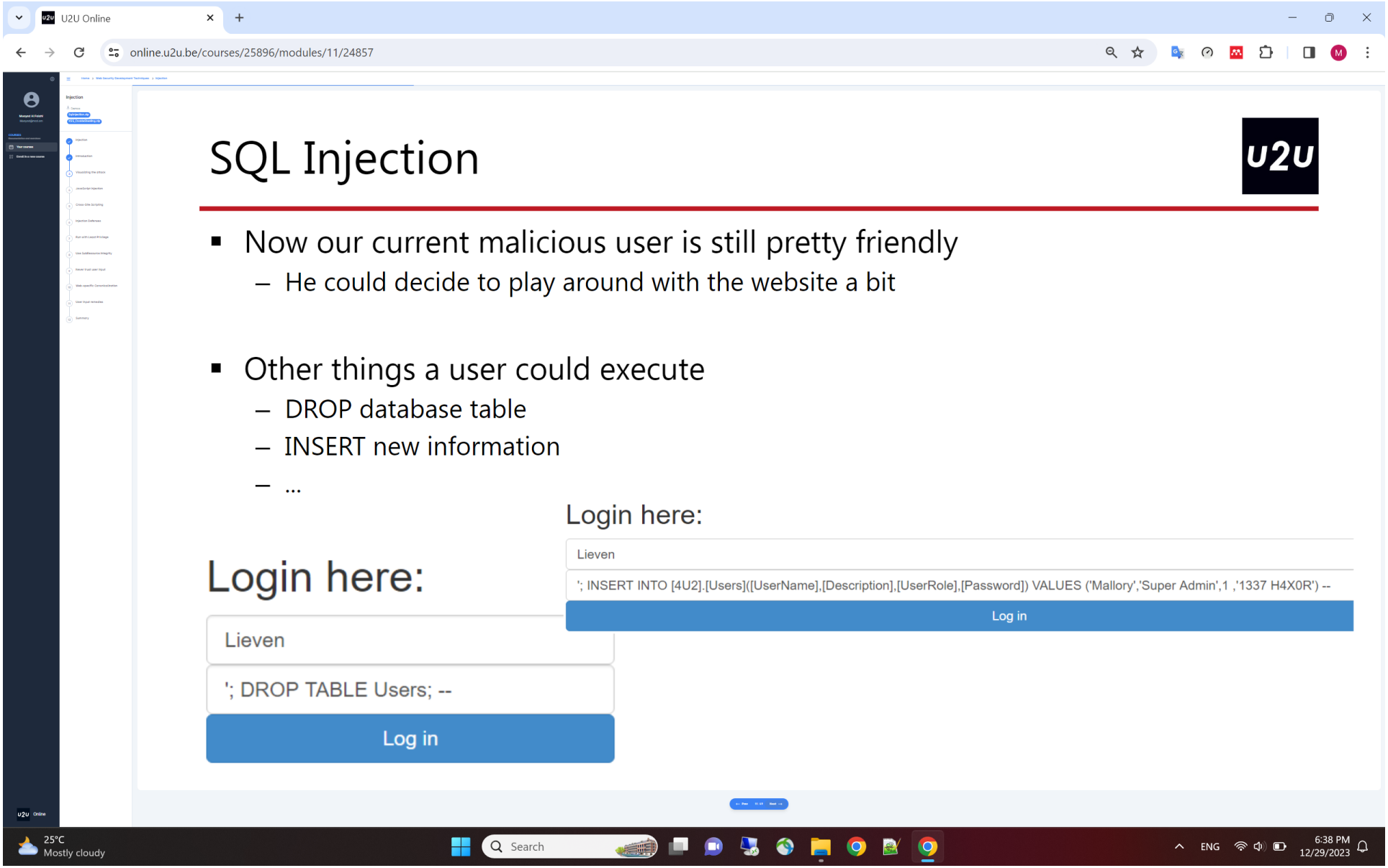
```
"SELECT * FROM Users WHERE u.UserName='Diedrik' AND u.Password='123456'
```

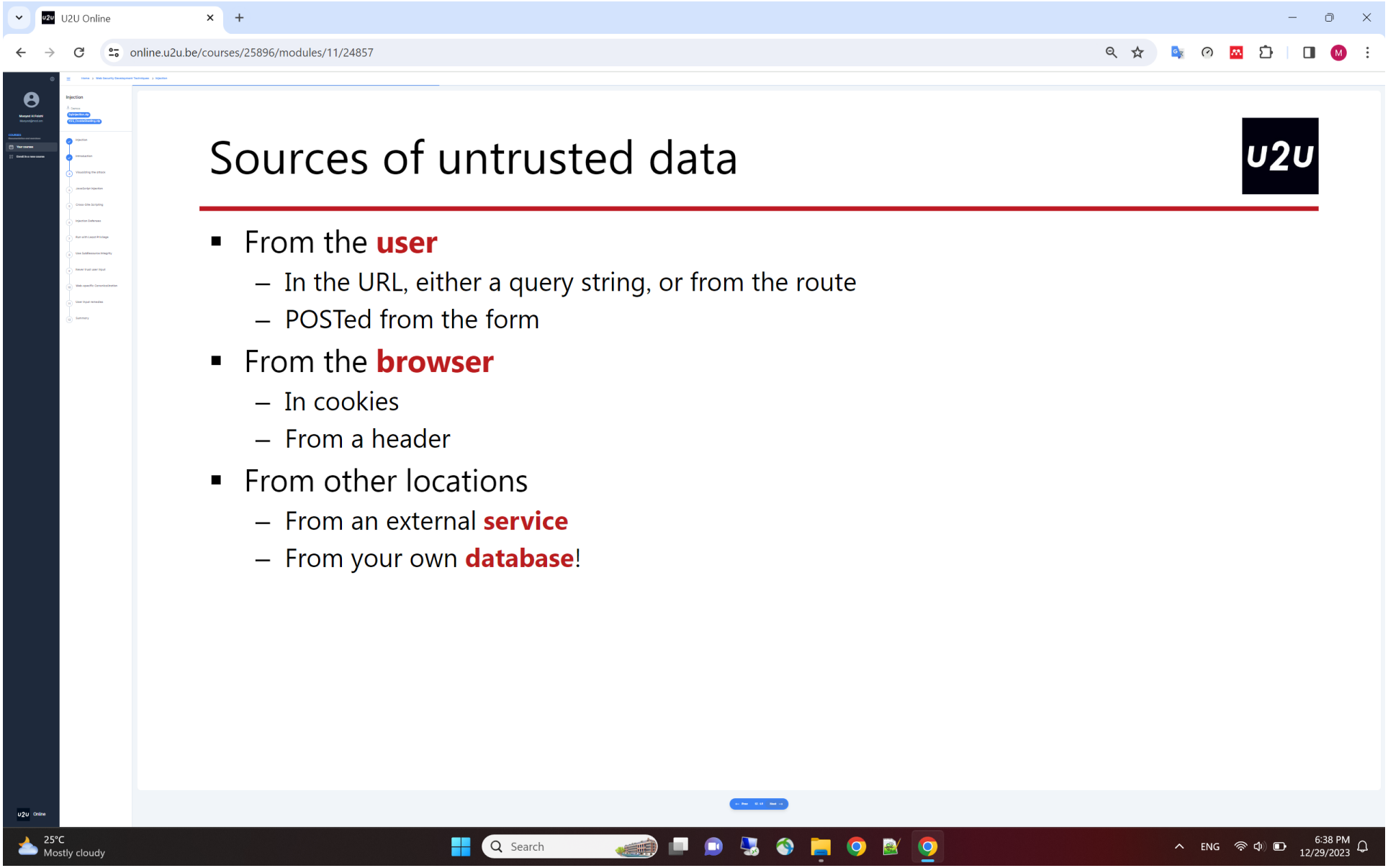
- Now this is of course when the user is friendly

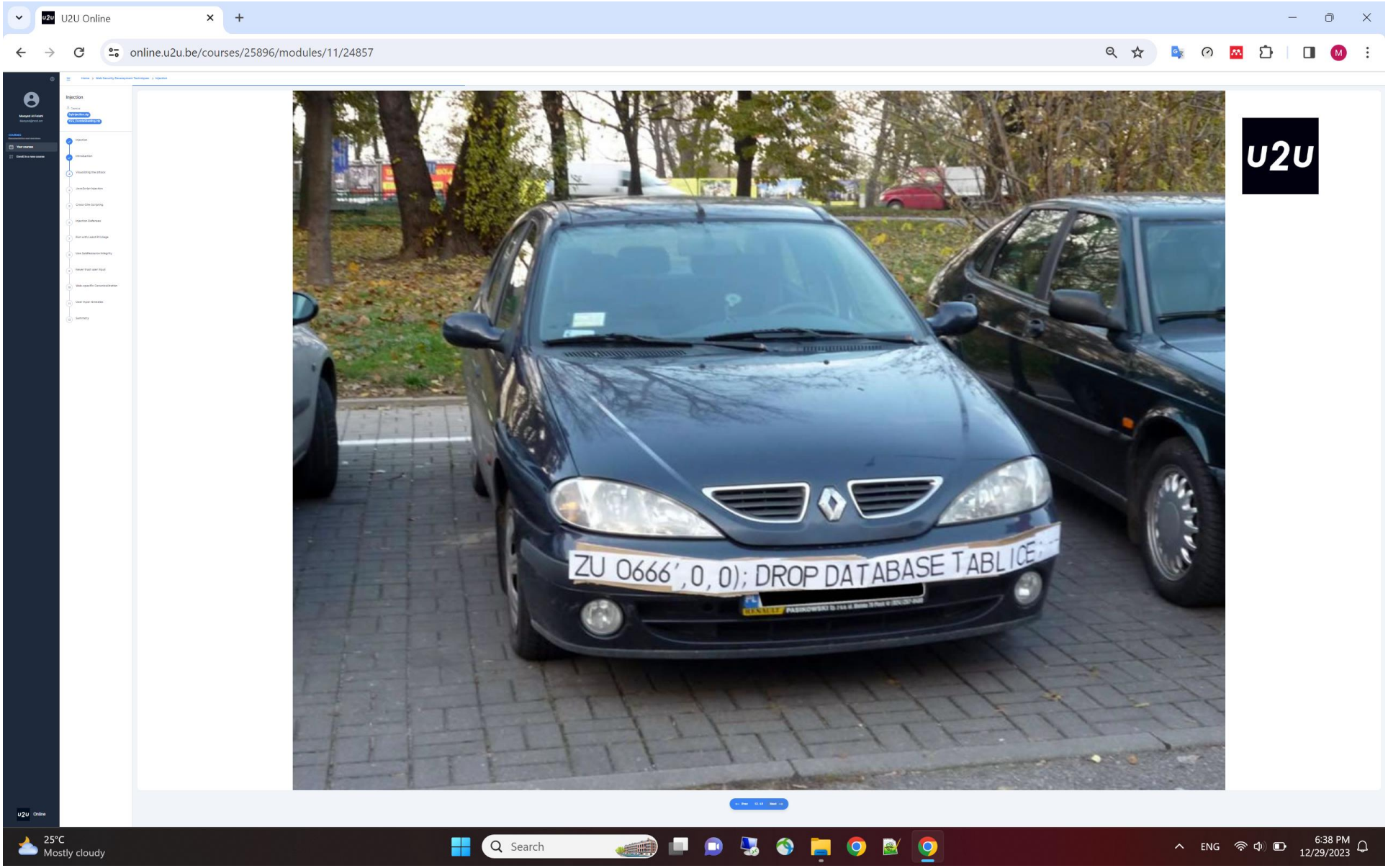


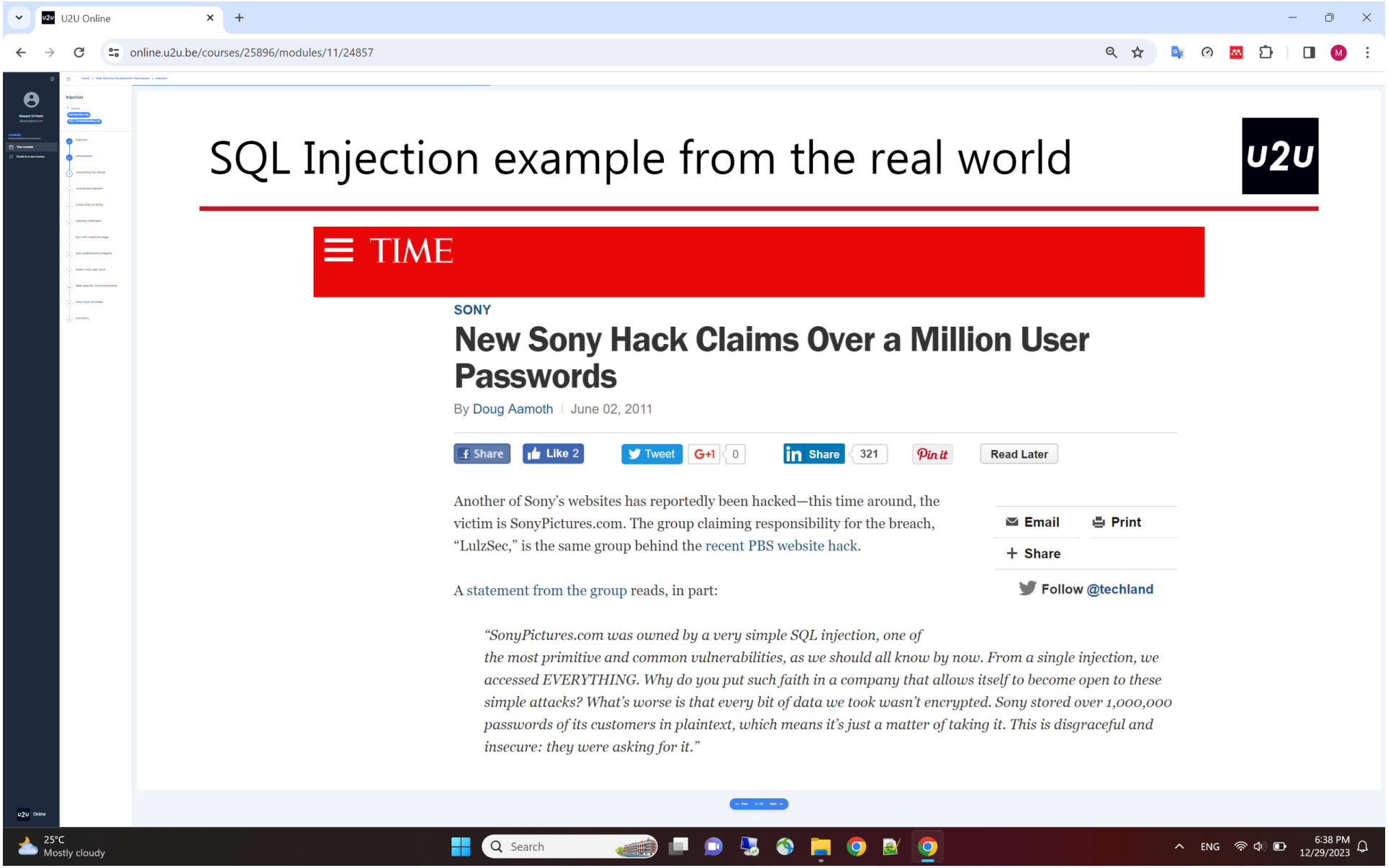

U2U Online
online.u2u.be/courses/25896/modules/11/24857
SQL Injection
u2u
What could happen if our user was less cooperative?
Our user could decide to type in something a bit more elusive:
Login here:
Diedrik
' OR 1=1; --
Log in
The statement now becomes
"SELECT * FROM Users WHERE u.UserName='Diedrik' AND u.Password='' OR 1=1; --'
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

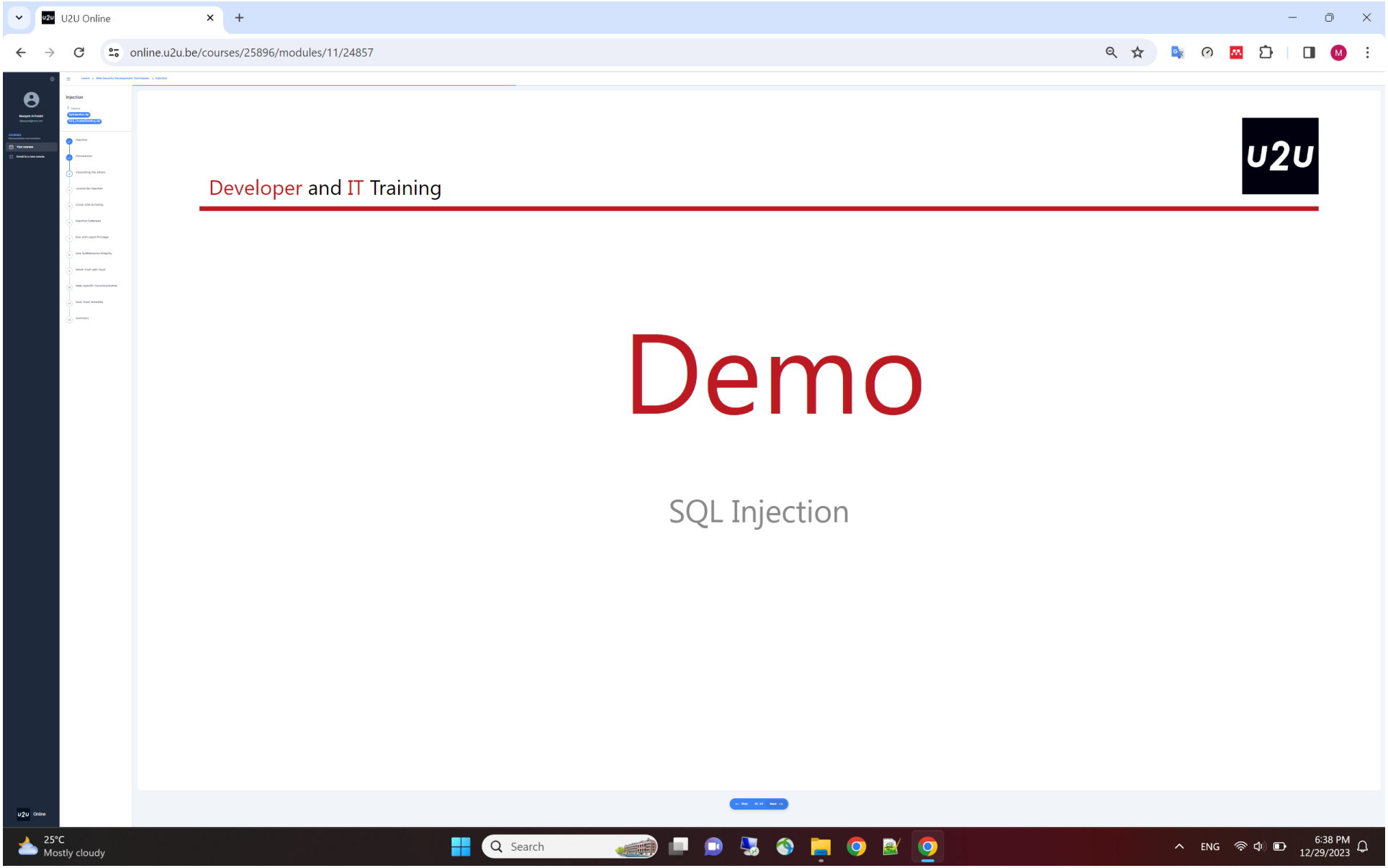

U2U Online
online.u2u.be/courses/25896/modules/11/24857
SQL Injection
u2u
That statement will look up a user with a certain username, combined with a certain password but also when 1 equals 1, which is always...
In other words, we'll get a user back in any case
Because this user object from the database is then stored as credentials for our malicious client, we'll be logged in as that user
Welcome Diedrik
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

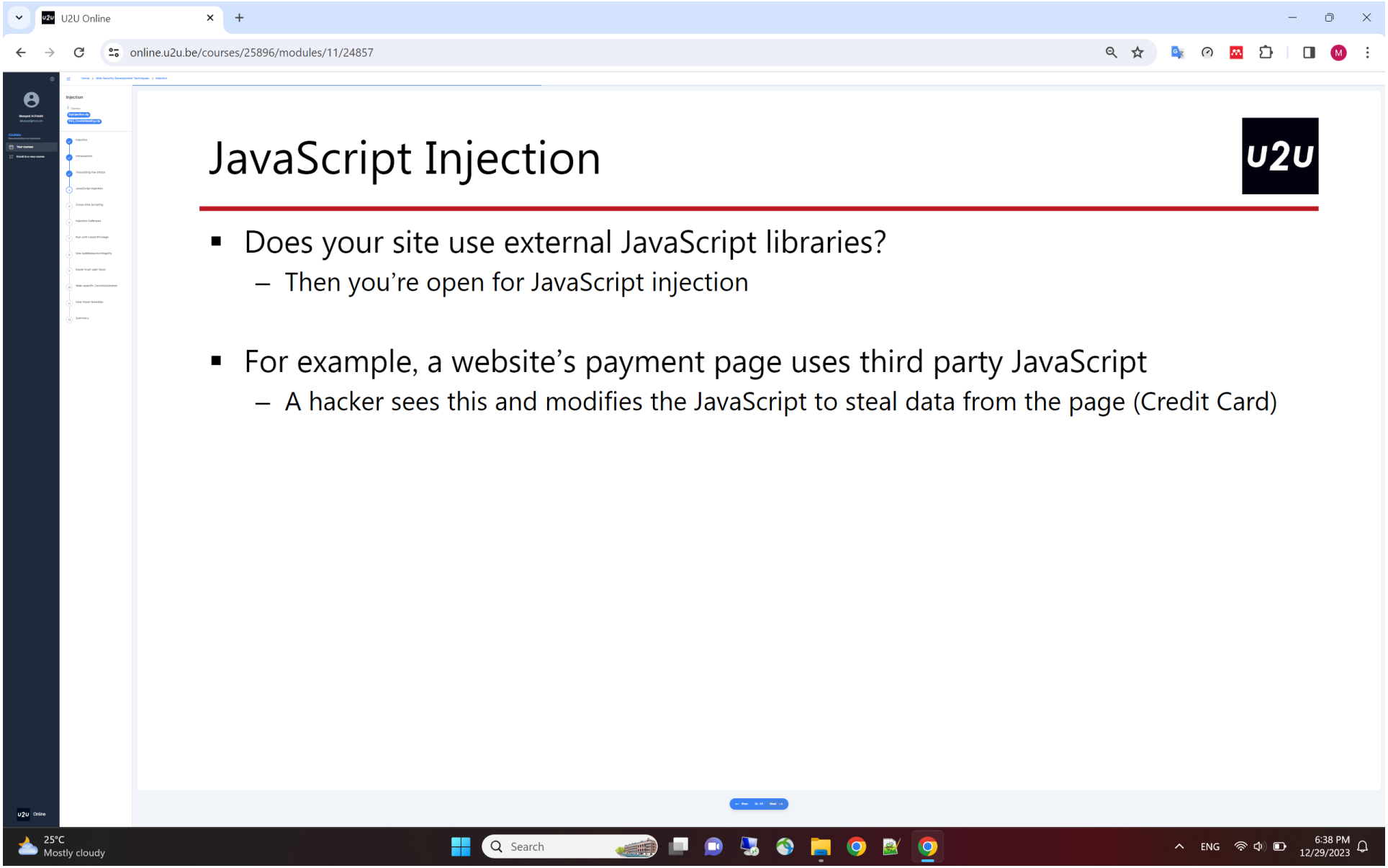

U2U Online
online.u2u.be/courses/25896/modules/11/24857
SQL Injection
u2u
Now our current malicious user is still pretty friendly
He could decide to play around with the website a bit
Other things a user could execute
DROP database table
INSERT new information
...
Login here:
Lieven
'; DROP TABLE Users; --
Log in
Login here:
Lieven
'; INSERT INTO [4U2].[Users]([UserName],[Description],[UserRole],[Password]) VALUES ('Mallory','Super Admin',1 ,'1337 H4X0R') --
Log in
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

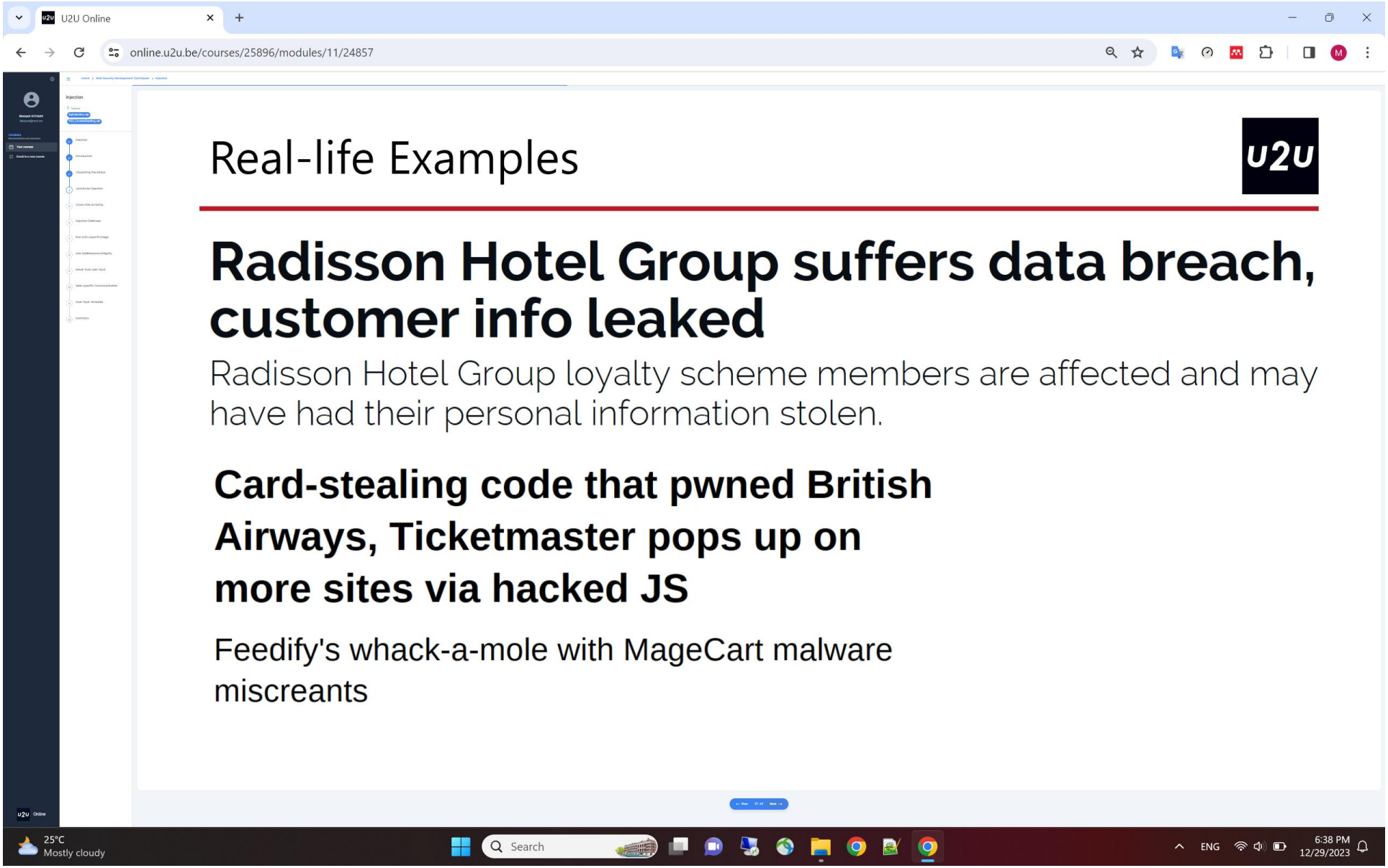

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Sources of untrusted data
u2u
From the user
In the URL, either a query string, or from the route
POSTed from the form
From the browser
In cookies
From a header
From other locations
From an external service
From your own database!
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

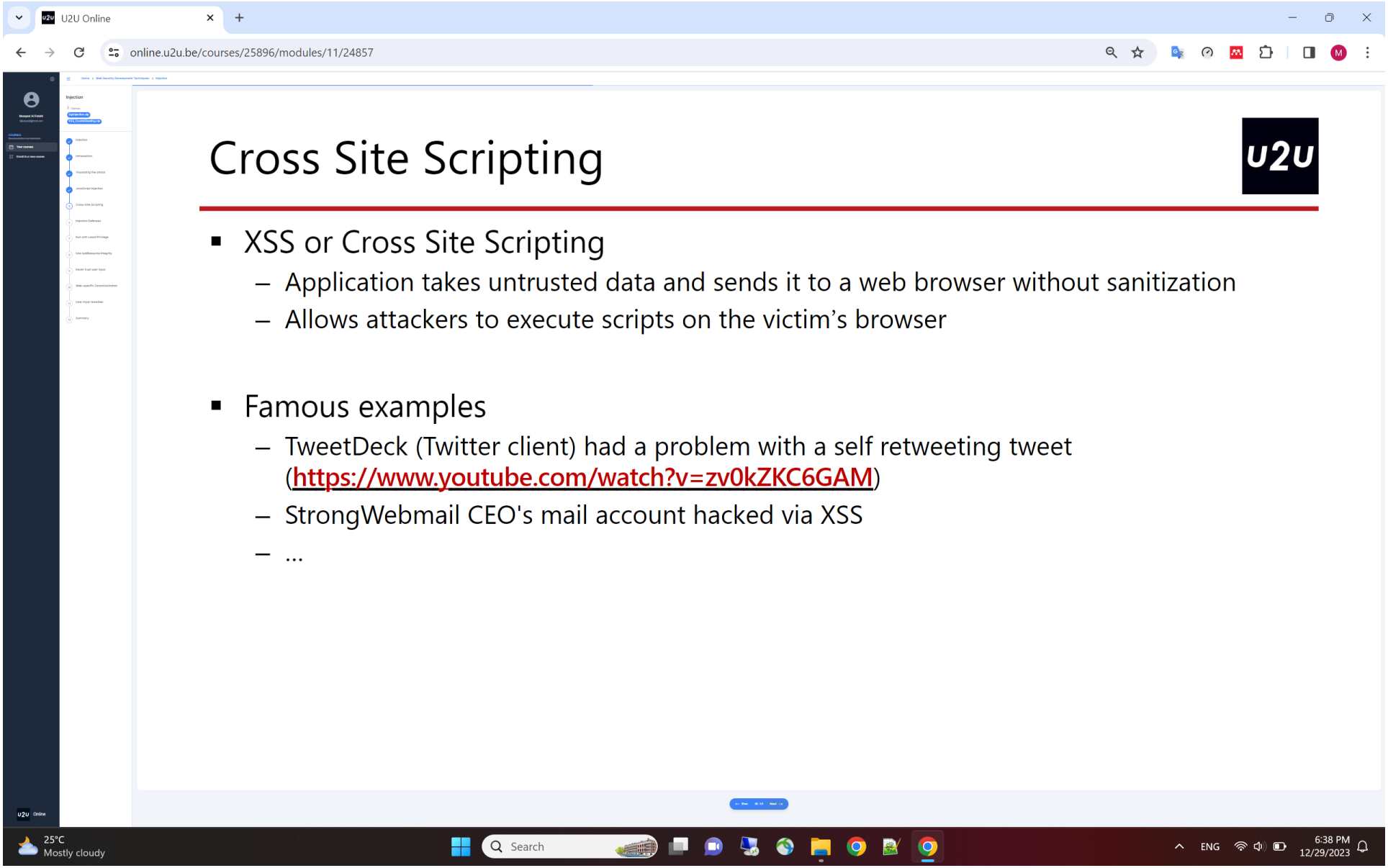

U2U Online
online.u2u.be/courses/25896/modules/11/24857
ZU 0666',0,0); DROP DATABASE TABLICE;--
u2u
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

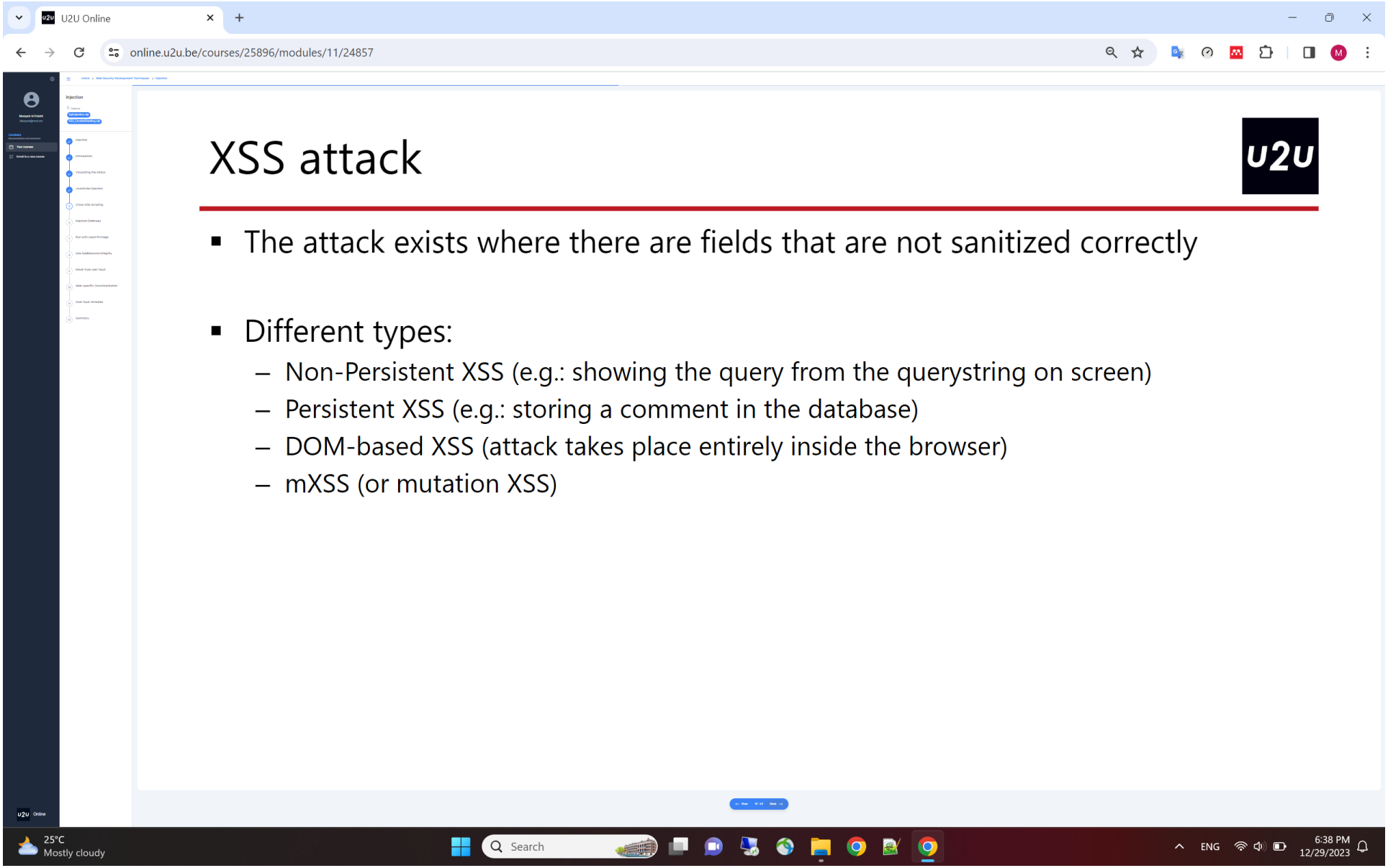

U2U Online
online.u2u.be/courses/25896/modules/11/24857
SQL Injection example from the real world
u2u
TIME
SONY
New Sony Hack Claims Over a Million User Passwords
By Doug Aamoth | June 02, 2011
Share
Like 2
Tweet
G+1 0
Share 321
Pin it
Read Later
Another of Sony's websites has reportedly been hacked—this time around, the victim is SonyPictures.com. The group claiming responsibility for the breach, "LulzSec," is the same group behind the recent PBS website hack.
A statement from the group reads, in part:
Email
Print
Share
Follow @techland
"SonyPictures.com was owned by a very simple SQL injection, one of the most primitive and common vulnerabilities, as we should all know by now. From a single injection, we accessed EVERYTHING. Why do you put such faith in a company that allows itself to become open to these simple attacks? What's worse is that every bit of data we took wasn't encrypted. Sony stored over 1,000,000 passwords of its customers in plaintext, which means it's just a matter of taking it. This is disgraceful and insecure: they were asking for it."
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

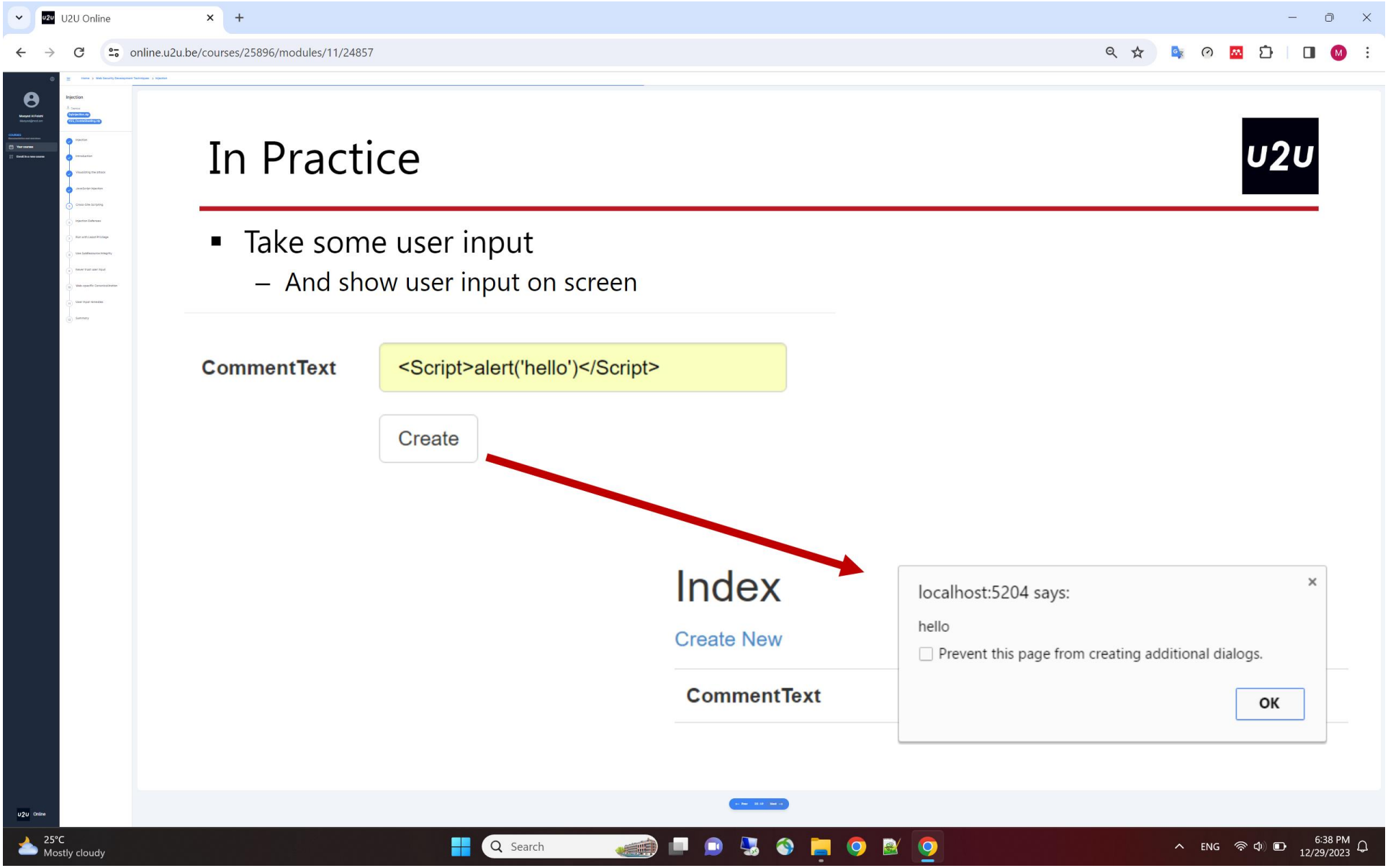

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Developer and IT Training
u2u
Demo
SQL Injection
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

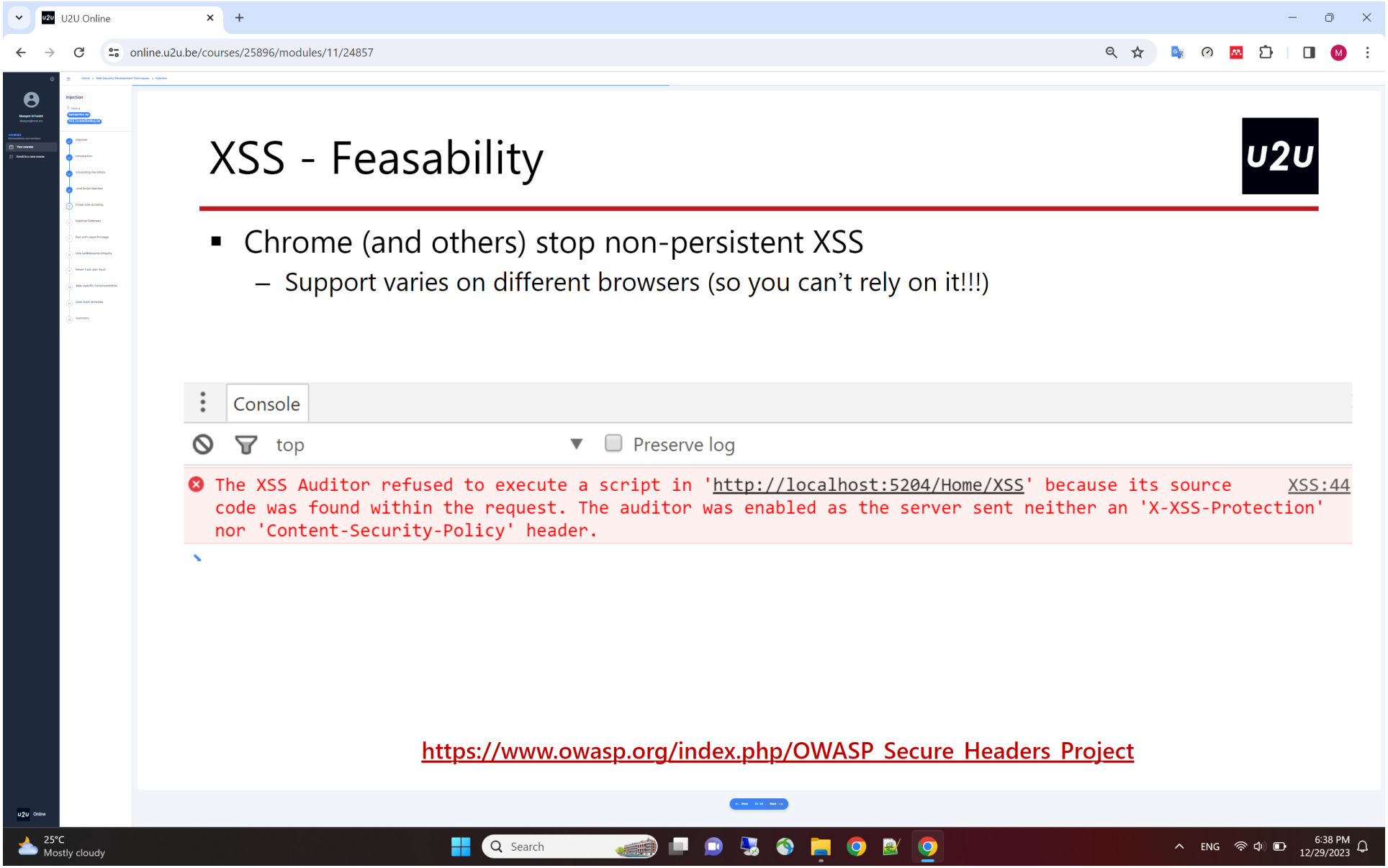

U2U Online
online.u2u.be/courses/25896/modules/11/24857
JavaScript Injection
u2u
Does your site use external JavaScript libraries?
Then you're open for JavaScript injection
For example, a website's payment page uses third party JavaScript
A hacker sees this and modifies the JavaScript to steal data from the page (Credit Card)
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Real-life Examples
U2U
Radisson Hotel Group suffers data breach, customer info leaked
Radisson Hotel Group loyalty scheme members are affected and may have had their personal information stolen.
Card-stealing code that pwned British Airways, Ticketmaster pops up on more sites via hacked JS
Feedify's whack-a-mole with MageCart malware miscreants
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Cross Site Scripting
u2u
XSS or Cross Site Scripting
Application takes untrusted data and sends it to a web browser without sanitization
Allows attackers to execute scripts on the victim's browser
Famous examples
TweetDeck (Twitter client) had a problem with a self retweeting tweet (https://www.youtube.com/watch?v=zv0kZKC6GAM)
StrongWebmail CEO's mail account hacked via XSS
...
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
XSS attack
u2u
The attack exists where there are fields that are not sanitized correctly
Different types:
Non-Persistent XSS (e.g.: showing the query from the querystring on screen)
Persistent XSS (e.g.: storing a comment in the database)
DOM-based XSS (attack takes place entirely inside the browser)
mXSS (or mutation XSS)
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
In Practice
u2u
Take some user input
And show user input on screen
CommentText
<Script>alert('hello')</Script>
Create
Index
Create New
CommentText
localhost:5204 says:
hello
Prevent this page from creating additional dialogs.
OK
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
XSS - Feasability
u2u
Chrome (and others) stop non-persistent XSS
Support varies on different browsers (so you can't rely on it!!!)
Console
top
Preserve log
The XSS Auditor refused to execute a script in 'http://localhost:5204/Home/XSS' because its source code was found within the request. The auditor was enabled as the server sent neither an 'X-XSS-Protection' nor 'Content-Security-Policy' header.
XSS:44
https://www.owasp.org/index.php/OWASP_Secure_Headers_Project
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

XSS - Feasability

- XSS could work
  - On a website that needs to take in raw HTML like an e-mail host or CMS system
  - On a website that needs to show HTML coming from the user like a rich comment section

U2U Online

online.u2u.be/courses/25896/modules/11/24857

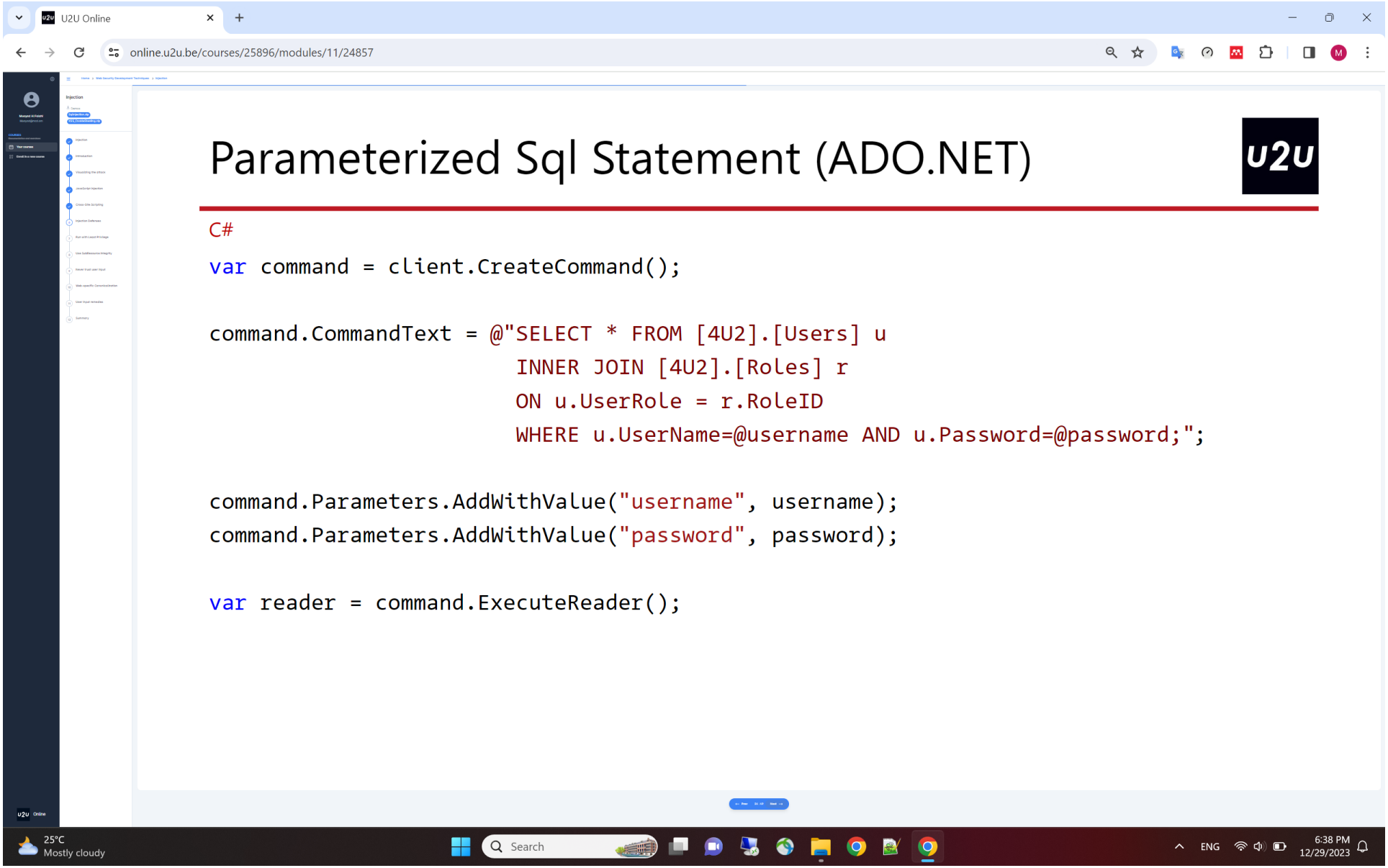
SQL Injection - Defenses

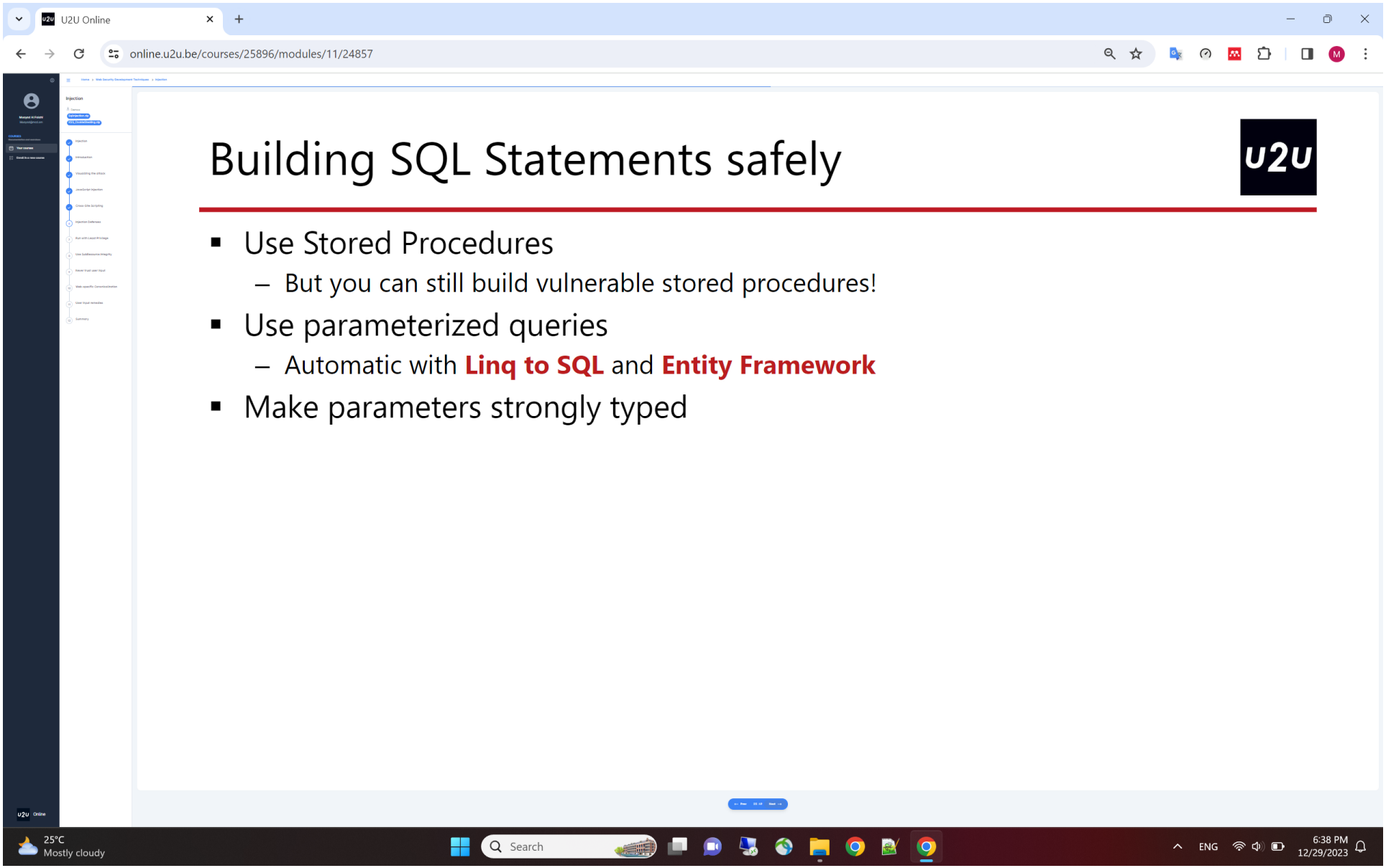
- Validate the input coming in and sanitize it
  - Whitelist data
- Escape the characters that are being used for SQL Injection attacks
  - e.g.: ‘ (single quote)
  - Most languages have an escape method
- Use parameterized SQL Statements - .NET supports this OOB
  - ADO.NET
  - Entity Framework
- Change Database permissions (run with minimal permissions)

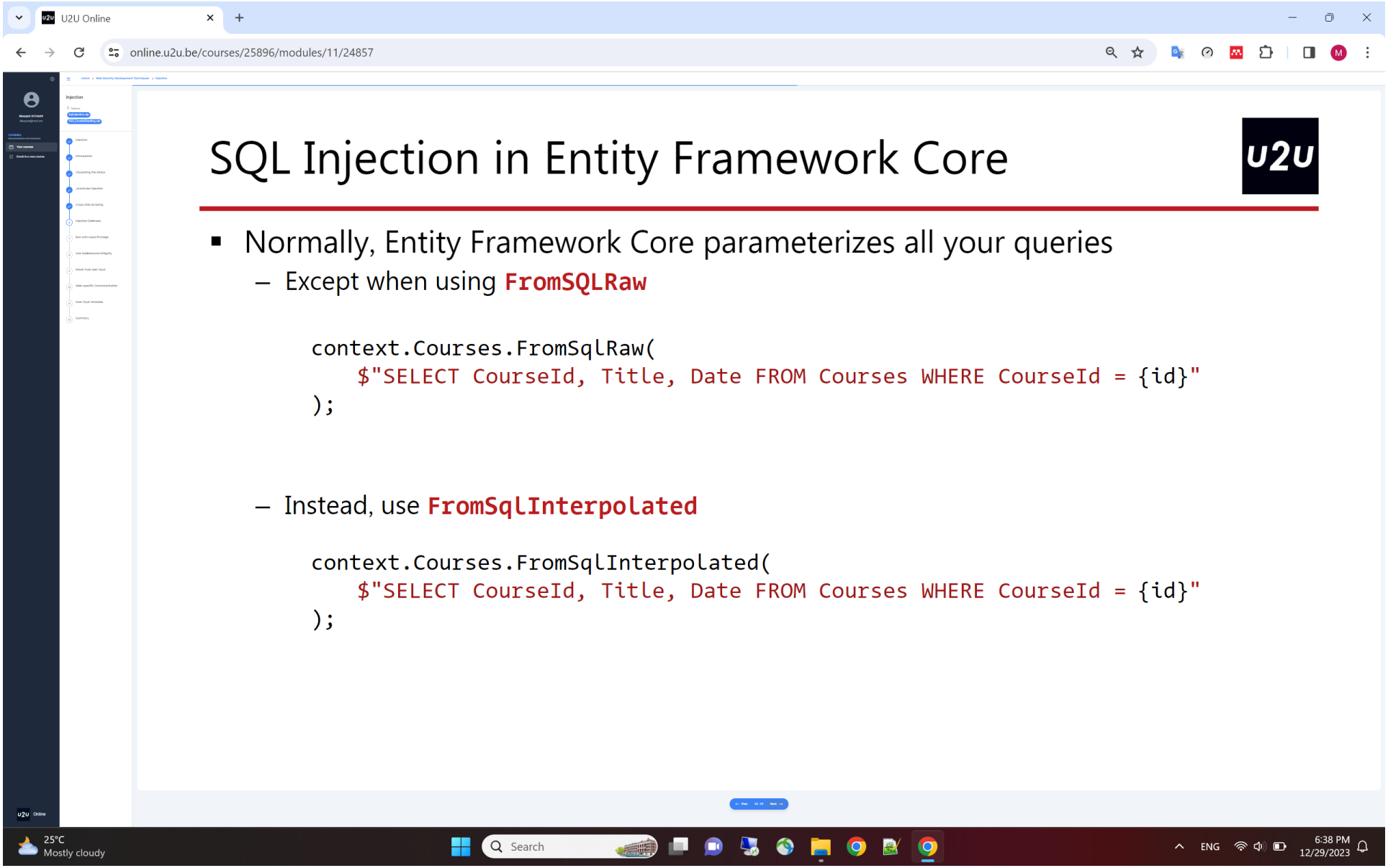
25°C Mostly cloudy

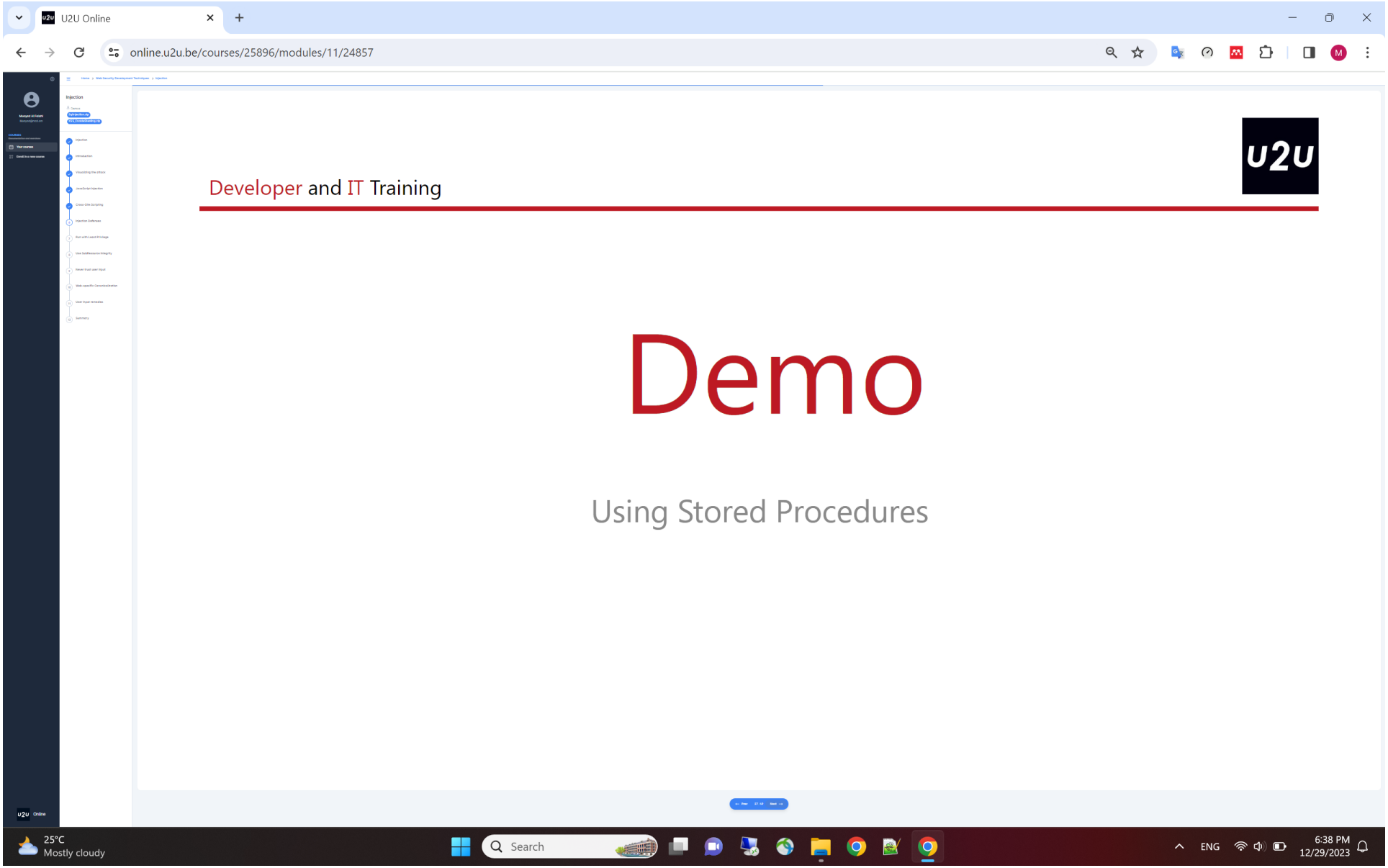
Search

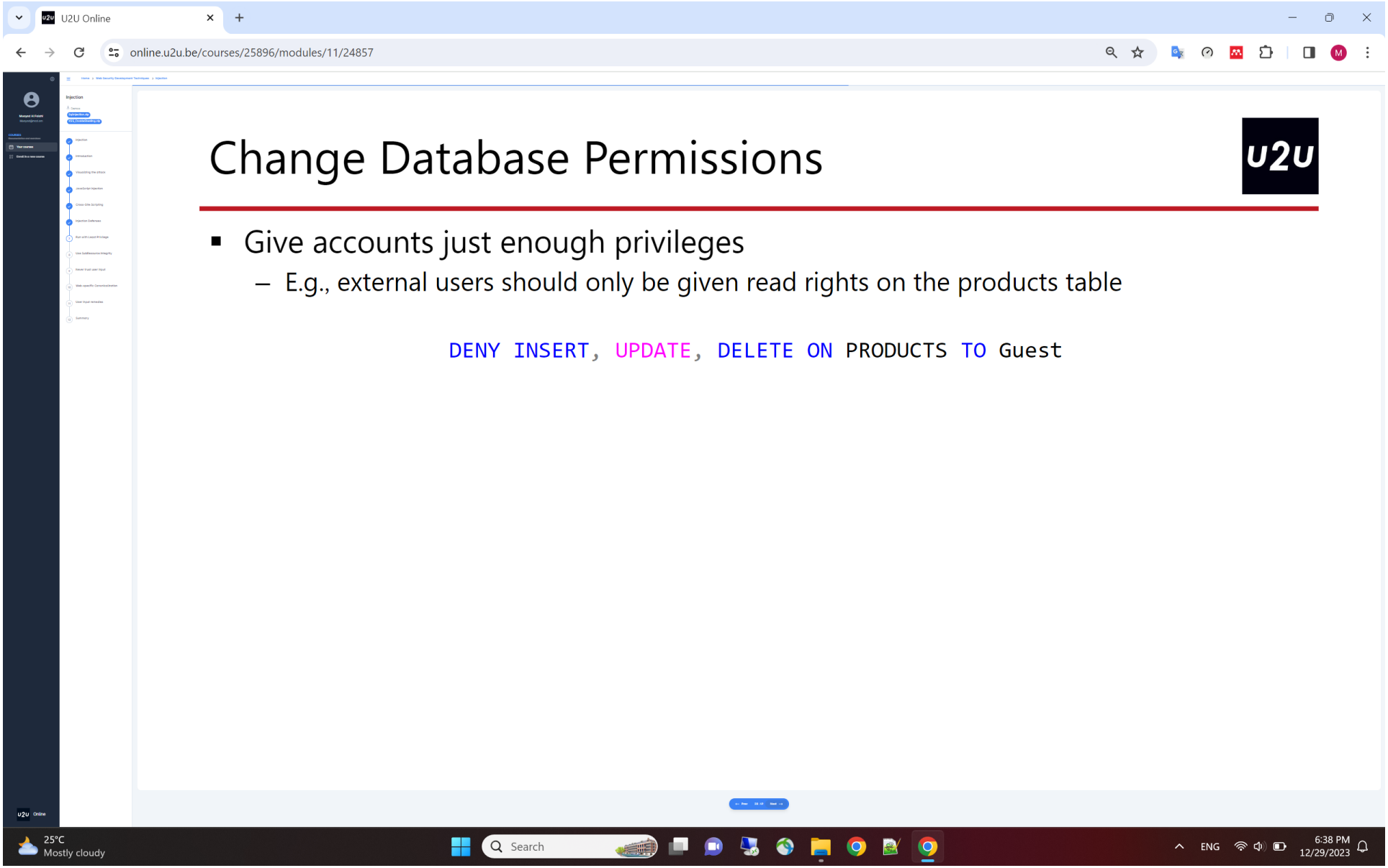
ENG 6:38 PM 12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Parameterized Sql Statement (ADO.NET)
u2u
C#
var command = client.CreateCommand();
command.CommandText = @"SELECT * FROM [4U2].[Users] u
INNER JOIN [4U2].[Roles] r
ON u.UserRole = r.RoleID
WHERE u.UserName=@username AND u.Password=@password;";
command.Parameters.AddWithValue("username", username);
command.Parameters.AddWithValue("password", password);
var reader = command.ExecuteReader();
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

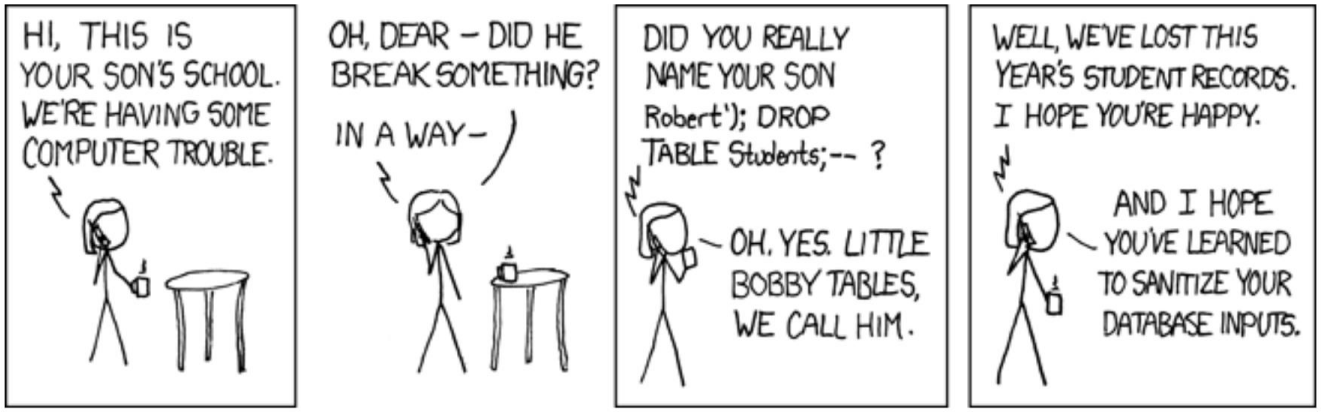

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Building SQL Statements safely
U2U
Use Stored Procedures
But you can still build vulnerable stored procedures!
Use parameterized queries
Automatic with Linq to SQL and Entity Framework
Make parameters strongly typed
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

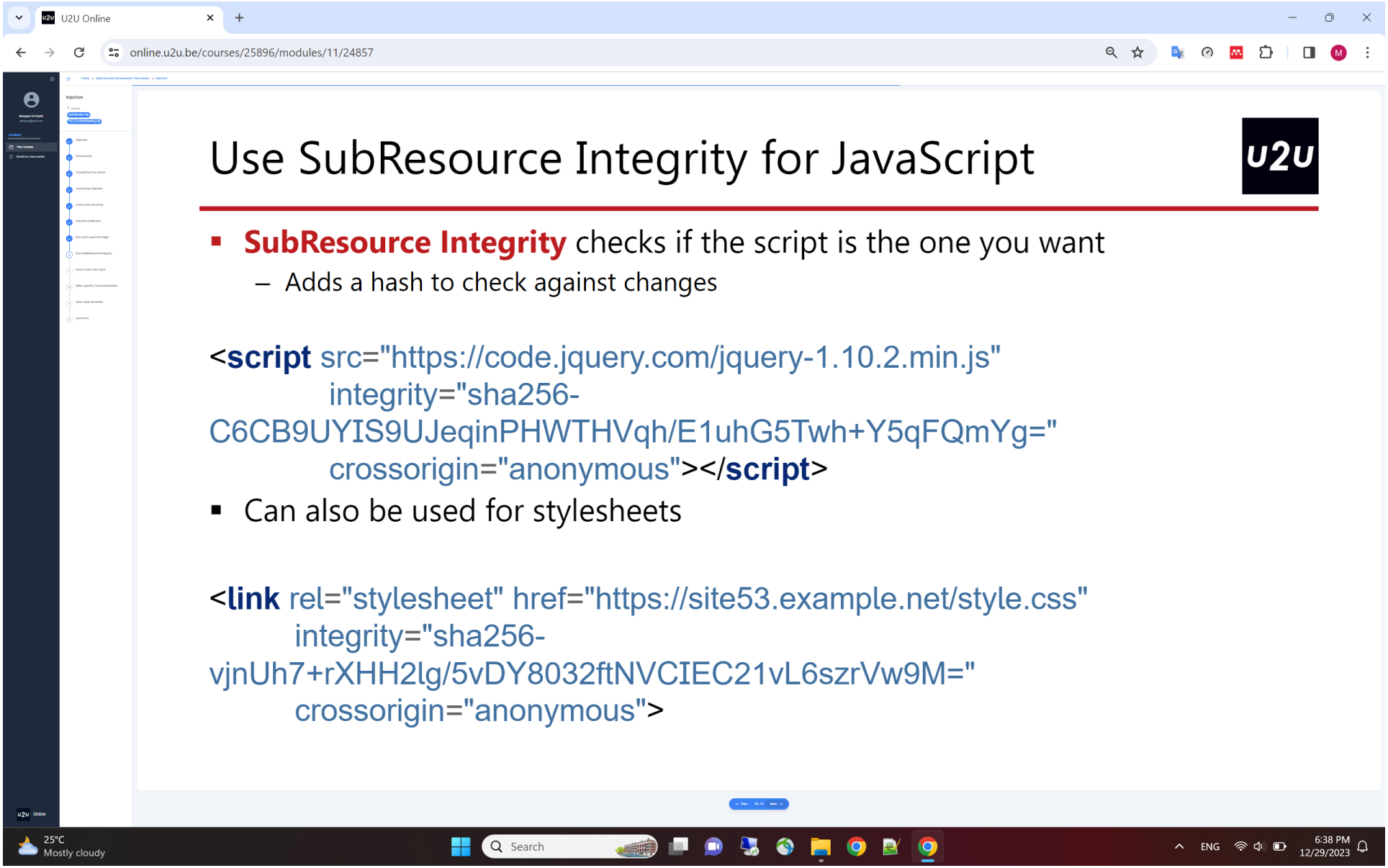

U2U Online
online.u2u.be/courses/25896/modules/11/24857
SQL Injection in Entity Framework Core
U2U
Normally, Entity Framework Core parameterizes all your queries
Except when using FromSQLRaw
context.Courses.FromSqlRaw(
$"SELECT CourseId, Title, Date FROM Courses WHERE CourseId = {id}"
);
Instead, use FromSqlInterpolated
context.Courses.FromSqlInterpolated(
$"SELECT CourseId, Title, Date FROM Courses WHERE CourseId = {id}"
);
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Developer and IT Training
u2u
Demo
Using Stored Procedures
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Change Database Permissions
u2u
Give accounts just enough privileges
E.g., external users should only be given read rights on the products table
DENY INSERT, UPDATE, DELETE ON PRODUCTS TO Guest
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

Bobby Tables – Credits to xkcd




U2U Online
online.u2u.be/courses/25896/modules/11/24857
Use SubResource Integrity for JavaScript
U2U
SubResource Integrity checks if the script is the one you want
Adds a hash to check against changes
<script src="https://code.jquery.com/jquery-1.10.2.min.js"
integrity="sha256-
C6CB9UYIS9UJeqinPHWTHVqh/E1uhG5Twh+Y5qFQmYg="
crossorigin="anonymous"></script>
Can also be used for stylesheets
<link rel="stylesheet" href="https://site53.example.net/style.css"
integrity="sha256-
vjnUh7+rXHH2lg/5vDY8032ftNVCIEC21vL6szrVw9M="
crossorigin="anonymous">
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

U2U Online

online.u2u.be/courses/25896/modules/11/24857

Example of using SRI

- If you're using a CDN, most provide an SRI link for you
- For example: https://code.jquery.com/

Code Integration

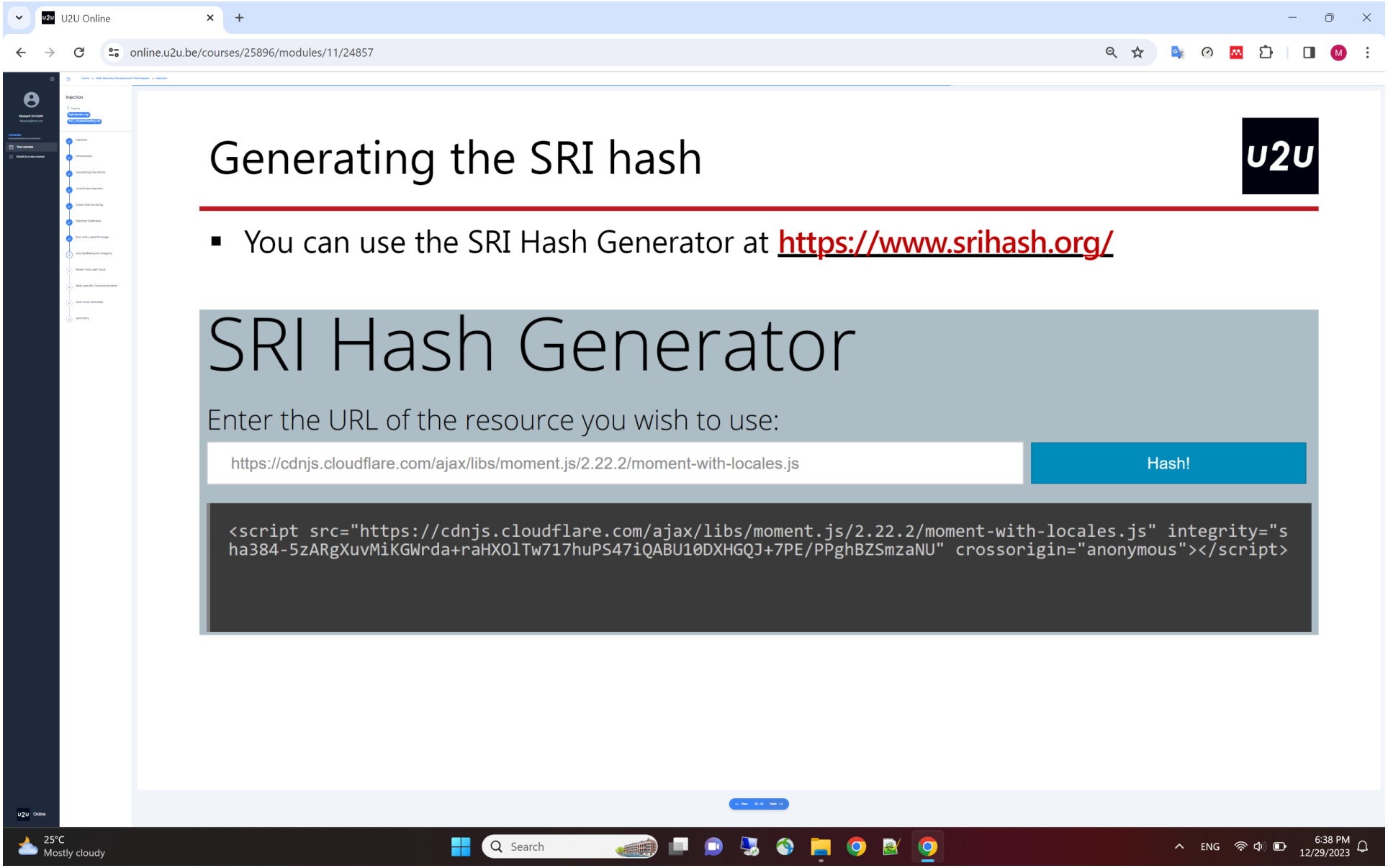
```
<script
  src="https://code.jquery.com/jquery-3.3.1.min.js"
  integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
  crossorigin="anonymous"></script>
```

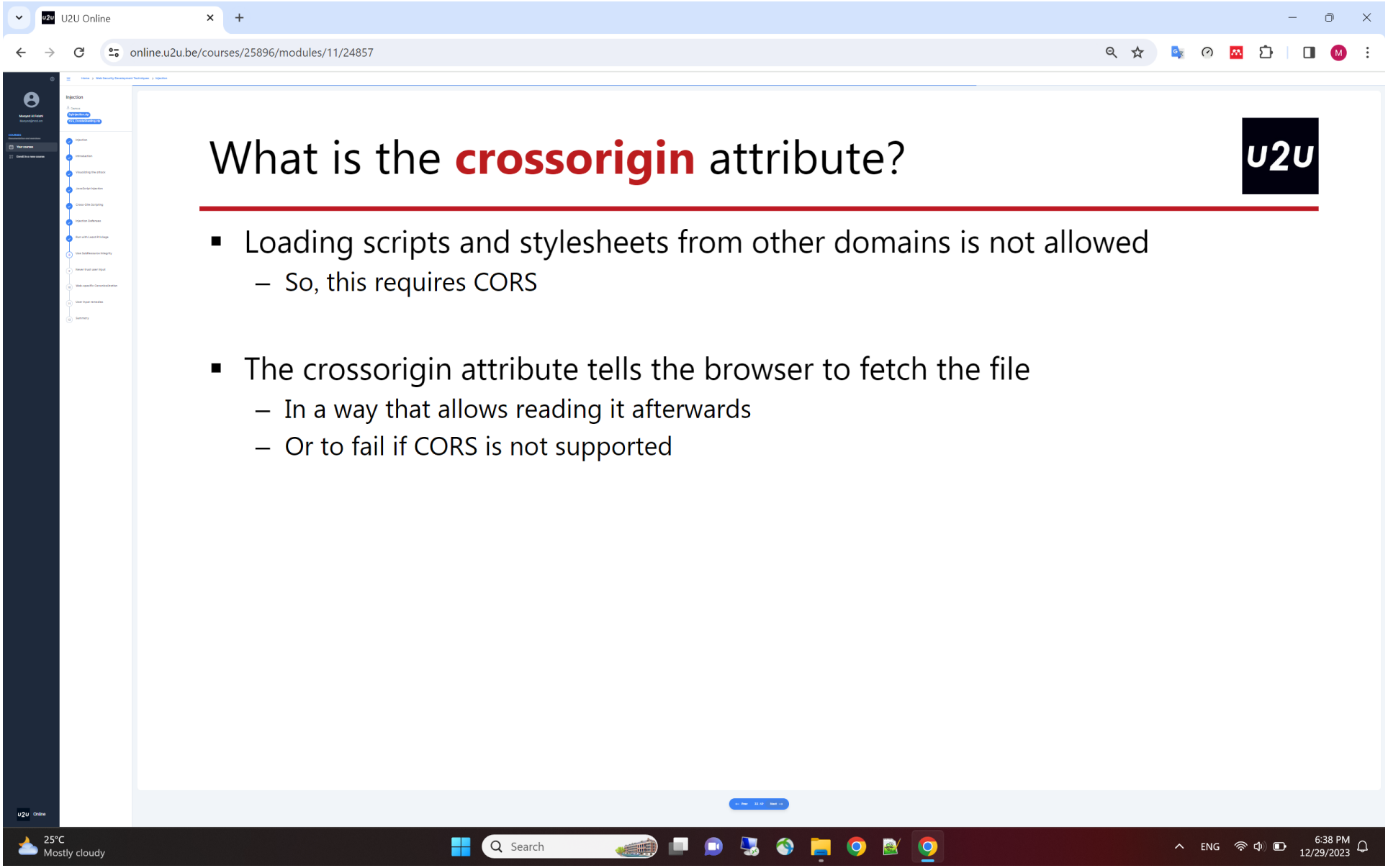
The `integrity` and `crossorigin` attributes are used for Subresource Integrity (SRI) checking. This allows browsers to ensure that resources hosted on third-party servers have not been tampered with. Use of SRI is recommended as a best-practice, whenever libraries are loaded from a third-party source. Read more at srihash.org

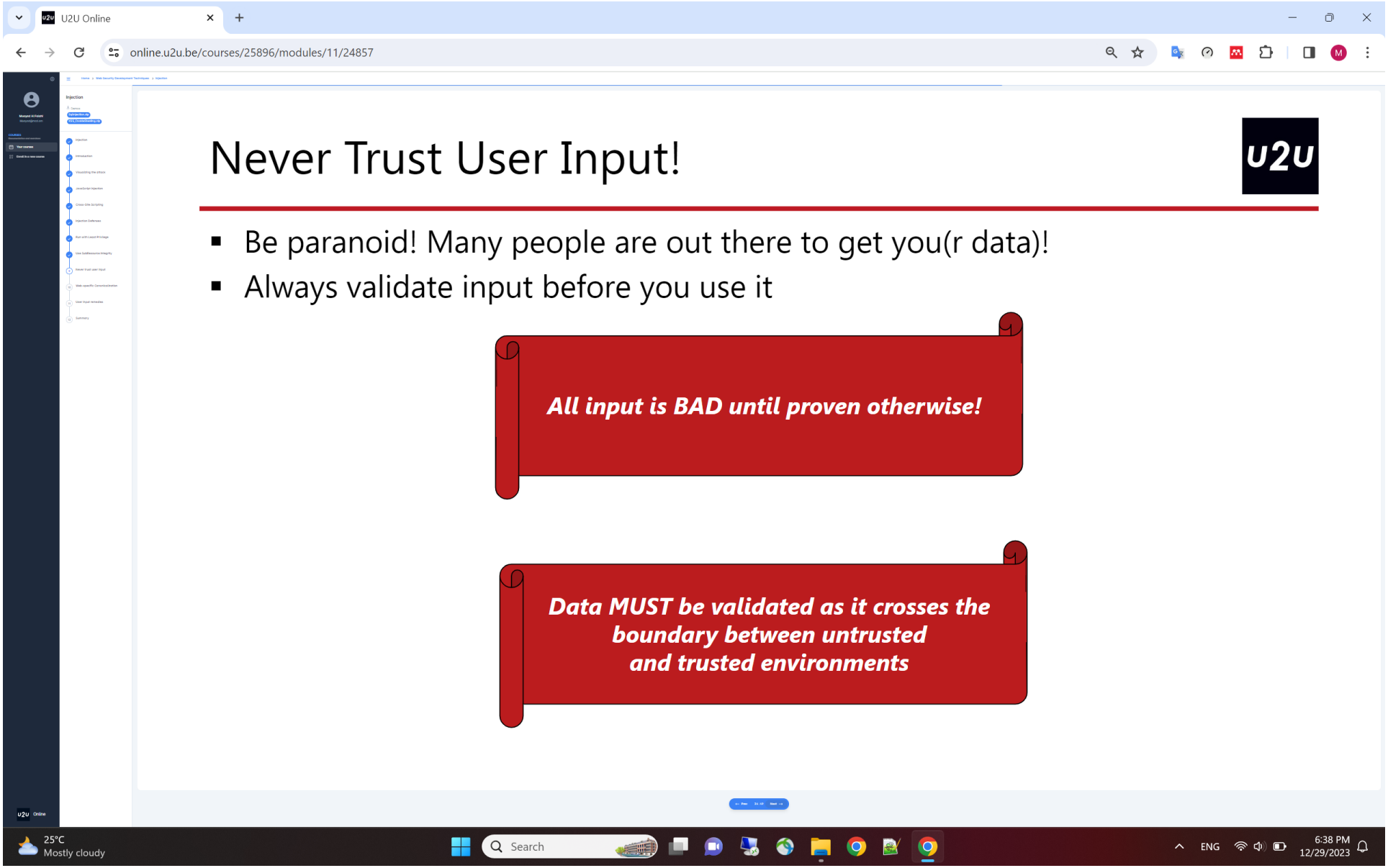
25°C Mostly cloudy

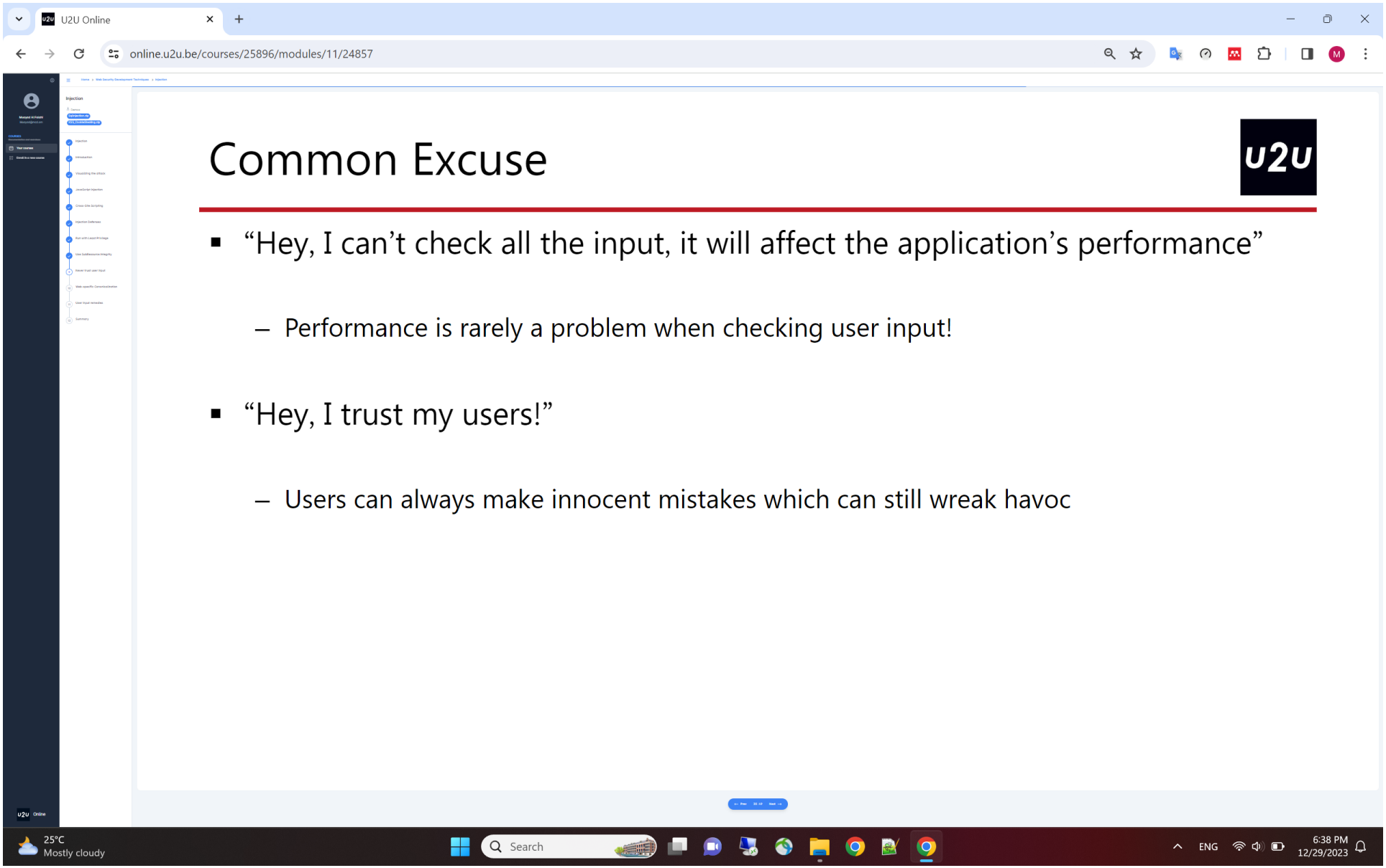
Search

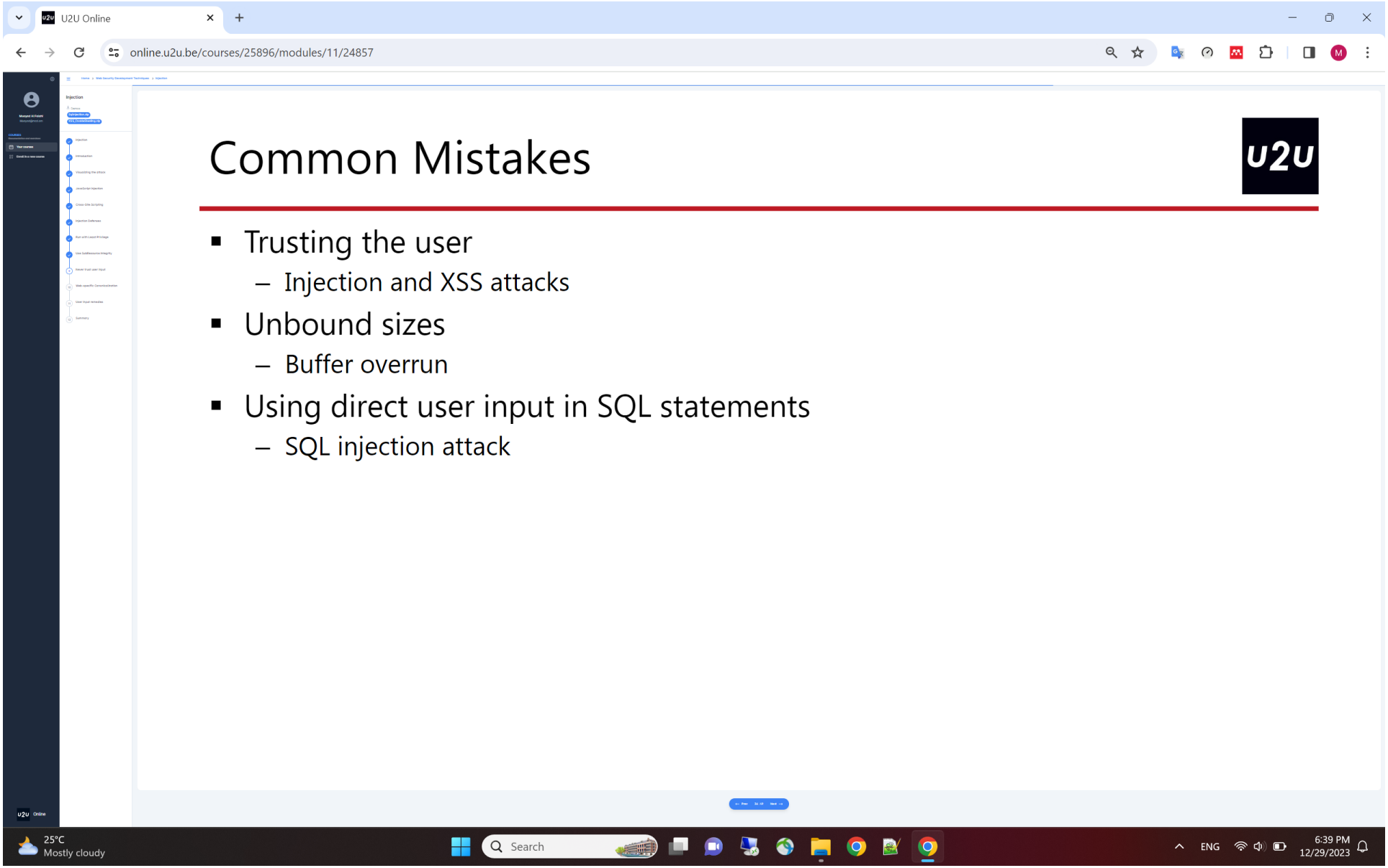
ENG 6:38 PM 12/29/2023

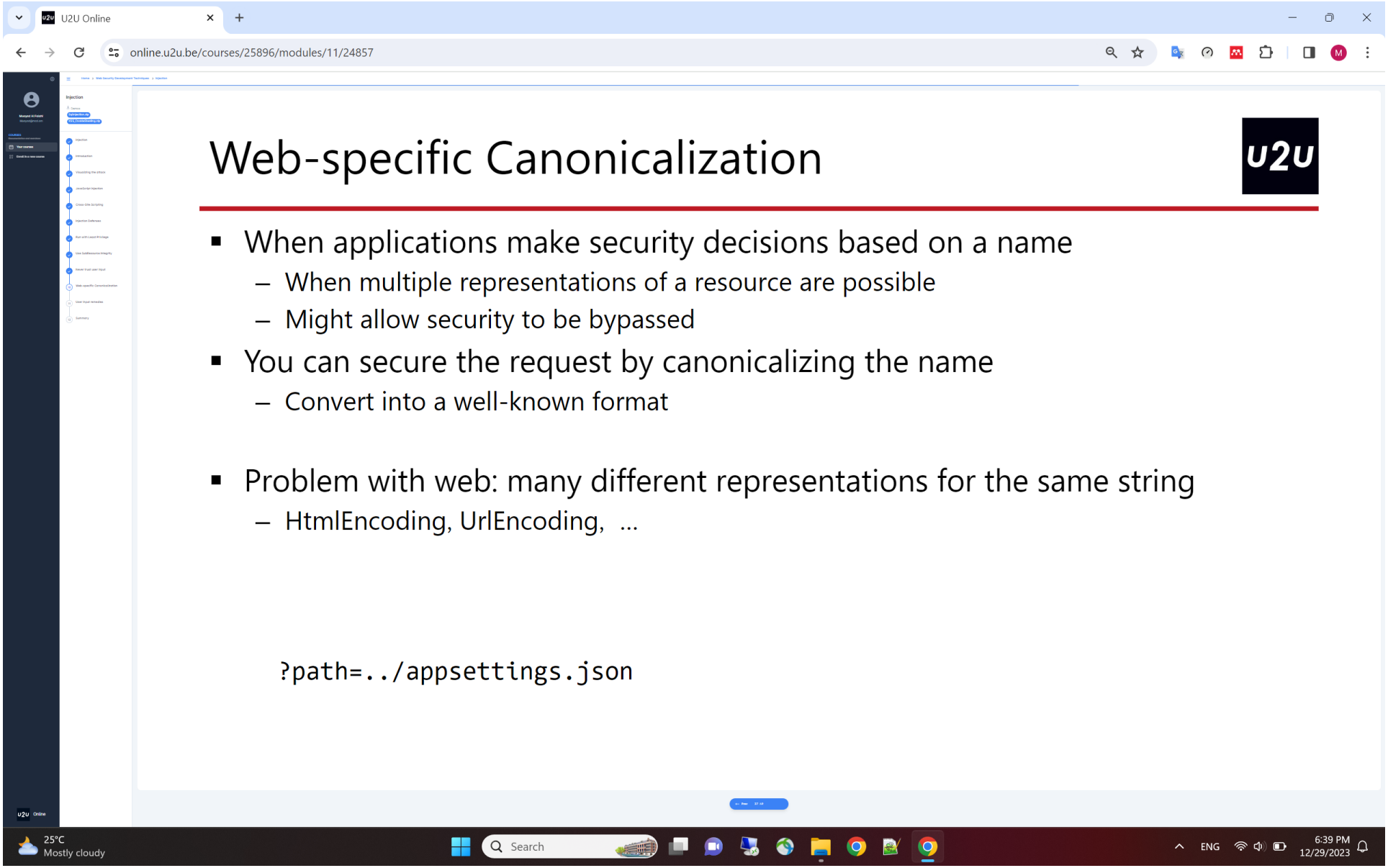

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Generating the SRI hash
U2U
You can use the SRI Hash Generator at https://www.srihash.org/
SRI Hash Generator
Enter the URL of the resource you wish to use:
https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment-with-locales.js
Hash!
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment-with-locales.js" integrity="sha384-5zARgXuvMiKGWrda+raHXOlTw717huPS47iQABU10DXHGQJ+7PE/PPghBZSmzaNU" crossorigin="anonymous"></script>
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

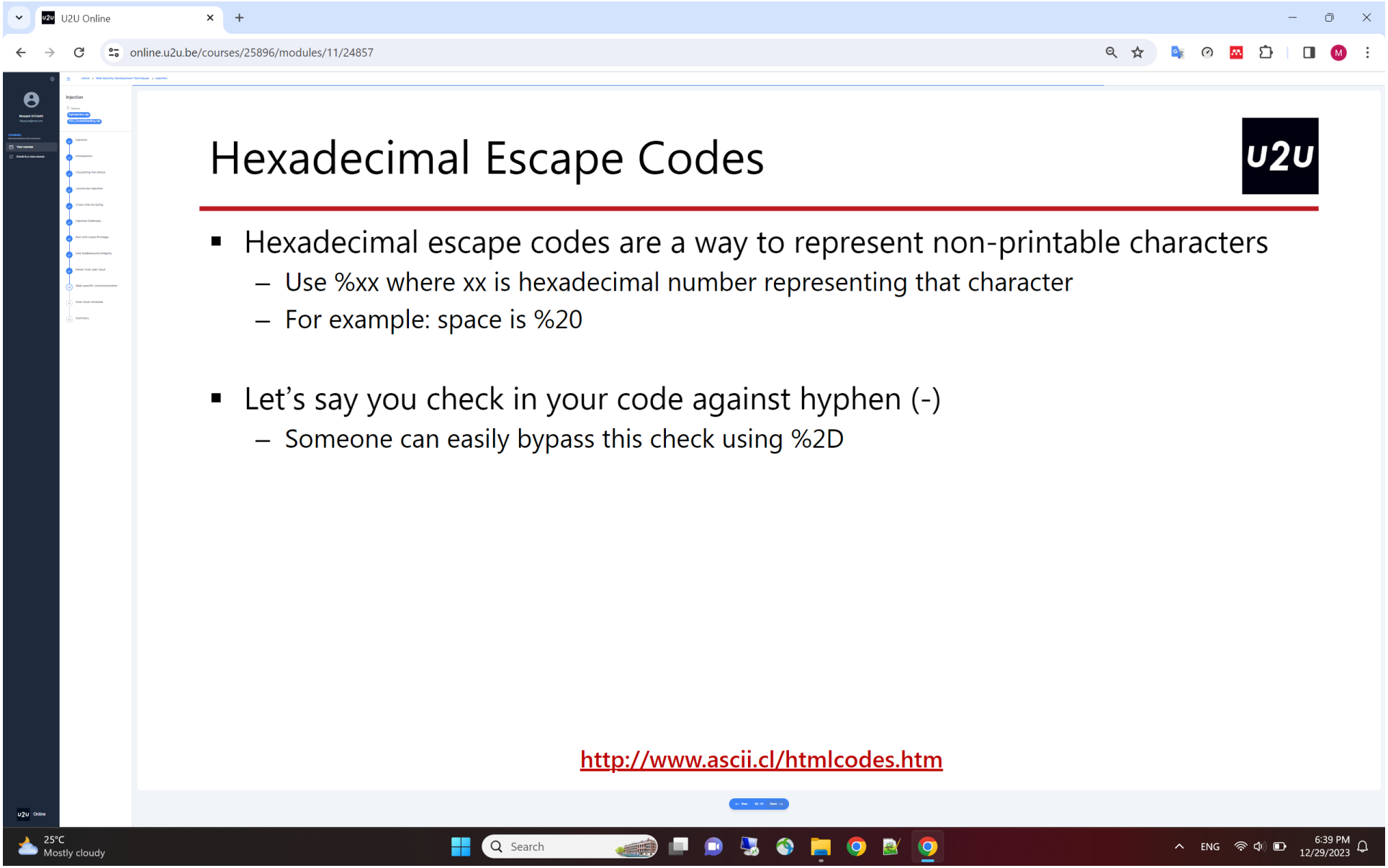

U2U Online
online.u2u.be/courses/25896/modules/11/24857
What is the crossorigin attribute?
U2U
Loading scripts and stylesheets from other domains is not allowed
So, this requires CORS
The crossorigin attribute tells the browser to fetch the file
In a way that allows reading it afterwards
Or to fail if CORS is not supported
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

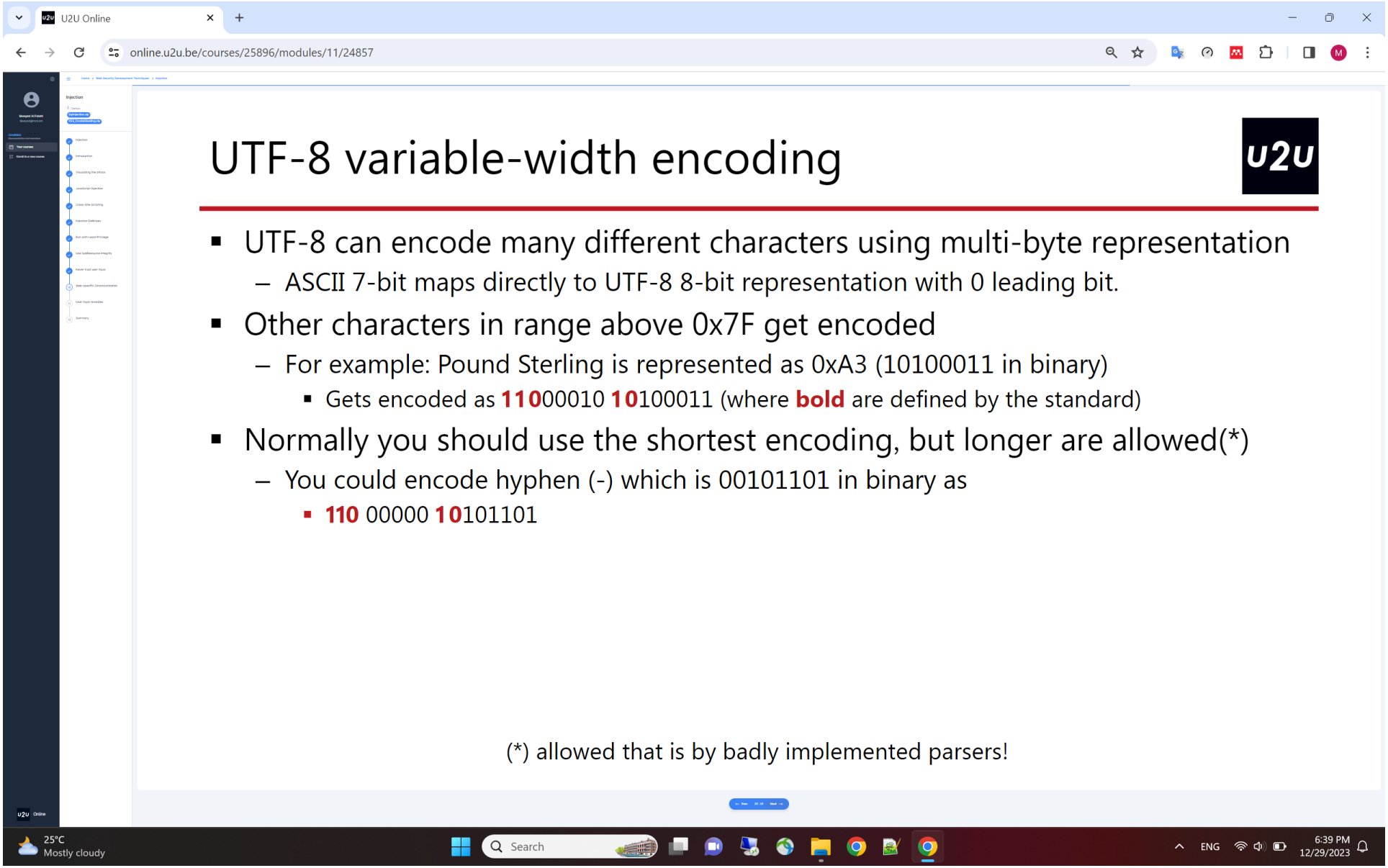

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Never Trust User Input!
u2u
Be paranoid! Many people are out there to get you(r data)!
Always validate input before you use it
All input is BAD until proven otherwise!
Data MUST be validated as it crosses the boundary between untrusted and trusted environments
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023

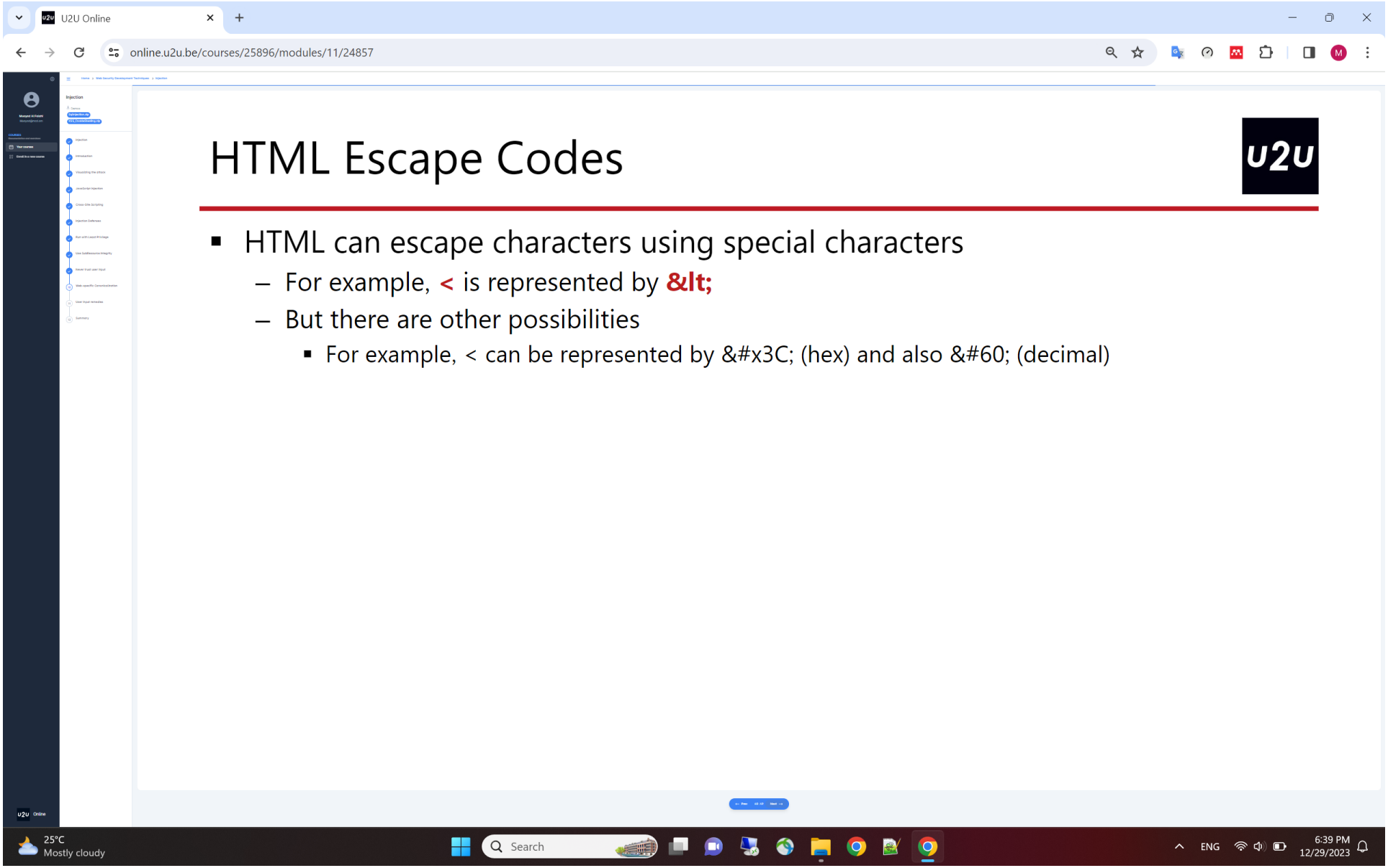

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Common Excuse
u2u
“Hey, I can’t check all the input, it will affect the application’s performance”
Performance is rarely a problem when checking user input!
“Hey, I trust my users!”
Users can always make innocent mistakes which can still wreak havoc
25°C
Mostly cloudy
Search
ENG
6:38 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Common Mistakes
u2u
Trusting the user
Injection and XSS attacks
Unbound sizes
Buffer overrun
Using direct user input in SQL statements
SQL injection attack
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Web-specific Canonicalization
U2U
When applications make security decisions based on a name
When multiple representations of a resource are possible
Might allow security to be bypassed
You can secure the request by canonicalizing the name
Convert into a well-known format
Problem with web: many different representations for the same string
HtmlEncoding, UrlEncoding, ...
?path=../appsettings.json
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023

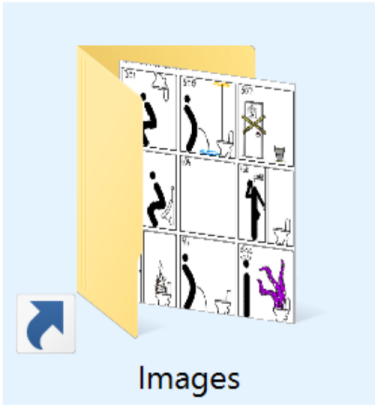

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Hexadecimal Escape Codes
u2u
Hexadecimal escape codes are a way to represent non-printable characters
Use %xx where xx is hexadecimal number representing that character
For example: space is %20
Let's say you check in your code against hyphen (-)
Someone can easily bypass this check using %2D
http://www.ascii.cl/htmlcodes.htm
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
UTF-8 variable-width encoding
U2U
UTF-8 can encode many different characters using multi-byte representation
ASCII 7-bit maps directly to UTF-8 8-bit representation with 0 leading bit.
Other characters in range above 0x7F get encoded
For example: Pound Sterling is represented as 0xA3 (10100011 in binary)
Gets encoded as 110 00010 10 100011 (where bold are defined by the standard)
Normally you should use the shortest encoding, but longer are allowed(*)
You could encode hyphen (-) which is 00101101 in binary as
110 00000 10 101101
(*) allowed that is by badly implemented parsers!
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023

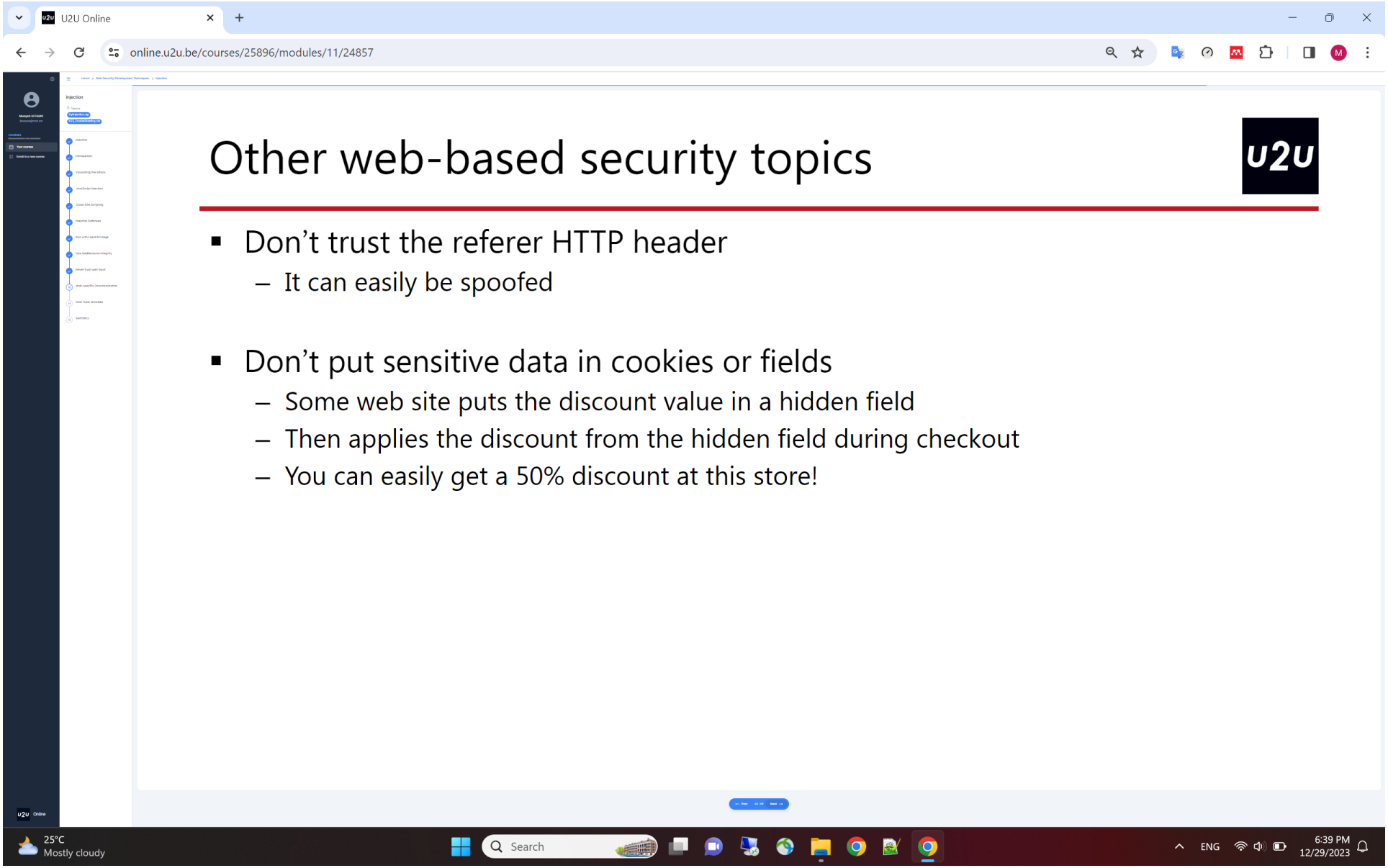

U2U Online
online.u2u.be/courses/25896/modules/11/24857
HTML Escape Codes
u2u
HTML can escape characters using special characters
For example, < is represented by <
But there are other possibilities
For example, < can be represented by < (hex) and also < (decimal)
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023

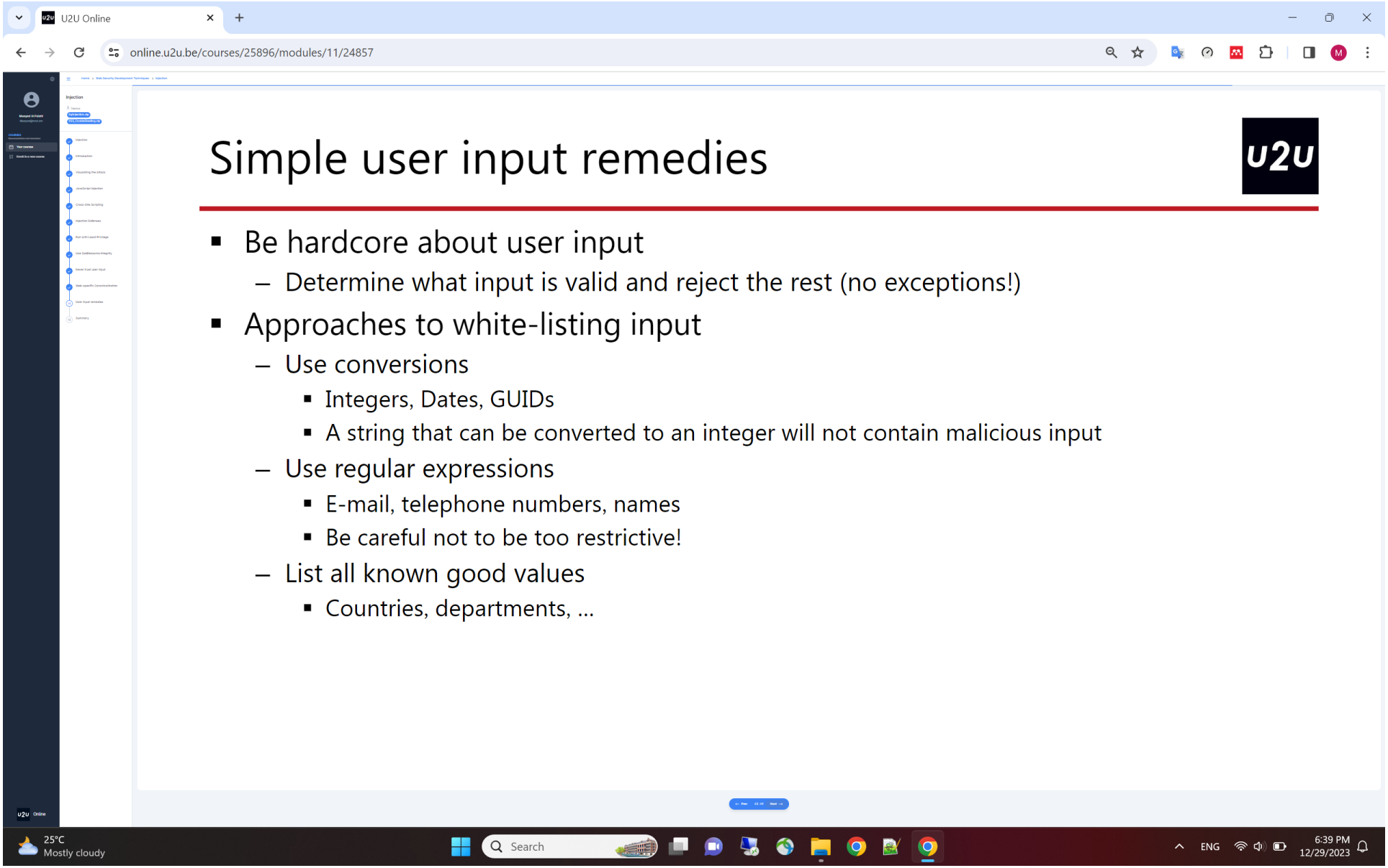
Don’t use parent paths in your application

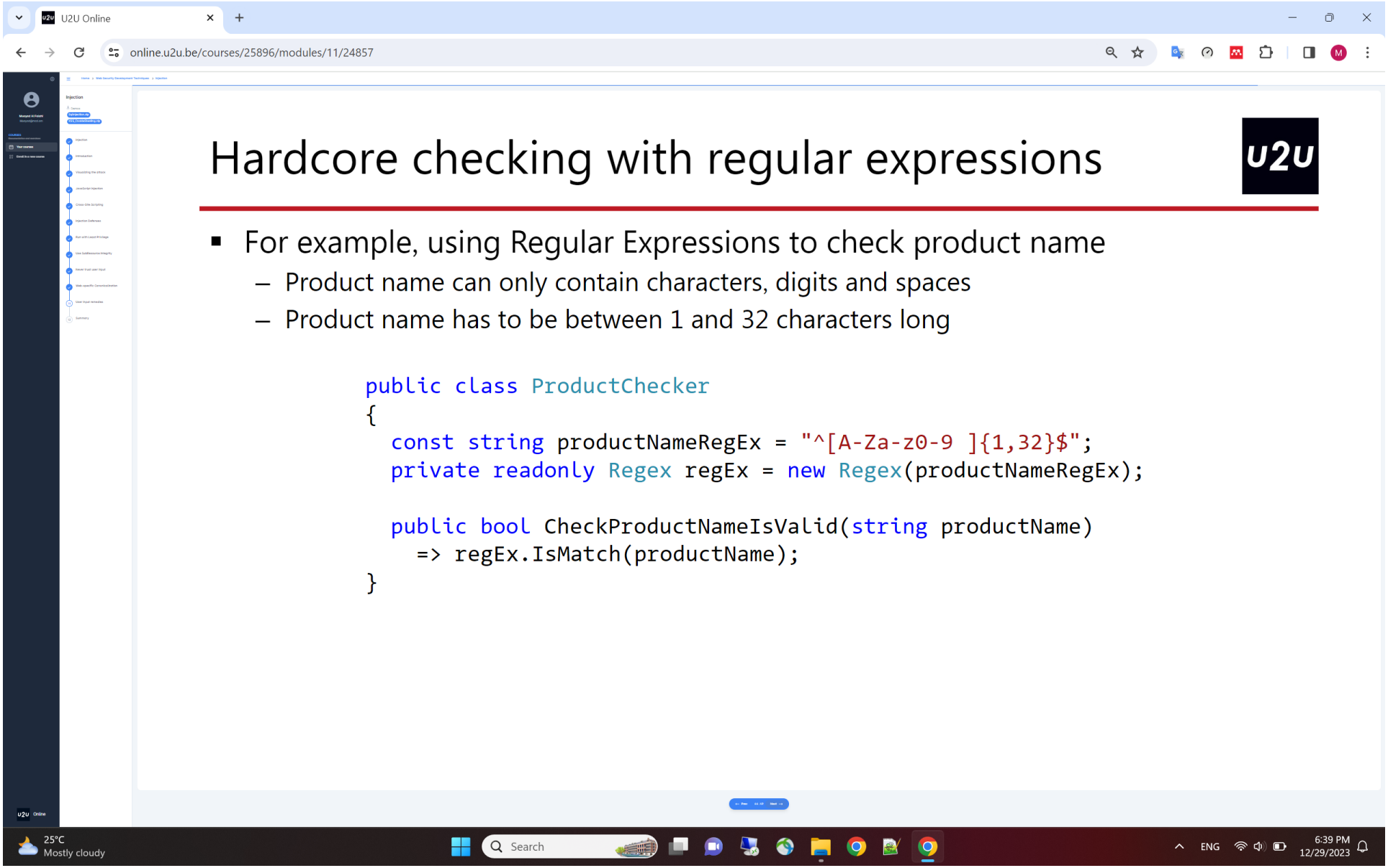
- Using parent paths (..) can lead to directory traversal
  - Avoid `../images/Logo.jpg`
- You can avoid parent paths by using symbolic links
  - Symbolic links are like shortcuts to another file or folder
- To create a symbolic link, use the **MKLINK** tool
  - From a command prompt with Admin privileges
  - For example: to create a link to the ..\images folder:

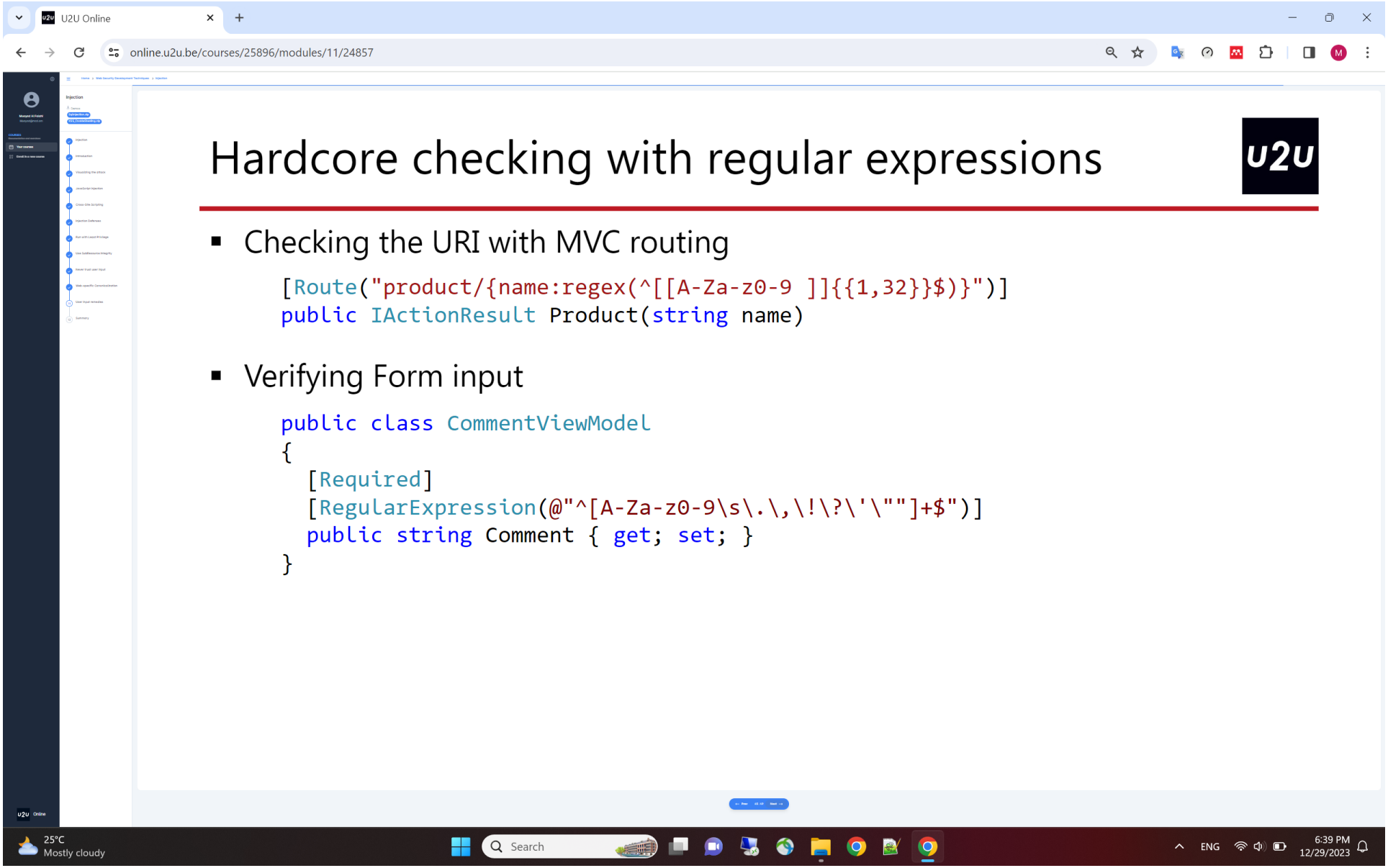


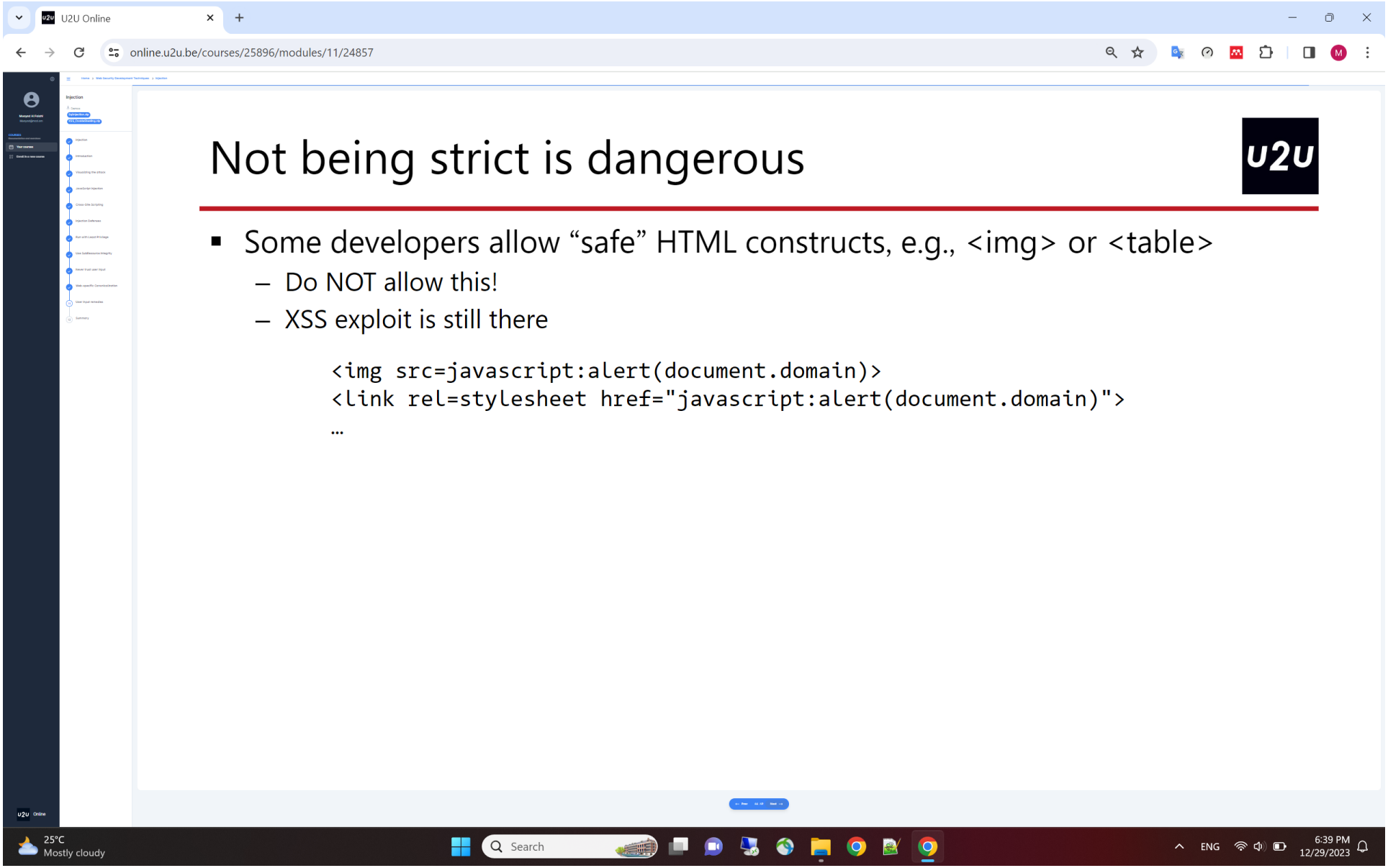
MKLINK /D Images ..\Images

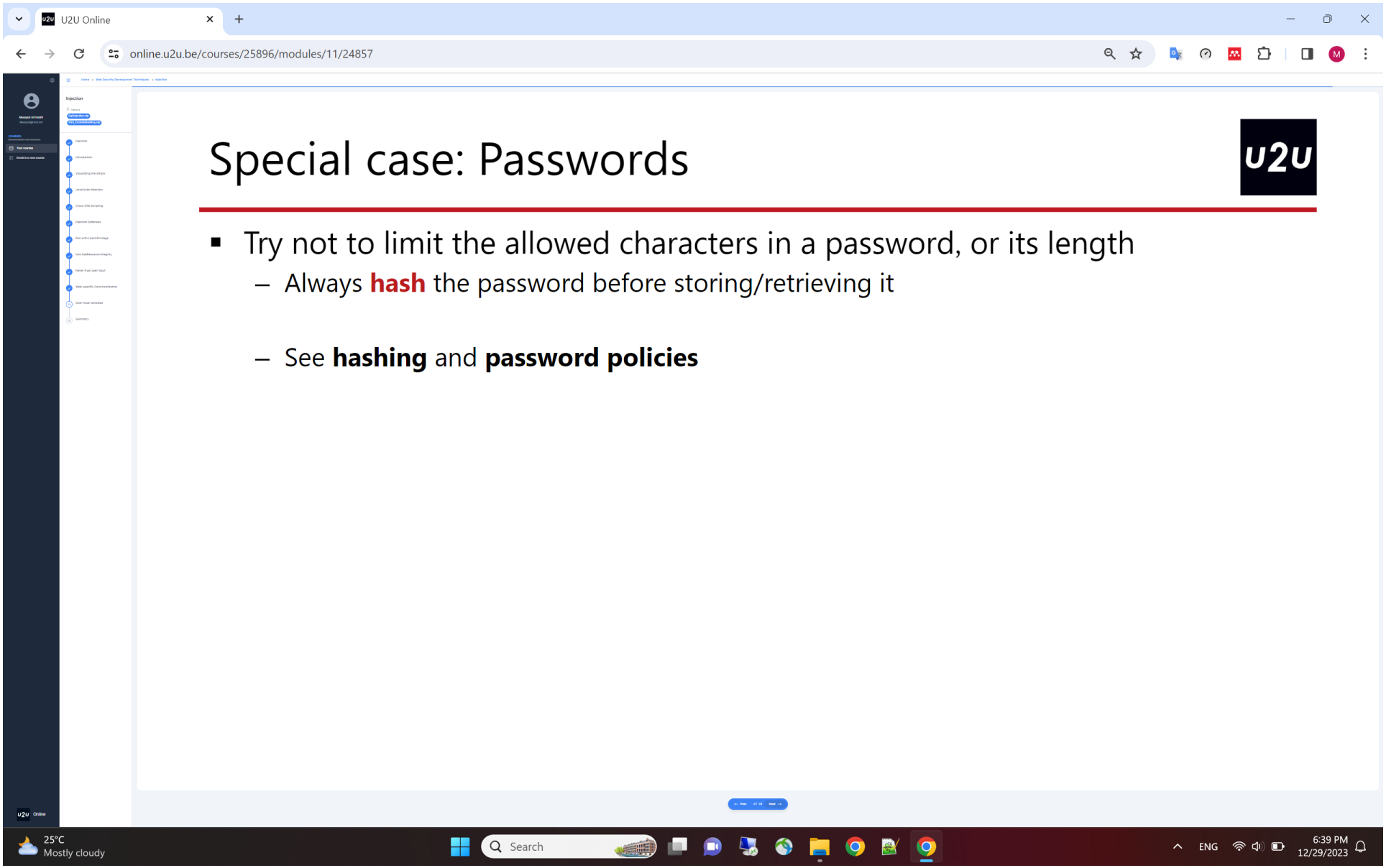

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Other web-based security topics
u2u
Don’t trust the referer HTTP header
It can easily be spoofed
Don’t put sensitive data in cookies or fields
Some web site puts the discount value in a hidden field
Then applies the discount from the hidden field during checkout
You can easily get a 50% discount at this store!
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023

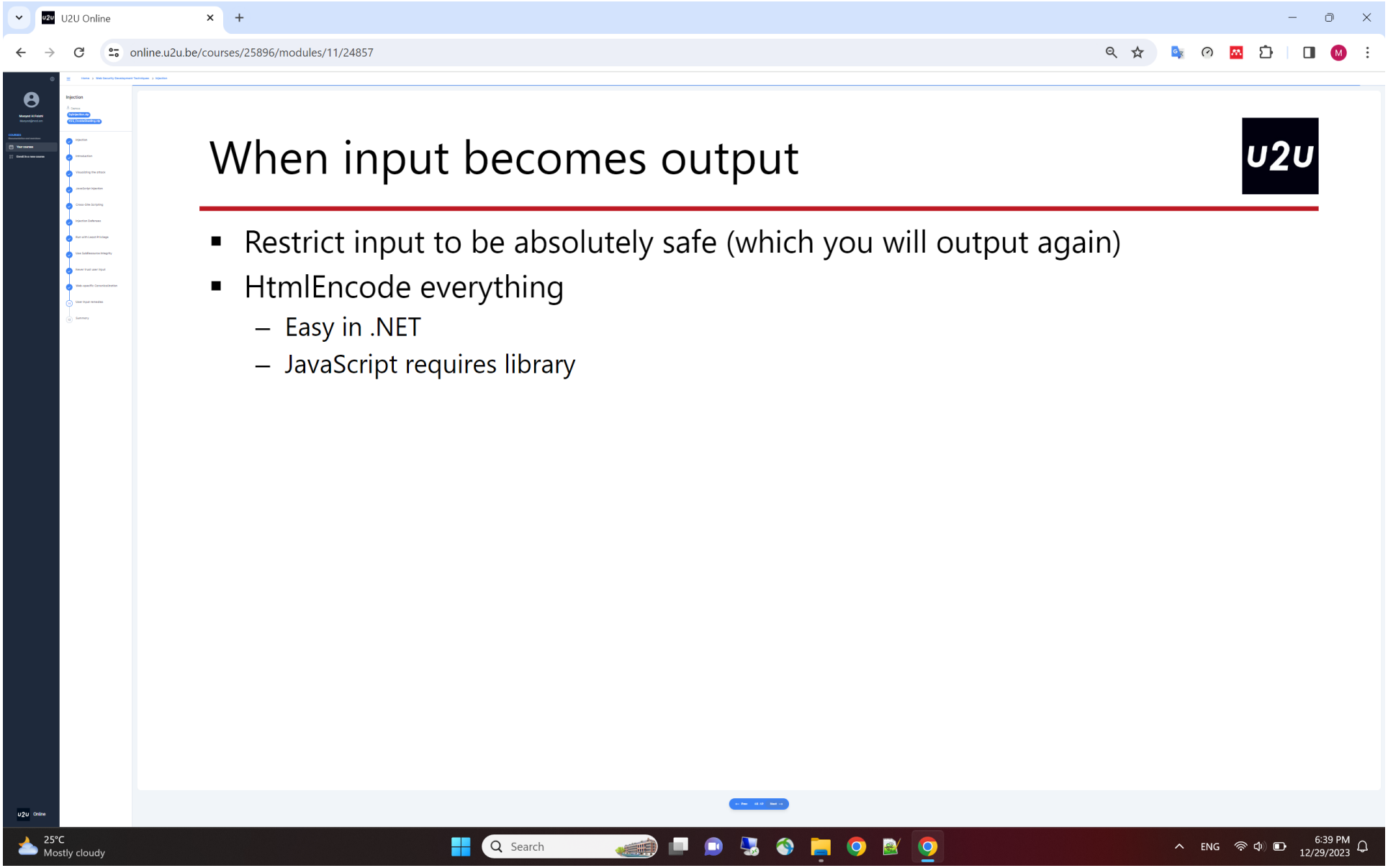

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Simple user input remedies
U2U
Be hardcore about user input
Determine what input is valid and reject the rest (no exceptions!)
Approaches to white-listing input
Use conversions
Integers, Dates, GUIDs
A string that can be converted to an integer will not contain malicious input
Use regular expressions
E-mail, telephone numbers, names
Be careful not to be too restrictive!
List all known good values
Countries, departments, ...
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023

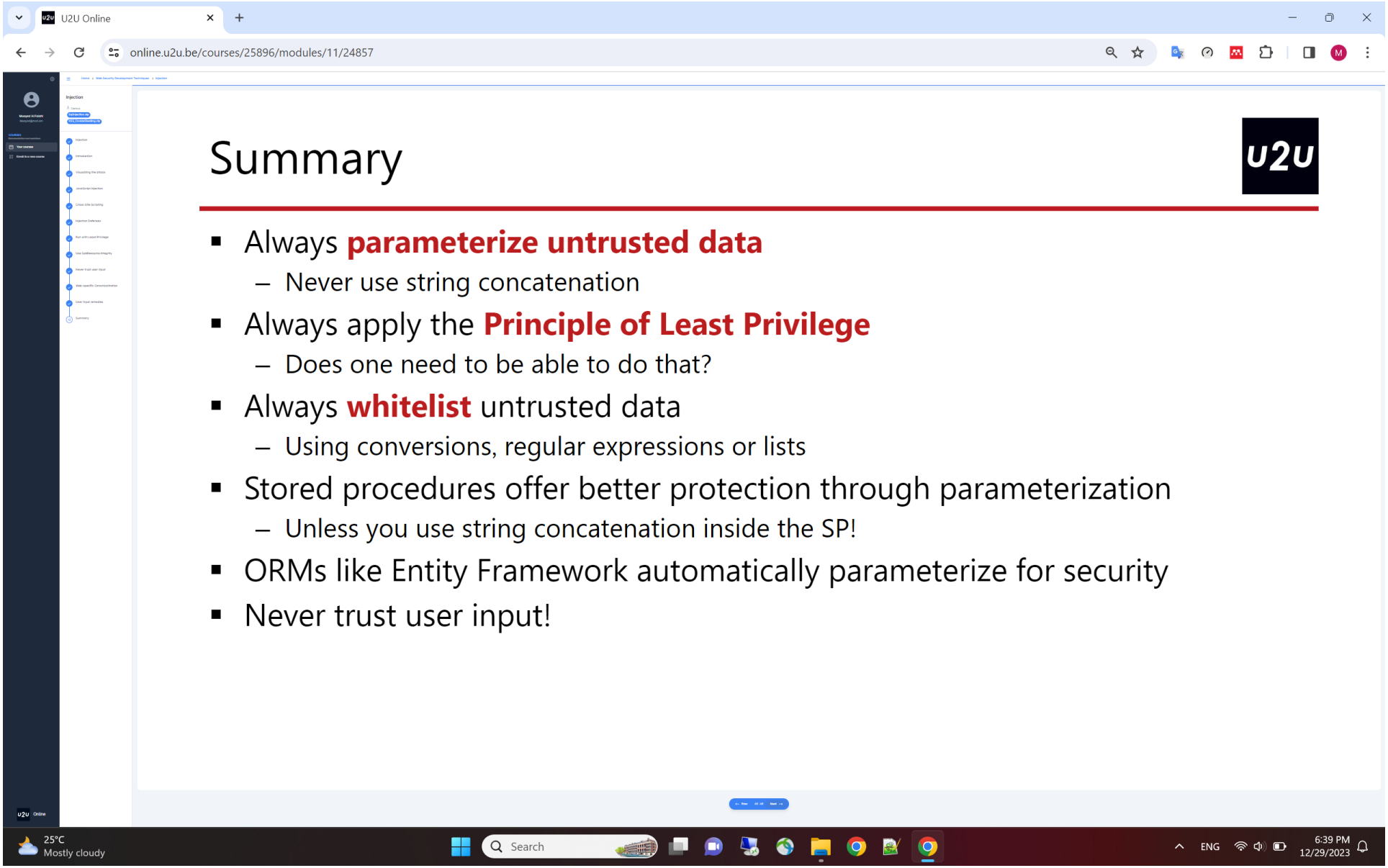

U2U Online
online.u2u.be/courses/25896/modules/11/24857
Hardcore checking with regular expressions
U2U
For example, using Regular Expressions to check product name
Product name can only contain characters, digits and spaces
Product name has to be between 1 and 32 characters long
public class ProductChecker
{
const string productNameRegEx = "^[A-Za-z0-9]{1,32}$";
private readonly Regex regEx = new Regex(productNameRegEx);
public bool CheckProductNameIsValid(string productName)
=> regEx.IsMatch(productName);
}
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Hardcore checking with regular expressions
U2U
Checking the URI with MVC routing
[Route("product/{name:regex(^[[A-Za-z0-9]]{{1,32}}$)}")]
public IActionResult Product(string name)
Verifying Form input
public class CommentViewModel
{
[Required]
[RegularExpression(@"^[A-Za-z0-9\s\.\,\!\?\'\""]+$")]
public string Comment { get; set; }
}
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Not being strict is dangerous
u2u
Some developers allow “safe” HTML constructs, e.g., <img> or <table>
Do NOT allow this!
XSS exploit is still there
<img src=javascript:alert(document.domain)>
<link rel=stylesheet href="javascript:alert(document.domain)">
…
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Special case: Passwords
u2u
Try not to limit the allowed characters in a password, or its length
Always hash the password before storing/retrieving it
See hashing and password policies
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
When input becomes output
U2U
Restrict input to be absolutely safe (which you will output again)
HtmlEncode everything
Easy in .NET
JavaScript requires library
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023


U2U Online
online.u2u.be/courses/25896/modules/11/24857
Summary
u2u
Always parameterize untrusted data
Never use string concatenation
Always apply the Principle of Least Privilege
Does one need to be able to do that?
Always whitelist untrusted data
Using conversions, regular expressions or lists
Stored procedures offer better protection through parameterization
Unless you use string concatenation inside the SP!
ORMs like Entity Framework automatically parameterize for security
Never trust user input!
25°C
Mostly cloudy
Search
ENG
6:39 PM
12/29/2023