International Cybersecurity and Digital Forensic Academy

PROGRAMME: CYBERSECURITY AND ETHICAL HACKING INTERNSHIP

ASSIGNMENT

PRESENTED BY

AYILARA BUSARI DARE

IDEAS/24/28133

COURSE CODE: INT309

COURSE TITLE: Web Technologies and Database Security

COURSE FACILITATOR:

8th February 2024

INT309: Web Technologies and Database Security - Lab 1: Introduction to Web Technologies and Understanding HTTP and HTTPS

Introduction

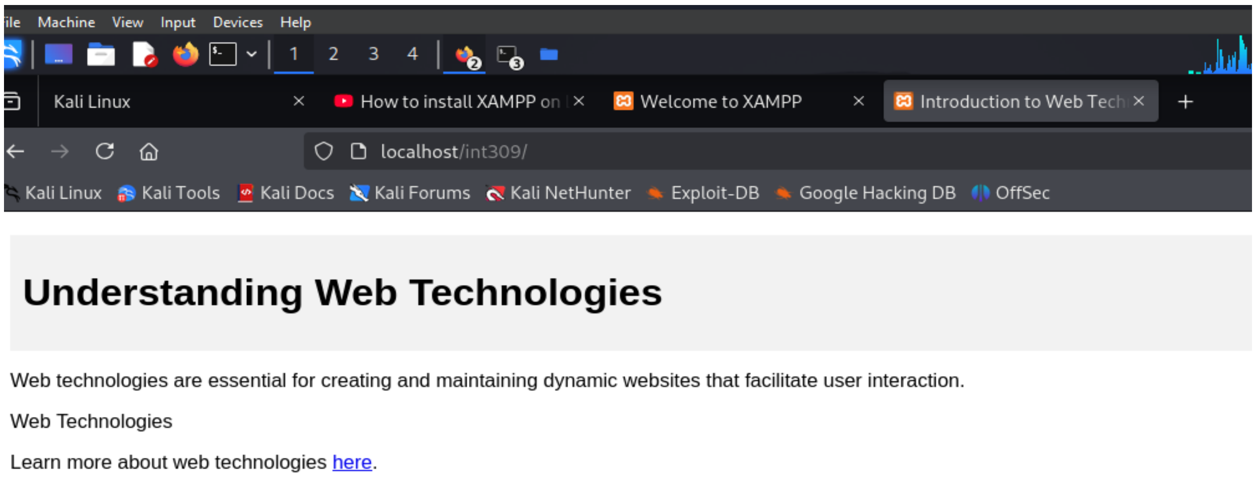
Web Technologies and Database Security encompass the measures and practices used to protect websites, web applications, and databases from cyber threats, ensuring data confidentiality, integrity, and availability.

What are Web Technologies: Web security refers to the measures taken to safeguard websites, web applications, and the underlying infrastructure from various cyber threats, including data breaches, malware, phishing attacks, and denial-of-service attacks.

Database Security: is a process of ensuring the right policies, measures and practice of used of database to protect database from unauthorized user or access, misuse, and data breaches in the data storage

Exercise 1: Creating a Simple Web Page

A web page created using html with file name index.php, the web page contains html structure such as the title, header that introduces the topic, the paragraph which explains the content of the website and link for further access external libraries.


ile Machine View Input Devices Help
1 2 3 4
Kali Linux
How to install XAMPP on
Welcome to XAMPP
Introduction to Web Tech
localhost/int309/
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Understanding Web Technologies

Web technologies are essential for creating and maintaining dynamic websites that facilitate user interaction.

Web Technologies

Learn more about web technologies here.

Reflection:

- Discuss what you learned about the structure of HTML documents. How do elements such as headers, paragraphs, and links contribute to a web page's content and usability?

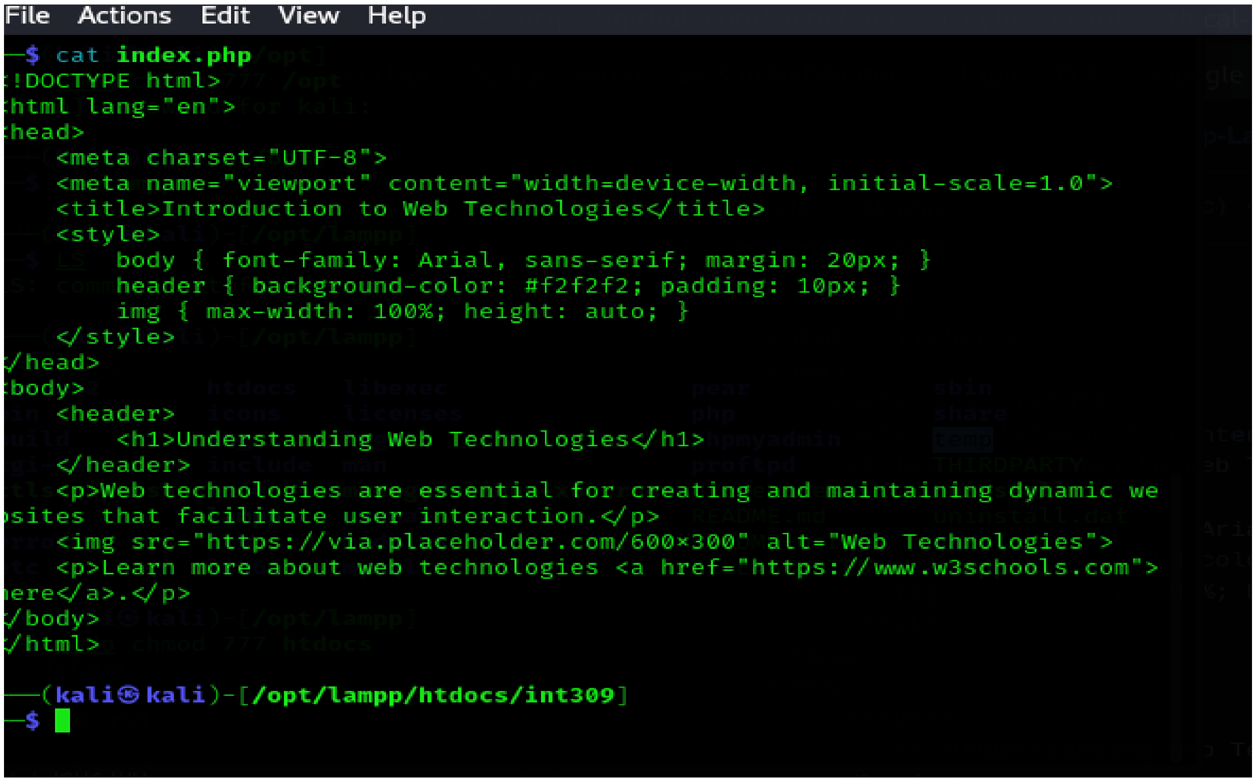
Answer:

The HTML document structures contain both open tags and close tags such as DOCTYPE html, html lang = "en", head, body and header.

The header tags have both open and close **tags (<header></header>)** this is used for housing the heading in the web page in order to introduce a topic, the design which usually contains **h1 to h6** tags depending on the desired heading front size.

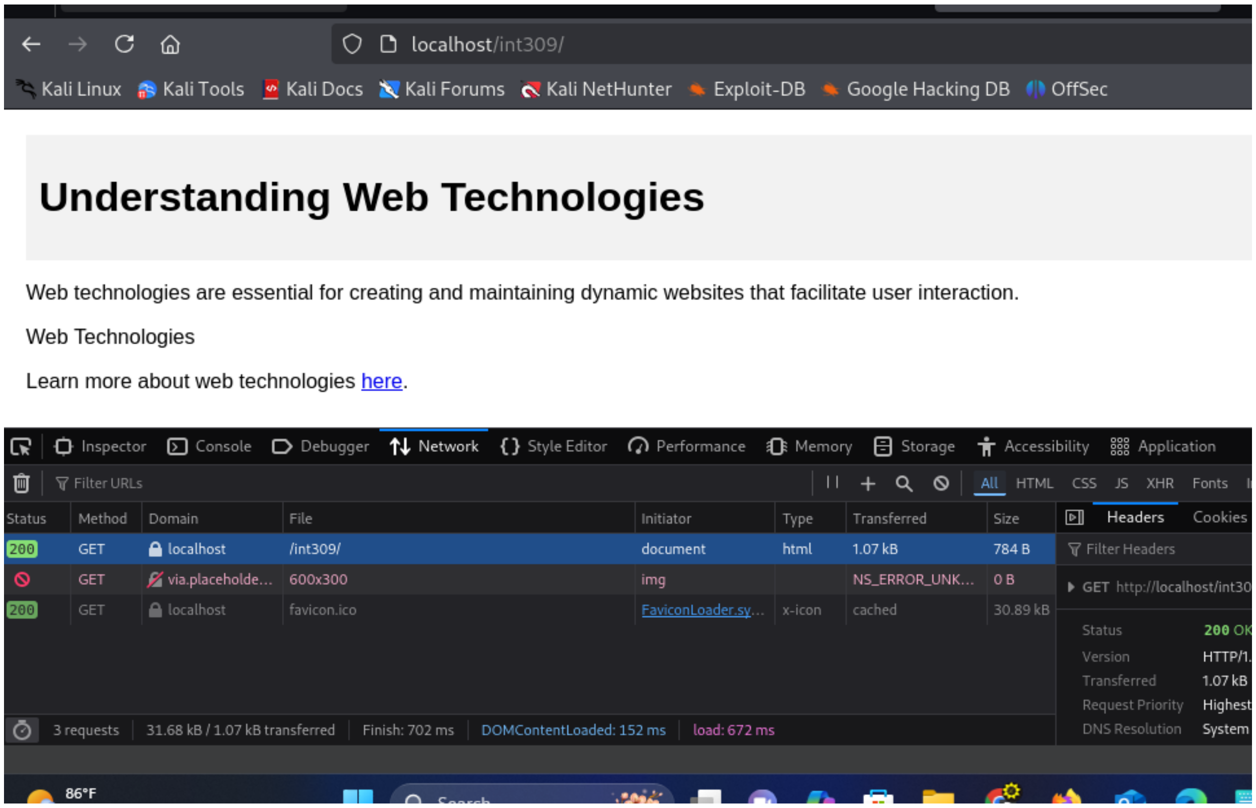
Paragraph in the HTML document is another tag that is represented **by tag <p></p>,** this differentiates one paragraph to the other and gives well arrange structure while navigation through the work.

Links contributed to easy navigating to an external resource for further learning or to accessing a page. This can be identified as **<a></a>**



```
—$ cat index.php
!DOCTYPE html>
html lang="en">
head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Introduction to Web Technologies</title>
   <style>
       body { font-family: Arial, sans-serif; margin: 20px; }
       header { background-color: #f2f2f2; padding: 10px; }
       img { max-width: 100%; height: auto; }
   </style>
/head>
body>
   <header>
       <h1>Understanding Web Technologies</h1>
   </header>
   <p>Web technologies are essential for creating and maintaining dynamic we
bsites that facilitate user interaction.</p>
   <img src="https://via.placeholder.com/600×300" alt="Web Technologies">
   <p>Learn more about web technologies <a href="https://www.w3schools.com">
ere</a>.</p>
/body>
/html>

——(kali㉿kali)-[/opt/lampp/htdocs/int309]
—$ 
```

Exercise 2: Analyzing HTTP Requests and Responses



Understanding Web Technologies

Web technologies are essential for creating and maintaining dynamic websites that facilitate user interaction.

Web Technologies

Learn more about web technologies here.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	localhost	/int309/	document	html	1.07 kB	784 B
	GET	via.placeholde...	600x300	img		NS_ERROR_UNK...	0 B
200	GET	localhost	favicon.ico	FaviconLoader.sy...	x-icon	cached	30.89 kB

3 requests | 31.68 kB / 1.07 kB transferred | Finish: 702 ms | DOMContentLoaded: 152 ms | load: 672 ms

Headers | Cookies

Filter Headers

GET http://localhost/int30

Status	200 OK
Version	HTTP/1.
Transferred	1.07 kB
Request Priority	Highest
DNS Resolution	System

Reflection:

- Analyze the significance of each element. Why is it important to understand these components when developing or securing web applications? Discuss the potential vulnerabilities associated with improperly handled requests and responses.

Answer:

To analyses each http requests element observed on the developer tool, go network tab, the request URL is the address of the web page visited which show where the website is secure or not.

Request Method shows that GET which means the sever was able to request and get index file from the int309 folder.

Response status code is 200 OK, this means a successful response status code indicates that a request has succeeded, as this show in the above pictures.

Response Headers contain different such content-type: text/html; charset=UTF-8, cache-control and date: Sat, 05 Apr 2025 11:12:27 GMT.

Why is it important to understand these components when developing or securing web applications? Discuss the potential vulnerabilities associated with improperly handled requests and responses.

1. **Request URL**:

- **Purpose:**

The URL (Uniform Resource Locator) specifies the location of the resource being requested from the server.

- **Importance:**

Developers need to understand URLs for routing requests to the correct endpoints, ensuring that the server can find and serve the requested content.

- **Security:**

Malformed or malicious URLs can lead to vulnerabilities like path traversal or injection attacks, making it essential to validate and sanitize URLs.

2. Request Method:

- **Purpose:**

The request method (e.g., GET, POST, PUT, DELETE) indicates the type of action the client wants to perform on the resource.

- **Importance:**

Developers need to use the correct method to ensure the server performs the intended operation, such as retrieving data (GET), creating a new resource (POST), updating an existing resource (PUT), or deleting a resource (DELETE).

- **Security:**

Misusing methods can lead to unintended consequences, such as deleting data when it shouldn't be deleted or allowing unauthorized access to resources.

3. Response Status Code:

- **Purpose:**

The status code (e.g., 200 OK, 404 Not Found, 500 Internal Server Error) informs the client about the outcome of the request.

- **Importance:**

Developers need to understand status codes to handle errors and provide meaningful feedback to users, such as displaying error messages or redirecting them to a different page.

- **Security:**

Certain status codes (like 403 Forbidden or 500 Internal Server Error) can indicate security issues or server problems, requiring developers to investigate and address them.

4. Response Headers:

- **Purpose:**

Response headers provide additional information about the response, such as the content type, server information, and catching instructions.

- **Importance:**

Developers need to understand headers to properly render the response, manage caching, and enforce security policies.

- **Security:**

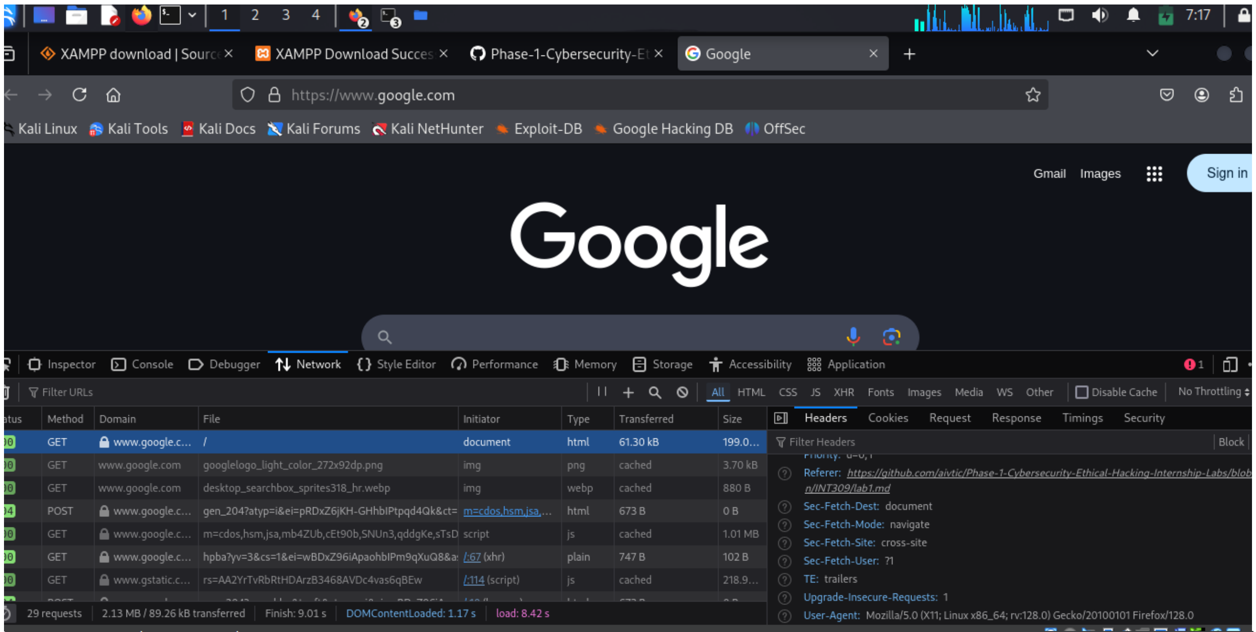
Headers like Content-Security-Policy (CSP) are crucial for preventing XSS (Cross-Site Scripting) and other security vulnerabilities, while Cache-Control headers help manage caching and improve performance

---

Exercise 3: Understanding HTTPS and Its Importance

Reflection:

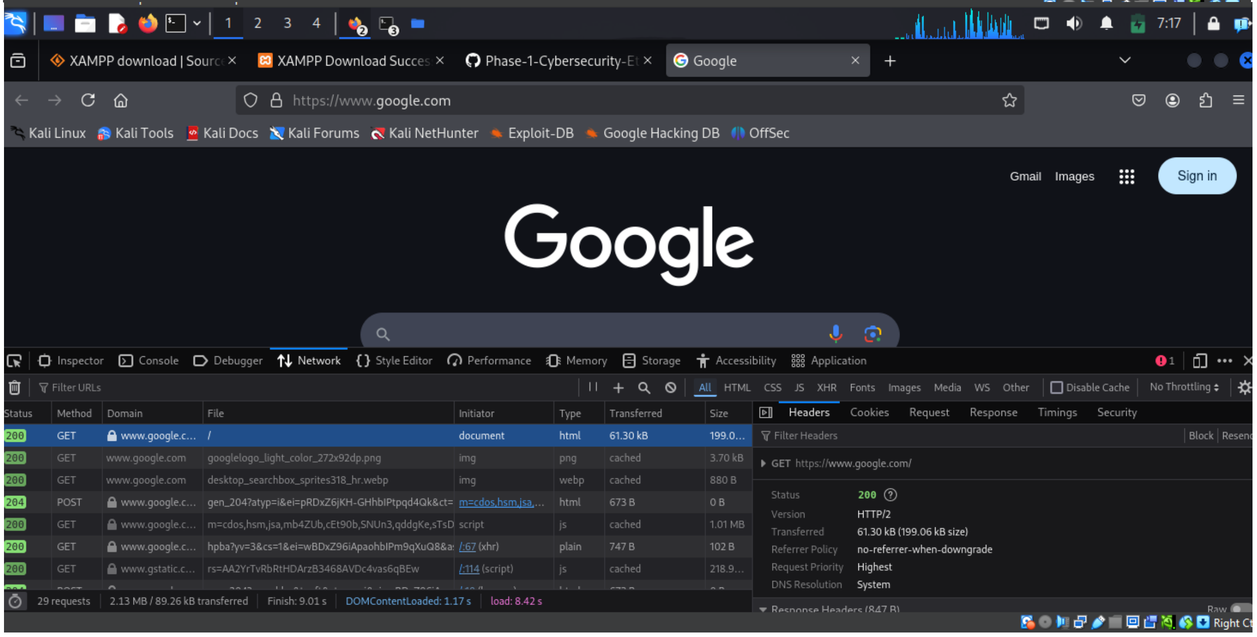
- Discuss the advantages of using HTTPS over HTTP, particularly in terms of data security and privacy. What protections does HTTPS provide against attacks such as eavesdropping and man-in-the-middle (MitM)?



Method	Domain	File	Initiator	Type	Transferred	Size
GET	www.google.c…	/	document	html	61.30 kB	199.0…
GET	www.google.com	googlelogo_light_color_272x92dp.png	img	png	cached	3.70 kB
GET	www.google.com	desktop_searchbox_sprites318_hr.webp	img	webp	cached	880 B
POST	www.google.c…	gen_204?atyp=i&ei=pRDxZ6jKH-GHhbIPtpqd4Qk&ct=	m=cdos,hsm,jsa,…	html	673 B	0 B
GET	www.google.c…	m=cdos,hsm,jsa,mb4ZUb,cEt90b,SNUn3,qddgKe,sTsD	script	js	cached	1.01 MB
GET	www.google.c…	hpba?yv=3&cs=1&ei=wBDxZ96iApaohbIPm9qXuQ8&a	/:67 (xhr)	plain	747 B	102 B
GET	www.gstatic.c…	rs=AA2YrTvRbRtHDArzB3468AVDc4vas6qBEw	/:114 (script)	js	cached	218.9…

29 requests | 2.13 MB / 89.26 kB transferred | Finish: 9.01 s | DOMContentLoaded: 1.17 s | load: 8.42 s

```
Referer: https://github.com/aivtic/Phase-1-Cybersecurity-Ethical-Hacking-Internship-Labs/blob
n/INT309/lab1.md
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
TE: trailers
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
```



Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.google.c…	/	document	html	61.30 kB	199.0…
200	GET	www.google.com	googlelogo_light_color_272x92dp.png	img	png	cached	3.70 kB
200	GET	www.google.com	desktop_searchbox_sprites318_hr.webp	img	webp	cached	880 B
204	POST	www.google.c…	gen_204?atyp=i&ei=pRDxZ6jKH-GHhbIPtpqd4Qk&ct=	m=cdos,hsm,jsa,…	html	673 B	0 B
200	GET	www.google.c…	m=cdos,hsm,jsa,mb4ZUb,cEt90b,SNUn3,qddgKe,sTsD	script	js	cached	1.01 MB
200	GET	www.google.c…	hpba?yv=3&cs=1&ei=wBDxZ96iApaohbIPm9qXuQ8&a	/:67 (xhr)	plain	747 B	102 B
200	GET	www.gstatic.c…	rs=AA2YrTvRbRtHDArzB3468AVDc4vas6qBEw	/:114 (script)	js	cached	218.9…

29 requests | 2.13 MB / 89.26 kB transferred | Finish: 9.01 s | DOMContentLoaded: 1.17 s | load: 8.42 s

```
GET https://www.google.com/

Status              200
Version             HTTP/2
Transferred         61.30 kB (199.06 kB size)
Referrer Policy     no-referrer-when-downgrade
Request Priority    Highest
DNS Resolution      System

Response Headers (847 B)
```

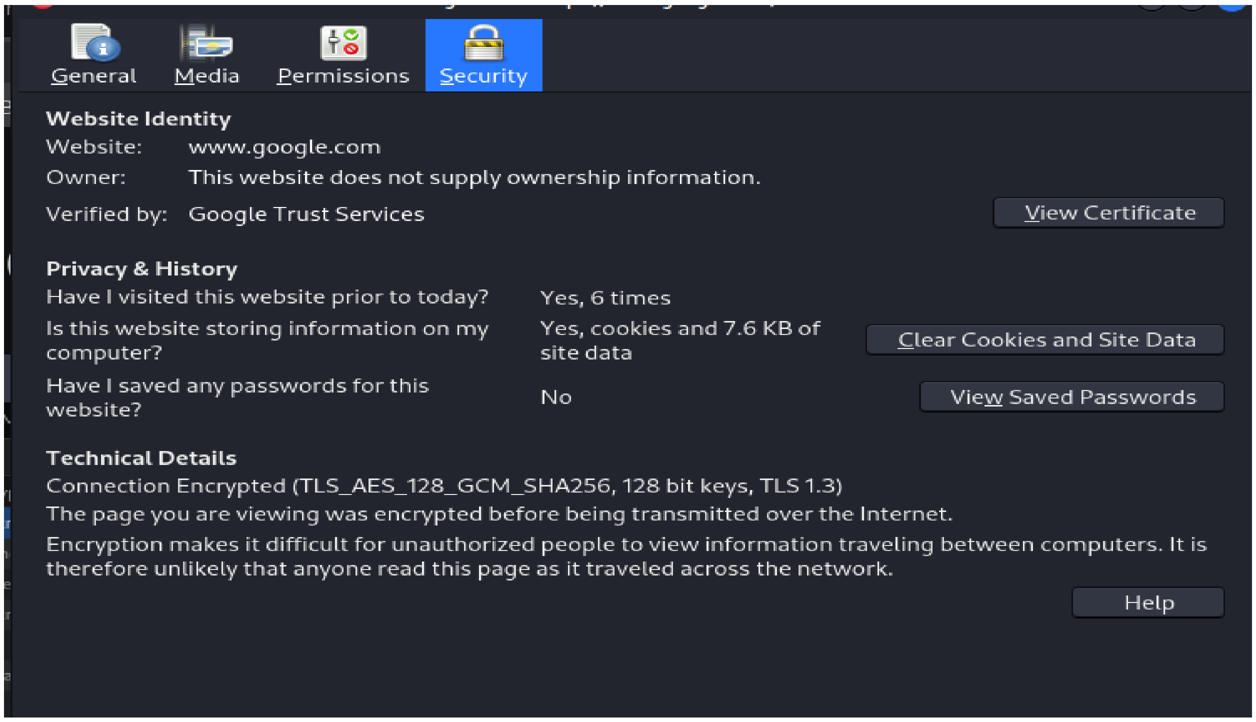
HTTPS offers superior data security and privacy compared to HTTP by encrypting data transmission, preventing eavesdropping and ensuring data integrity, while HTTP transmits data in plain text, making it vulnerable to interception and manipulation.

Encryption: HTTPS uses encryption (typically TLS/SSL) to scramble data during transmission, making it unreadable to anyone who intercepts the data, even if they have the tools to sniff network traffic.

Protection against eavesdropping: With HTTP, data is sent in plain text, which means anyone with the right tools can easily read the information being transmitted. HTTPS prevents this by encrypting the data.

Protection against man-in-the-middle attacks: A man-in-the-middle attack occurs when an attacker intercepts communication between a client and a server. HTTPS helps prevent this by verifying the identity of the server and encrypting the communication.

A padlock icon shows on the https address which indicates secure web address and safe to visit, when click on this, it allows to view security certificates of such website, the encryption method, date of registration, expiring date and issuer. As shown below.



Website Identity

Website: www.google.com

Owner: This website does not supply ownership information.

Verified by: Google Trust Services

Privacy & History

Have I visited this website prior to today? Yes, 6 times

Is this website storing information on my computer? Yes, cookies and 7.6 KB of site data

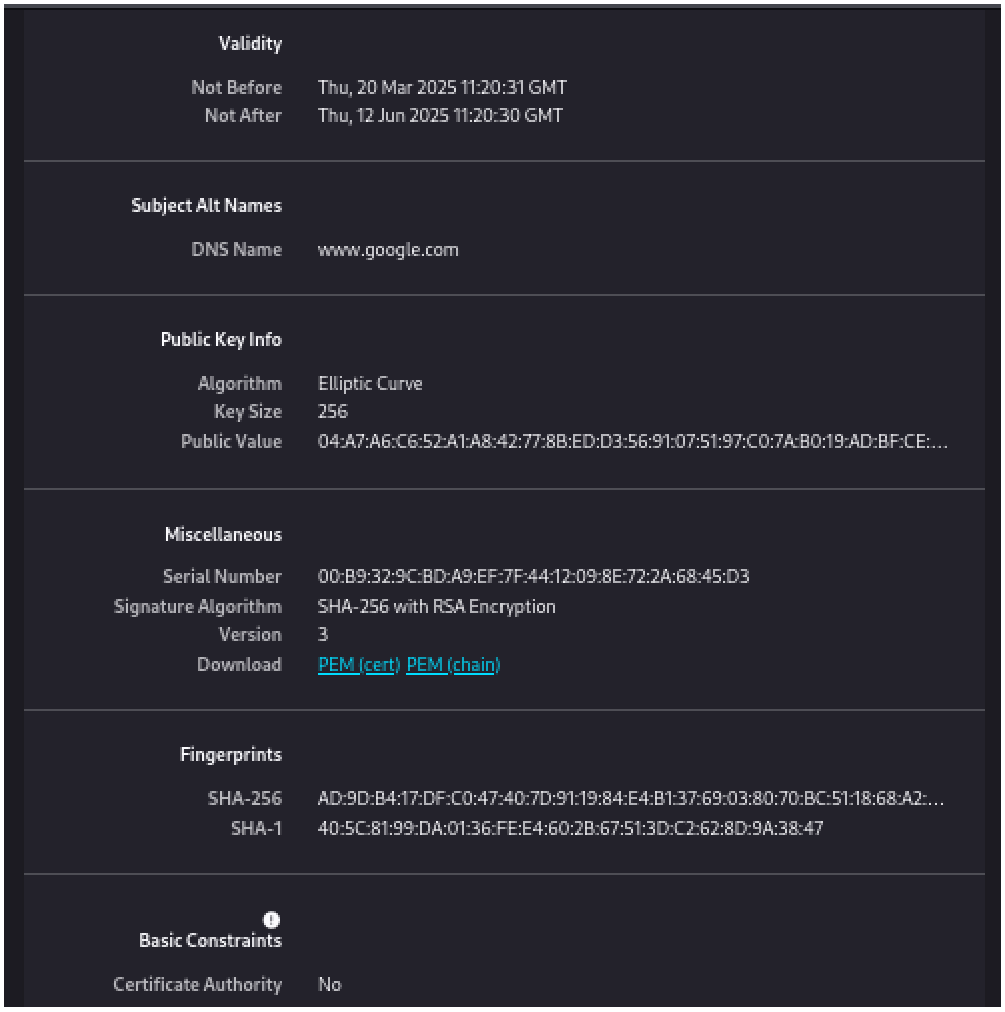
Have I saved any passwords for this website? No

Technical Details

Connection Encrypted (TLS_AES_128_GCM_SHA256, 128 bit keys, TLS 1.3)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

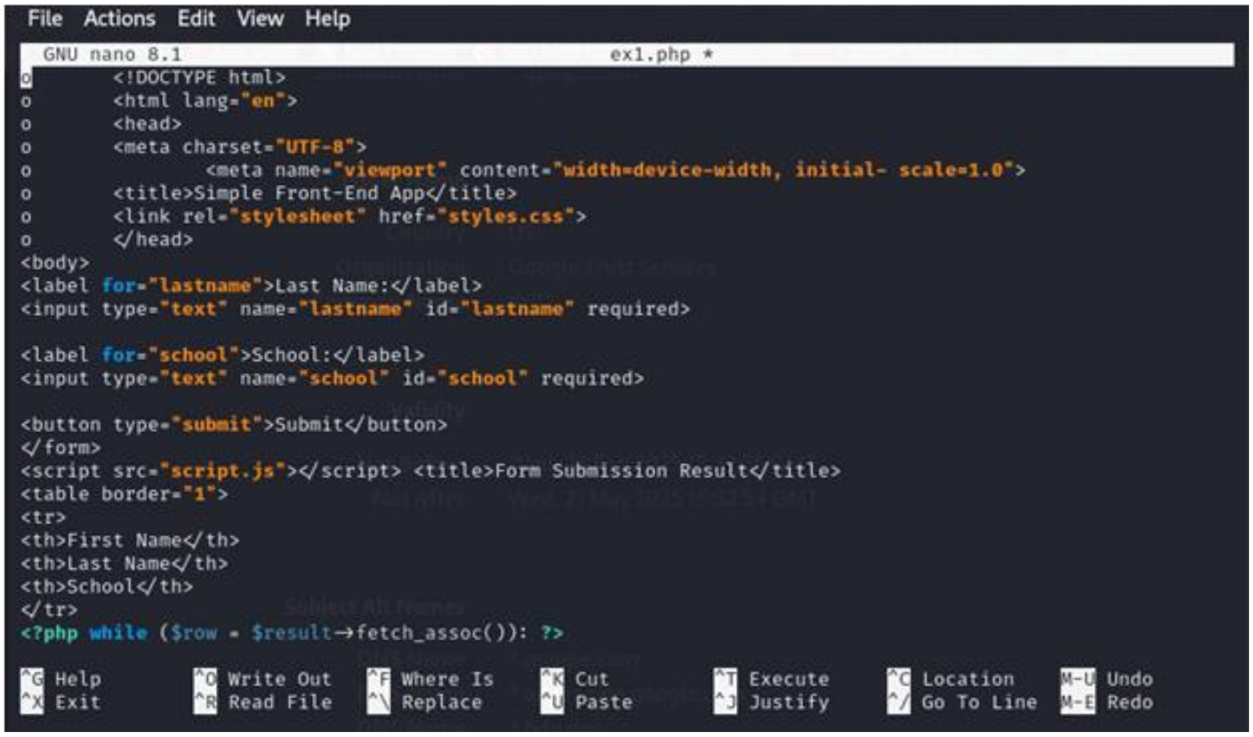


Validity	
Not Before	Thu, 20 Mar 2025 11:20:31 GMT
Not After	Thu, 12 Jun 2025 11:20:30 GMT
Subject Alt Names	
DNS Name	www.google.com
Public Key Info	
Algorithm	Elliptic Curve
Key Size	256
Public Value	04:A7:A6:C6:52:A1:A8:42:77:8B:ED:D3:56:91:07:51:97:C0:7A:B0:19:AD:BF:CE:…
Miscellaneous	
Serial Number	00:B9:32:9C:BD:A9:EF:7F:44:12:09:8E:72:2A:68:45:D3
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)
Fingerprints	
SHA-256	AD:9D:B4:17:DF:C0:47:40:7D:91:19:84:E4:B1:37:69:03:80:70:BC:51:18:68:A2:…
SHA-1	40:5C:81:99:DA:01:36:FE:E4:60:2B:67:51:3D:C2:62:8D:9A:38:47
Basic Constraints	
Certificate Authority	No

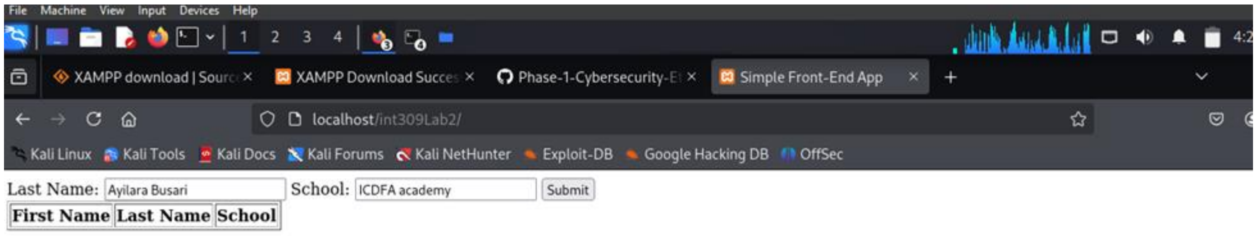
INT309: Web Technologies and Database Security – Lab 2: Web Development Components and Security Threats.

Exercise 1: Exploring Front-End Development Components

1. **Identify Front-End Technologies:**
   - Research and create a list of common front-end technologies used in web development, including:
     - ▪ HTML
     - ▪ CSS
     - ▪ JavaScript
     - ▪ Front-end frameworks (e.g., React, Angular, Vue.js)
2. **Create a Simple Front-End Application:**
   - Build a basic web application using HTML, CSS, and JavaScript that includes:
     - ▪ A header with a navigation menu.
     - ▪ A form for user input (e.g., contact form).
     - ▪ Interactive elements (e.g., a button that changes the text when clicked).

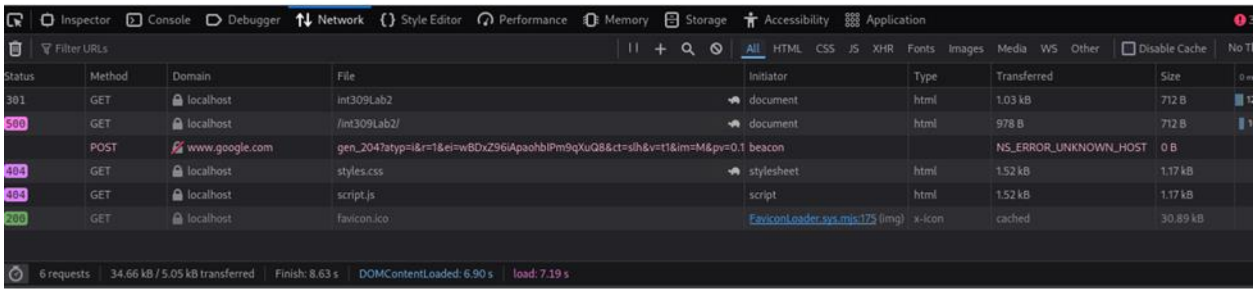


```
o       <!DOCTYPE html>
o       <html lang="en">
o       <head>
o       <meta charset="UTF-8">
o               <meta name="viewport" content="width=device-width, initial- scale=1.0">
o       <title>Simple Front-End App</title>
o       <link rel="stylesheet" href="styles.css">
o       </head>
<body>
<label for="lastname">Last Name:</label>
<input type="text" name="lastname" id="lastname" required>

<label for="school">School:</label>
<input type="text" name="school" id="school" required>

<button type="submit">Submit</button>
</form>
<script src="script.js"></script> <title>Form Submission Result</title>
<table border="1">
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>School</th>
</tr>
<?php while ($row = $result→fetch_assoc()): ?>
```


localhost/int309Lab2/

Last Name: Ayilara Busari School: ICDFA academy Submit

First Name	Last Name	School



Status	Method	Domain	File	Initiator	Type	Transferred	Size
301	GET	localhost	int309Lab2	document	html	1.03 kB	712 B
500	GET	localhost	/int309Lab2/	document	html	978 B	712 B
	POST	www.google.com	gen_204?atyp=i&r=1&ei=wBDxZ96iApaohblPm9qXuQ8&ct=slh&v=t1&im=M&pv=0.1	beacon		NS_ERROR_UNKNOWN_HOST	0 B
404	GET	localhost	styles.css	stylesheet	html	1.52 kB	1.17 kB
404	GET	localhost	script.js	script	html	1.52 kB	1.17 kB
200	GET	localhost	favicon.ico	FaviconLoader.sys.mjs:175 (img)	x-icon	cached	30.89 kB

6 requests | 34.66 kB / 5.05 kB transferred | Finish: 8.63 s | DOMContentLoaded: 6.90 s | load: 7.19 s

Reflection:

- Discuss how front-end technologies interact to create a cohesive user experience. What potential security issues arise from poorly implemented front-end code (e.g., XSS, CSRF)?

Front end technologies are an interface a user can interact with while making use of web pages and this will enhance the user experience if the design is good and user friendly.

- **HTML:** this is skeleton structure of web design; it is used for content write up and define element of web page.
- **CSS:** it is used as a style website for attractive user interface
- **JavaScript**: this is used for interactive and to build a responsive website.
- Front-end frameworks (e.g., React, Angular, Vue.js**):** these are frame that already contains inbuilt html, css and Javascript and its easy for building a website.

Security Issues from Poorly Implemented Front-End Code

Potential security issues may arise due to poor implementation of front-end code such as **improper sanitization of user input in JavaScript code**, this can cause problems such as **cross site script (XSS)** and **Cross-Site Request Forgery (CSRF)**.

- **Cross site script (XSS)**: where a malicious actor inject malicious script into website or stores it in the vulnerable code which are then executed by other users browsers, potentially compromising their data or action. If a website is not properly implemented it can also lead to CSRF where an attacker crafts a malicious website or clicks the link, their browser automatically includes their authentication cookies (or other session data) with the request to the target website.

- **Cross-Site Request Forgery (CSRF)**: Exploits user sessions to execute unauthorized actions on trusted websites.
- **Clickjacking:** Embeds a malicious site within an invisible frame, tricking users into unintended actions.
- **Insecure JavaScript Libraries:** Using outdated or vulnerable libraries can expose websites to attacks.

Mitigation Strategies: Implement Content Security Policy (CSP), input validation, secure authentication, and regular updates to prevent these threats.

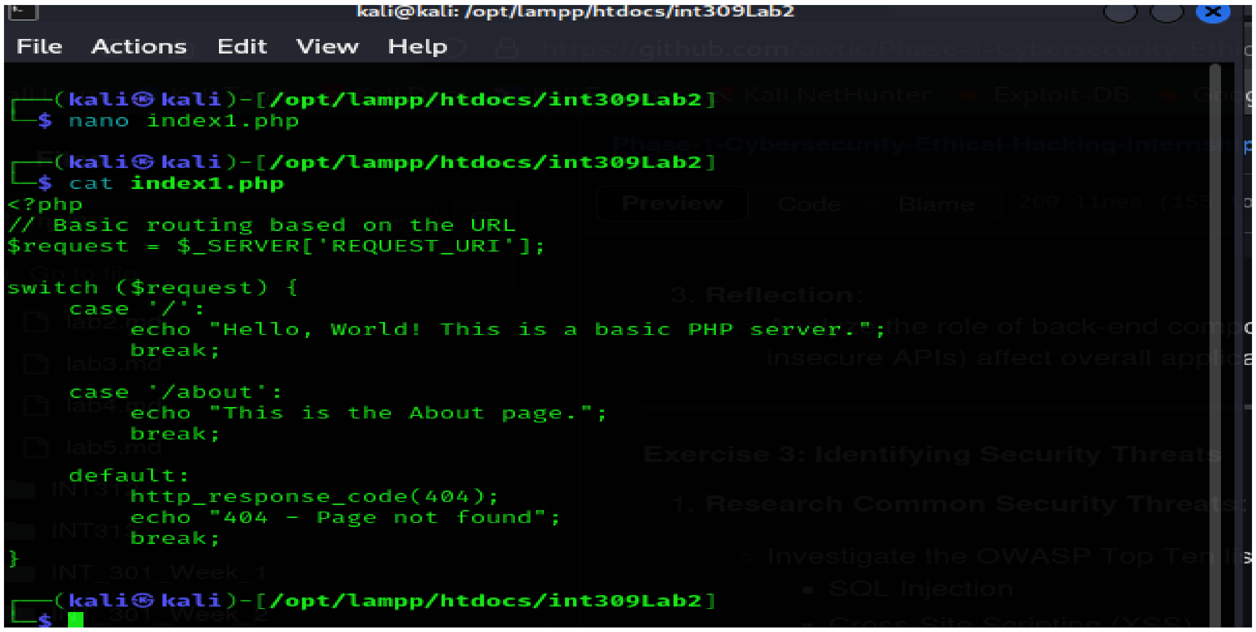
Exercise 2: Understanding Back-End Development Components

1. Identify Back-End Technologies:

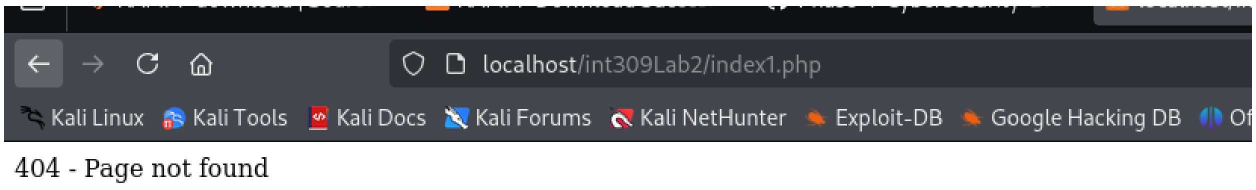
- Programming Languages: PHP, Python, Java, Node.js, Ruby
- Databases: MySQL, PostgreSQL, MongoDB, SQLite
- Web Frameworks: Django (Python), Express.js (Node.js), Laravel (PHP), Spring Boot (Java)

Create a Simple Back-End Application:

- Using a chosen back-end technology, set up a basic server that responds to HTTP requests. For example, using Node.js:


kali@kali: /opt/lampp/htdocs/int309Lab2
File Actions Edit View Help

```
┌──(kali㉿kali)-[/opt/lampp/htdocs/int309Lab2]
└─$ nano index1.php

┌──(kali㉿kali)-[/opt/lampp/htdocs/int309Lab2]
└─$ cat index1.php
<?php
// Basic routing based on the URL
$request = $_SERVER['REQUEST_URI'];

switch ($request) {
    case '/':
        echo "Hello, World! This is a basic PHP server.";
        break;

    case '/about':
        echo "This is the About page.";
        break;

    default:
        http_response_code(404);
        echo "404 - Page not found";
        break;
}

┌──(kali㉿kali)-[/opt/lampp/htdocs/int309Lab2]
└─$ 
```


localhost/int309Lab2/index1.php

404 - Page not found

Reflection:

- Analyze the role of back-end components in web applications. How do vulnerabilities in the back-end (e.g., SQL injection, insecure APIs) affect overall application security?

Ans

Back-end components are the foundation of web applications, handling data storage, processing, and core functionalities, so vulnerabilities in these areas can lead to severe security breaches, including data theft, unauthorized access, and system manipulation.

Role of Back-End Components:

- **Data Storage and Management:**

Back-end systems, including databases, store and manage sensitive user data, financial information, and other critical data.

- **Application Logic and Functionality:**

The back-end executes the core logic of the application, handling user requests, processing data, and interacting with external services.

- **Authentication and Authorization:**

Back-end systems are responsible for verifying user identities and controlling access to resources, ensuring only authorized users can access sensitive information.

- **API Management:**

Back-end components often expose APIs, which are the interfaces that allow different applications to communicate and exchange data.

Vulnerabilities and Their Impact:

- **SQL Injection:**

A common vulnerability that allows attackers to manipulate database queries, potentially gaining unauthorized access to data, modifying data, or even gaining control of the database server.

- **Insecure APIs:**

APIs that lack proper authentication, authorization, or input validation can be exploited by attackers to gain unauthorized access to data or functionalities.

- **Cross-Site Scripting (XSS):**

An attacker can inject malicious scripts into a web application, allowing them to steal user data, redirect users to malicious sites, or compromise the user's session.

- **Cross-Site Request Forgery (CSRF):**

An attacker can trick a user into performing actions on a web application without their knowledge, such as making unauthorized changes to their account or data.

- **Broken Authentication and Authorization:**

Weak authentication mechanisms or insufficient authorization controls can allow attackers to bypass security measures and gain unauthorized access to sensitive data or functionalities.

- **Data Breaches:**

Compromised back-end systems can lead to data breaches, where sensitive information is stolen and exposed to unauthorized parties.

- **Unauthorized Access:**

Vulnerabilities can allow attackers to gain unauthorized access to the application, database, or underlying server.

- **System Manipulation:**

Attackers can exploit vulnerabilities to manipulate the application's functionality, potentially causing disruptions or even taking control of the system.

- **Financial Loss:**

Data breaches and system compromises can lead to significant financial losses for businesses, including costs associated with remediation, legal fees, and reputational damage.

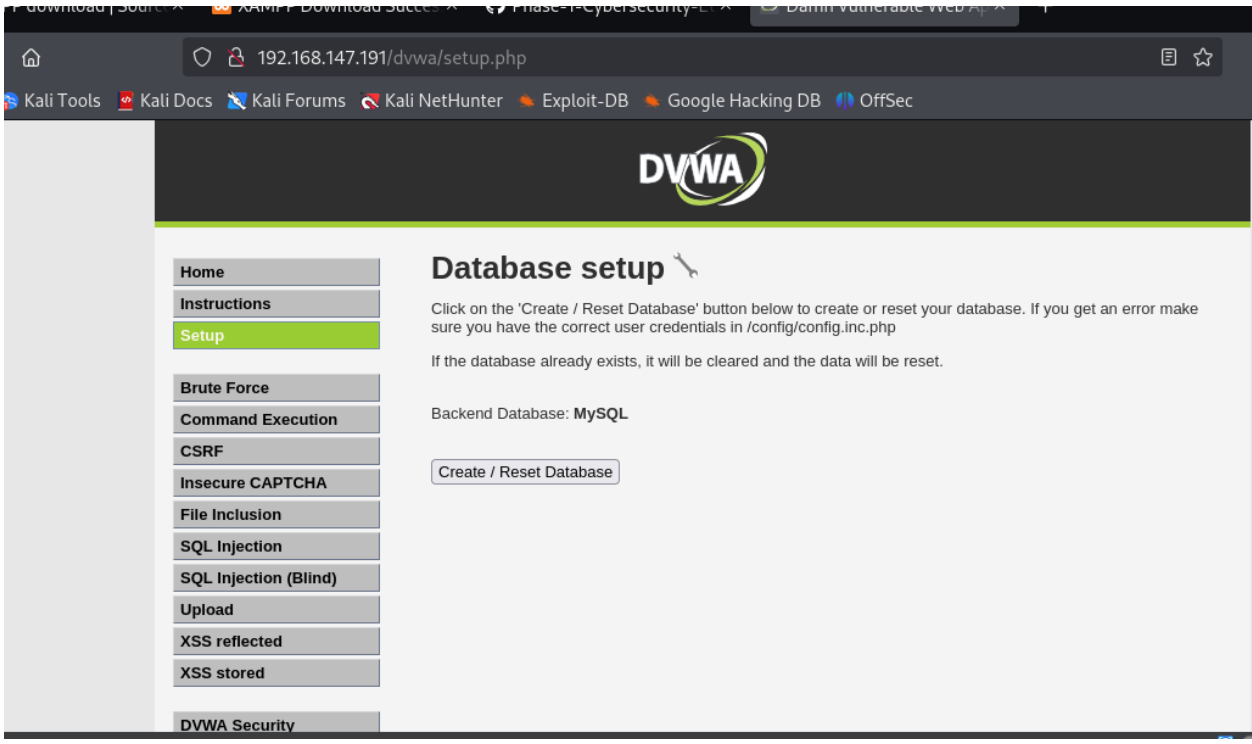
- **Reputational Damage:**

Security breaches can severely damage a company's reputation, leading to loss of customer trust and potential business losses.

Exercise 3: Identifying Security Threats

• **Research Common Security Threats**:

- Investigate the OWASP Top Ten list and identify at least five common security threats in web development. Focus on:
  - SQL Injection
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)
  - Security Misconfiguration
  - Insecure Deserialization



Database setup

Click on the 'Create / Reset Database' button below to create or reset your database. If you get an error make sure you have the correct user credentials in /config/config.inc.php

If the database already exists, it will be cleared and the data will be reset.

Backend Database: **MySQL**

Reflection:

- Discuss the impact of these security threats on user data and application integrity. What measures can developers implement to mitigate these risks?

Impact of Security Threats & Mitigation Strategies SQL Injection (SQLi)

Impact:

• Attackers can bypass authentication and gain unauthorized access.

• User data leakage, leading to identity theft.

• Attackers can delete, modify, or steal entire databases.

• Some SQLi attacks enable remote code execution, allowing full system control.

Mitigation Strategies:

✓ Use Prepared Statements (Parameterized Queries) to prevent injection.

✓ Use Web Application Firewalls (WAFs) to filter malicious requests.

✓ Limit database permissions for application users (least privilege).

✓ Sanitize and validate user input before processing it.

Cross-Site Scripting (XSS)

Impact:

• Attackers inject malicious JavaScript into web pages.

• Can steal cookies, session tokens, and login credentials.

• Redirect users to phishing sites or deface the website.

• Can spread malware through user interactions.

Mitigation Strategies:

✓ Escape user input to prevent JavaScript execution.

✓ Use Content Security Policy (CSP) to restrict script execution.

✓ Sanitize input fields to allow only safe characters.

✓ Implement HTTP-only and Secure cookies to prevent cookie theft.

Cross-Site Request Forgery (CSRF)

Impact:

• Attackers trick users into executing unwanted actions on their accounts.

• Can lead to unauthorized fund transfers, password changes, or account deletion.

• Exploits users who are already logged in to the target application.

Mitigation Strategies:

✓ Use CSRF tokens to verify request authenticity.

✓ Enforce SameSite cookie attributes to restrict cross-site requests.

✓ Implement Multi-Factor Authentication (MFA) for critical actions.

✓ Use Referrer and Origin headers to validate requests.

Security Misconfiguration

Impact:

• Default credentials (e.g., admin/admin) expose sensitive data.

• Excessive permissions can lead to privilege escalation.

• Unnecessary features or debug modes increase attack surface.

• Insecure error messages reveal system details to attackers.

Mitigation Strategies:

✓ Disable default accounts and change default passwords.

✓ Use the principle of least privilege for permissions.

✓ Regularly update and patch software to fix vulnerabilities.

✓ Disable unnecessary services and features in production.

Insecure Deserialization

Impact:

• Attackers can manipulate serialized objects to execute malicious code.

• Can lead to remote code execution (RCE) and system compromise.

• Can be used to elevate privileges or tamper with application data.

Mitigation Strategies:

✓ Avoid using serialized objects from untrusted sources.

✓ Use digital signatures to verify object integrity.

✓ Implement strong input validation before deserializing data.

✓ Use safer data formats like JSON instead of serialized objects.

Conclusion

Security threats like SQLi, XSS, CSRF, Security Misconfigurations, and Insecure Deserialization can compromise user data, application integrity, and system security. Developers should implement secure coding practices, input validation, least privileged principles, and security headers to mitigate these risks.

INT309: Web Technologies and Database Security – Lab 3: Introduction to Databases and SQL

Definition: SQL stands for Structured Query Language, SQL is a standard language for storing, manipulating and retrieving data in databases

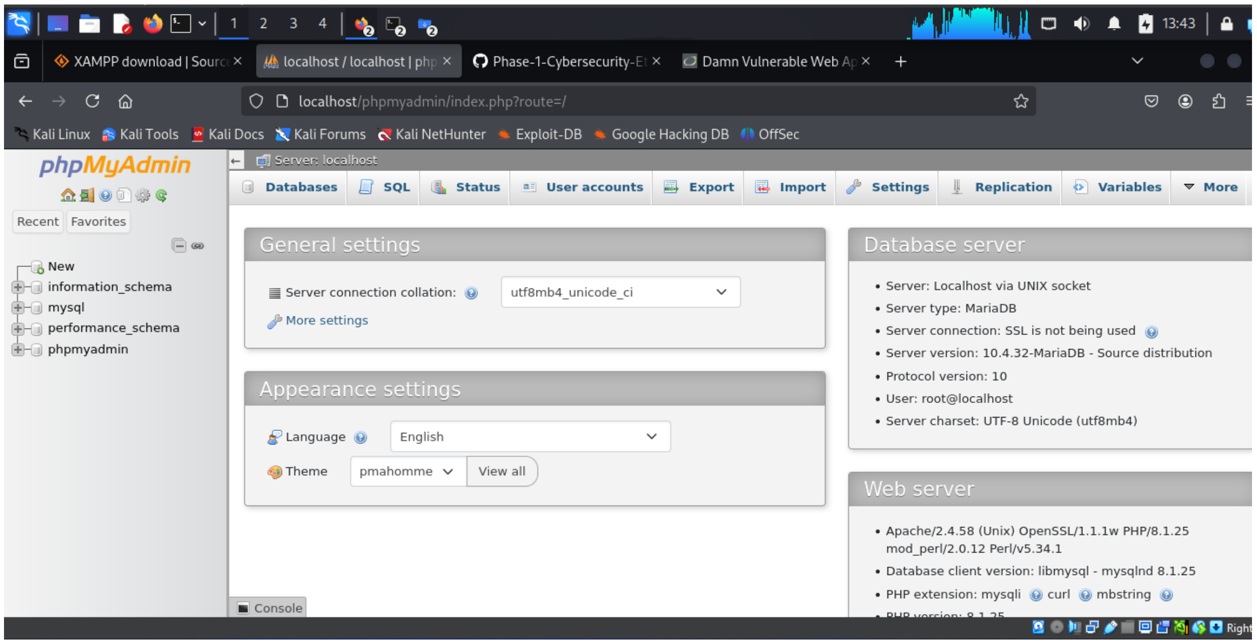
Exercise 1: Setting Up MySQL with XAMPP

Install and Run XAMPP:

- If you haven't already, download and install XAMPP from https://www.apachefriends.org/.
- Launch XAMPP and start the **Apache** and **MySQL** services from the XAMPP Control Panel.

Access phpMyAdmin:

- Open your web browser and navigate to http://localhost/phpmyadmin/.
- phpMyAdmin is a web-based tool provided by XAMPP for managing MySQL databases



General settings

Server connection collation: utf8mb4_unicode_ci

More settings

Appearance settings

Language: English

Theme: pmahomme | View all

Database server

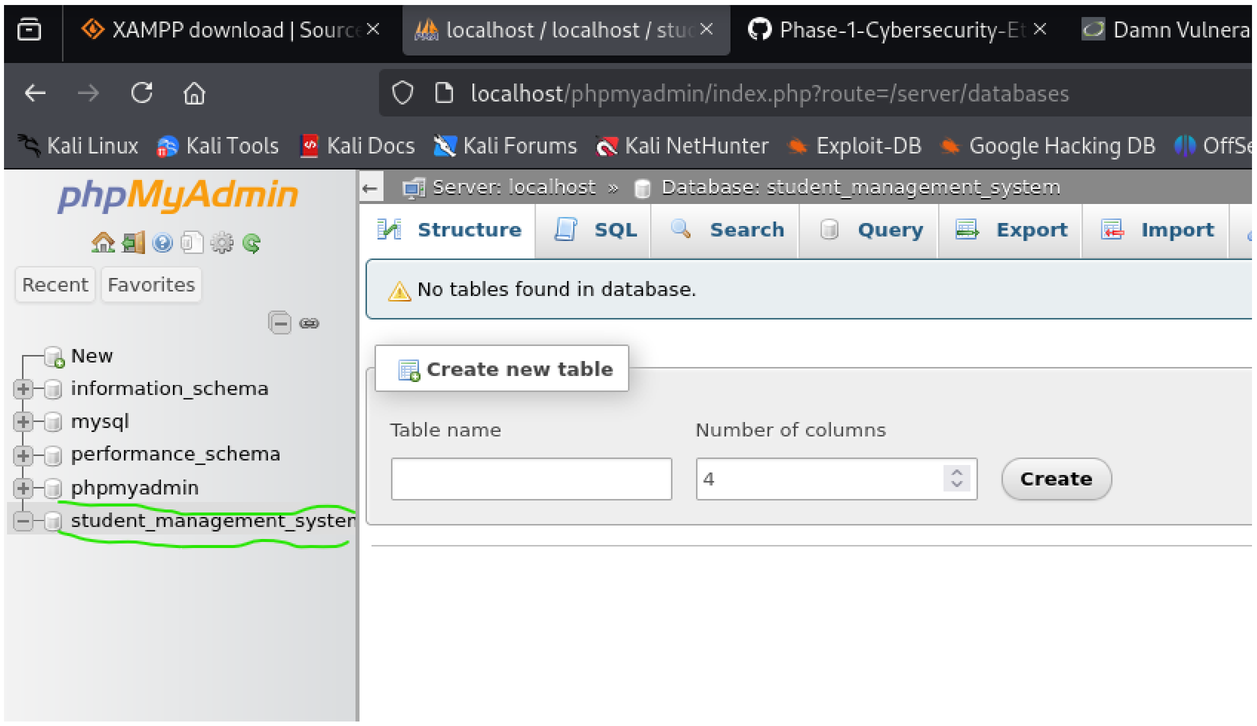
- Server: Localhost via UNIX socket
- Server type: MariaDB
- Server connection: SSL is not being used
- Server version: 10.4.32-MariaDB - Source distribution
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8mb4)

Web server

- Apache/2.4.58 (Unix) OpenSSL/1.1.1w PHP/8.1.25 mod_perl/2.0.12 Perl/v5.34.1
- Database client version: libmysql - mysqlnd 8.1.25
- PHP extension: mysqli curl mbstring
- PHP version: 8.1.25

Create a New Database:

- In phpMyAdmin, click on the **Databases** tab.
- Enter the name *student_management_system* for your new database and click **Create**.
- This database will store information about students, courses, and enrollments.



No tables found in database.

Reflection:

- Explain why a web application like a Student Management System would need a relational database. What are the benefits of using MySQL in such applications?

ANSWER

Databases are essential for the role it play when come to data storage in web application such as student Management system, In the school environment the administration need to save student data e.g. student results, Bio data, Matric No, Blood-group etc. for easy access when allocate a student. And it is easy to navigate and query table for editing and deleting data.

Benefits of MySQL in such Applications are:

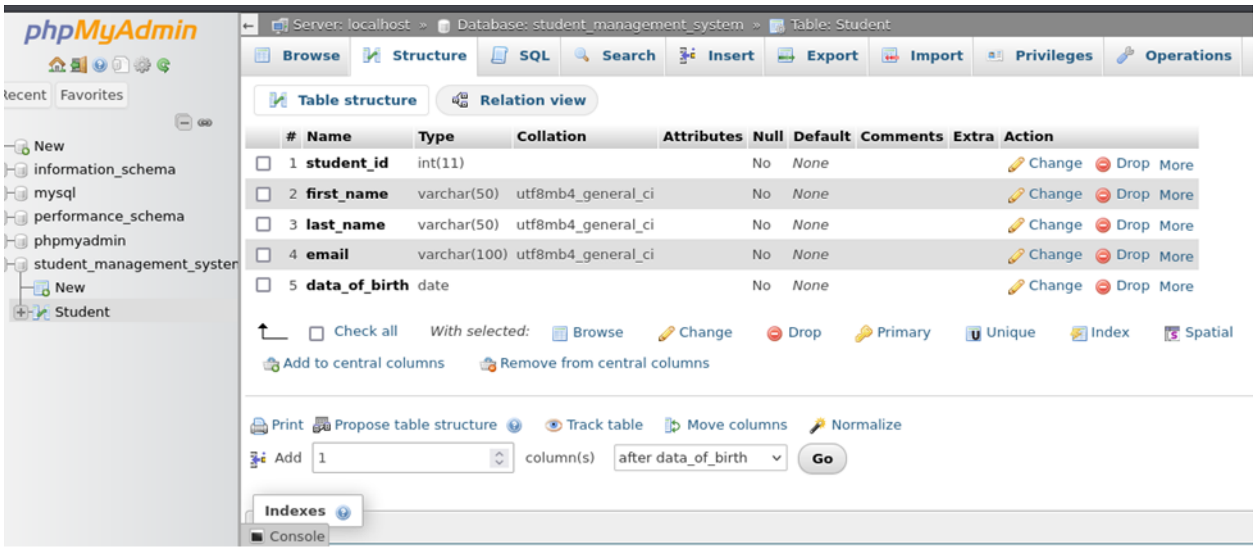
- **Manipulation of data:** it can be used to add, delete and edit data from database.
- **Store data:** it can be used to store and query table from database
- **retrieving data in databases:** it can be used to recall data and search for any data on the database

---

Exercise 2: Creating Tables and Defining Columns

Create a students Table:

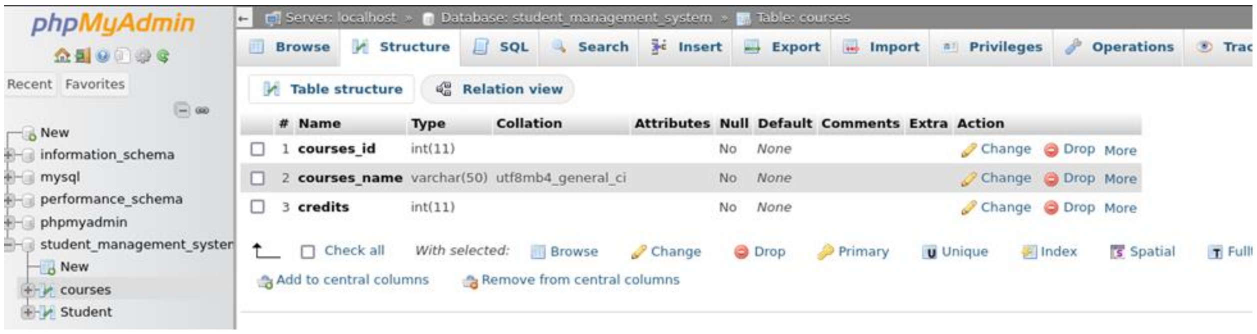
- After creating the database, click on the student_management_system database.
- Create a new table named students with the following columns:
  - student_id (INT, Primary Key, Auto Increment)
  - first_name (VARCHAR(50))
  - last_name (VARCHAR(50))
  - email (VARCHAR(100), Unique)
  - date_of_birth (DATE)
- Click **Save** to create the table.


phpMyAdmin
Recent Favorites
New
information_schema
mysql
performance_schema
phpmyadmin
student_management_syster
New
Student
Server: localhost » Database: student_management_system » Table: Student
Browse Structure SQL Search Insert Export Import Privileges Operations
Table structure Relation view
Check all With selected: Browse Change Drop Primary Unique Index Spatial
Add to central columns Remove from central columns
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after data_of_birth Go
Indexes
Console

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	student_id	int(11)			No	None			Change Drop More
2	first_name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
3	last_name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
4	email	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
5	data_of_birth	date			No	None			Change Drop More

Create a courses Table:

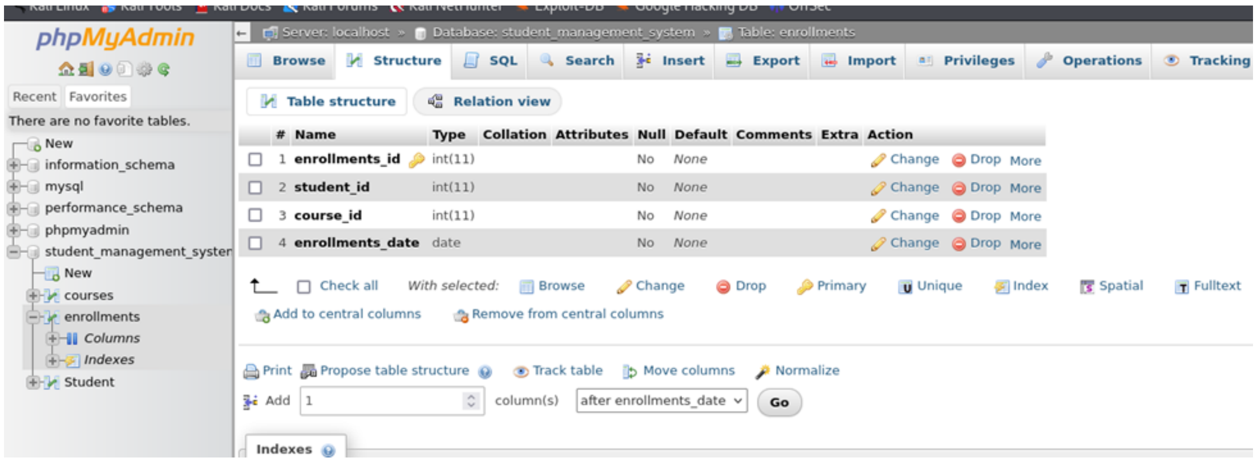
- Create a second table named courses with the following columns:
    - course_id (INT, Primary Key, Auto Increment)
    - course_name (VARCHAR (100))
    - credits (INT)
- Click **Save**.


phpMyAdmin
Recent Favorites
New
information_schema
mysql
performance_schema
phpmyadmin
student_management_syster
New
courses
Student
Server: localhost » Database: student_management_system » Table: courses
Browse Structure SQL Search Insert Export Import Privileges Operations Trac
Table structure Relation view
Check all With selected: Browse Change Drop Primary Unique Index Spatial Full
Add to central columns Remove from central columns

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	courses_id	int(11)			No	None			Change Drop More
2	courses_name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
3	credits	int(11)			No	None			Change Drop More

Create an enrollments Table (Relationship Between Students and Courses):

- Create a third table named enrollments to track which students are enrolled in which courses. This table should have the following columns:
    - enrollment_id (INT, Primary Key, Auto Increment)
    - student_id (INT, Foreign Key referencing students.student_id)
    - course_id (INT, Foreign Key referencing courses.course_id)
    - enrollment_date (DATE)
- Click **Save** to complete the table creation.


Server: localhost » Database: student_management_system » Table: enrollments

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	enrollments_id	int(11)			No	None			Change Drop More
2	student_id	int(11)			No	None			Change Drop More
3	course_id	int(11)			No	None			Change Drop More
4	enrollments_date	date			No	None			Change Drop More

Add 1 column(s) after enrollments_date Go

Reflection:

- Discuss how relationships are established between tables in a relational database. Why are **foreign keys** important in maintaining data integrity?

ANS

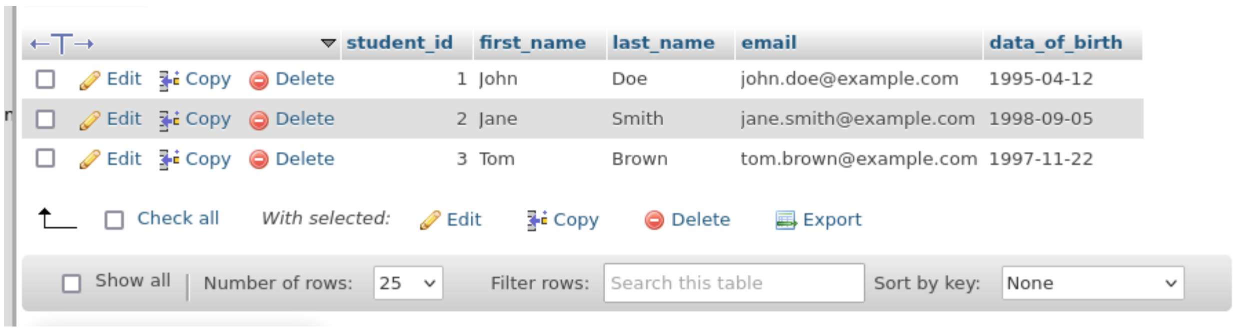
The relationship was established between tables using **primary key**, the primary key is unique identification that is common between two tables or more.

Foreign keys is important in maintaining data integrity, ensuring that the data link to the right owner of the data, for instance in the above table created student_id and course_id are foreign key that link the three table together and ensuring it link to the rightful owner of date in order to maintain data integrity without mix up the data.

Exercise 3: Inserting Data into the Tables

Insert Sample Data into the students Table:

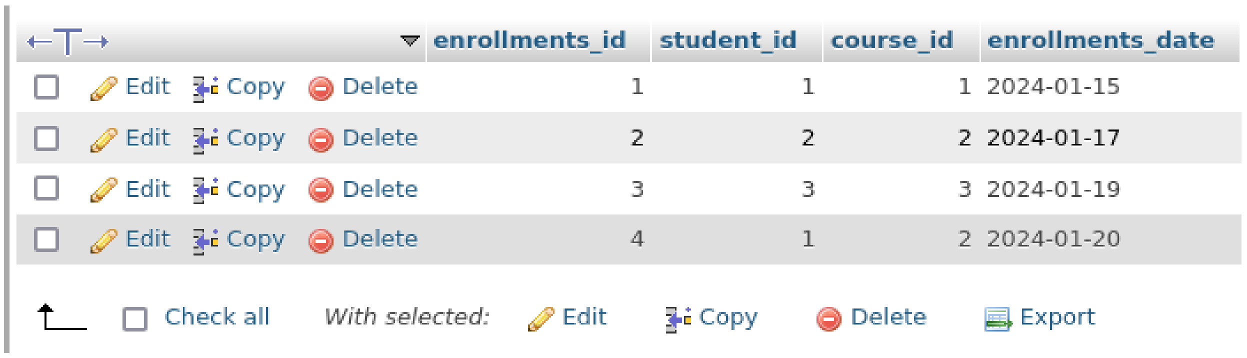
- Open the SQL tab in phpMyAdmin and execute the following SQL statement to insert some students:
- INSERT INTO students (first_name, last_name, email, date_of_birth)
- VALUES
- ('John', 'Doe', 'john.doe@example.com', '1995-04-12'),
- ('Jane', 'Smith', 'jane.smith@example.com', '1998-09-05'),
- ('Tom', 'Brown', 'tom.brown@example.com', '1997-11-22').



	student_id	first_name	last_name	email	data_of_birth
Edit Copy Delete	1	John	Doe	john.doe@example.com	1995-04-12
Edit Copy Delete	2	Jane	Smith	jane.smith@example.com	1998-09-05
Edit Copy Delete	3	Tom	Brown	tom.brown@example.com	1997-11-22

Insert Sample Data into the courses Table:

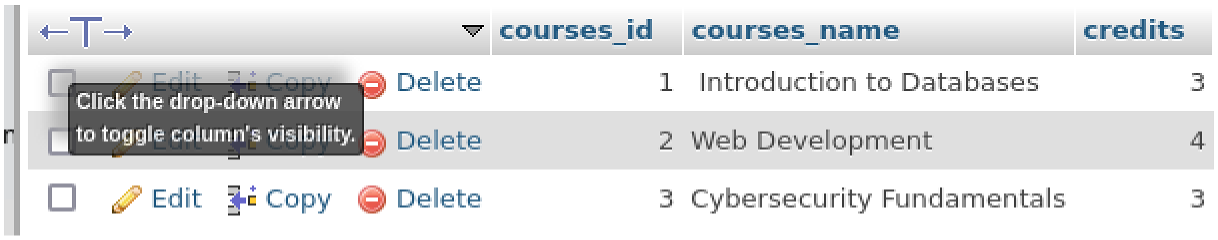
- Insert some sample courses using the following SQL statement:
- INSERT INTO courses (course_name, credits)
- VALUES
- ('Introduction to Databases', 3),
- ('Web Development', 4),
- ('Cybersecurity Fundamentals', 3).



	enrollments_id	student_id	course_id	enrollments_date
Edit Copy Delete	1	1	1	2024-01-15
Edit Copy Delete	2	2	2	2024-01-17
Edit Copy Delete	3	3	3	2024-01-19
Edit Copy Delete	4	1	2	2024-01-20

Insert Sample Data into the enrollments Table:

- Enroll the students into different courses using this SQL statement:
- INSERT INTO enrollments (student_id, course_id, enrollment_date)
- VALUES
- (1, 1, '2024-01-15'),
- (2, 2, '2024-01-17'),
- (3, 3, '2024-01-19'),
- (1, 2, '2024-01-20').



	courses_id	courses_name	credits
Edit Copy Delete	1	Introduction to Databases	3
Edit Copy Delete	2	Web Development	4
Edit Copy Delete	3	Cybersecurity Fundamentals	3

Reflection:

- Explain how inserting data into tables allows the database to become useful for managing information. Why is **data consistency** crucial when inserting related data across tables?

ANS

Inserting data into table allows databases to become useful for managing information because the data will be arranged in such a way that it will be easy to query while searching for information and provide unique identification for each data on the database.

Why is **data consistency** crucial when inserting related data across tables?

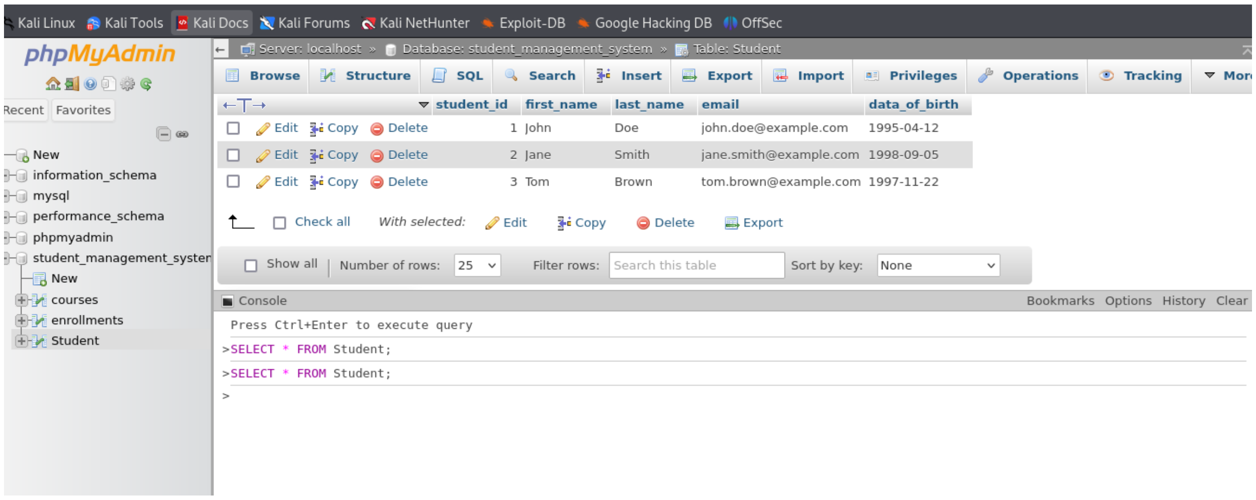
Data consistency is important to match one table to other related table if there is not consistency while inserting data there will mixed up while query the data or missed data.

Exercise 4: Querying the Database

Retrieve All Students:

- Use a SQL query to retrieve all records from the student's table:

SELECT * FROM students.



	student_id	first_name	last_name	email	data_of_birth
Edit Copy Delete	1	John	Doe	john.doe@example.com	1995-04-12
Edit Copy Delete	2	Jane	Smith	jane.smith@example.com	1998-09-05
Edit Copy Delete	3	Tom	Brown	tom.brown@example.com	1997-11-22

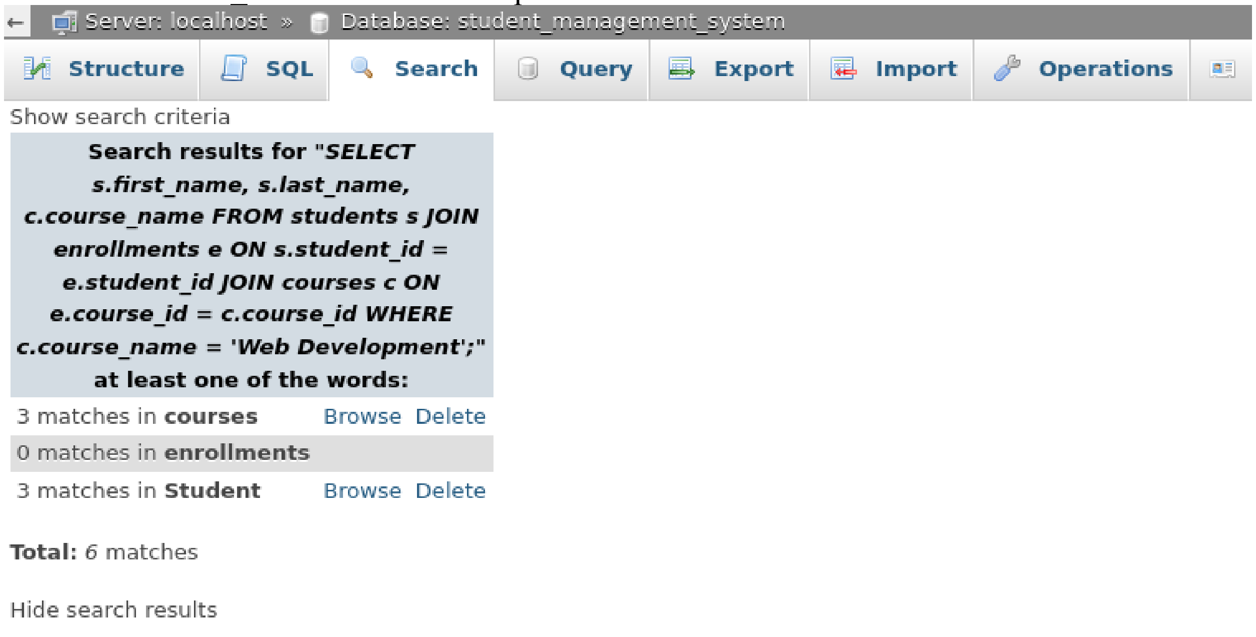
```
Press Ctrl+Enter to execute query
>SELECT * FROM Student;
>SELECT * FROM Student;
>
```

Retrieve Students Enrolled in a Specific Course:

- Write a SQL query to retrieve students enrolled in "Web Development":
- SELECT s.first_name, s.last_name, c.course_name
- FROM students s

- JOIN enrollments e ON s.student_id = e.student_id
- JOIN courses c ON e.course_id = c.course_id

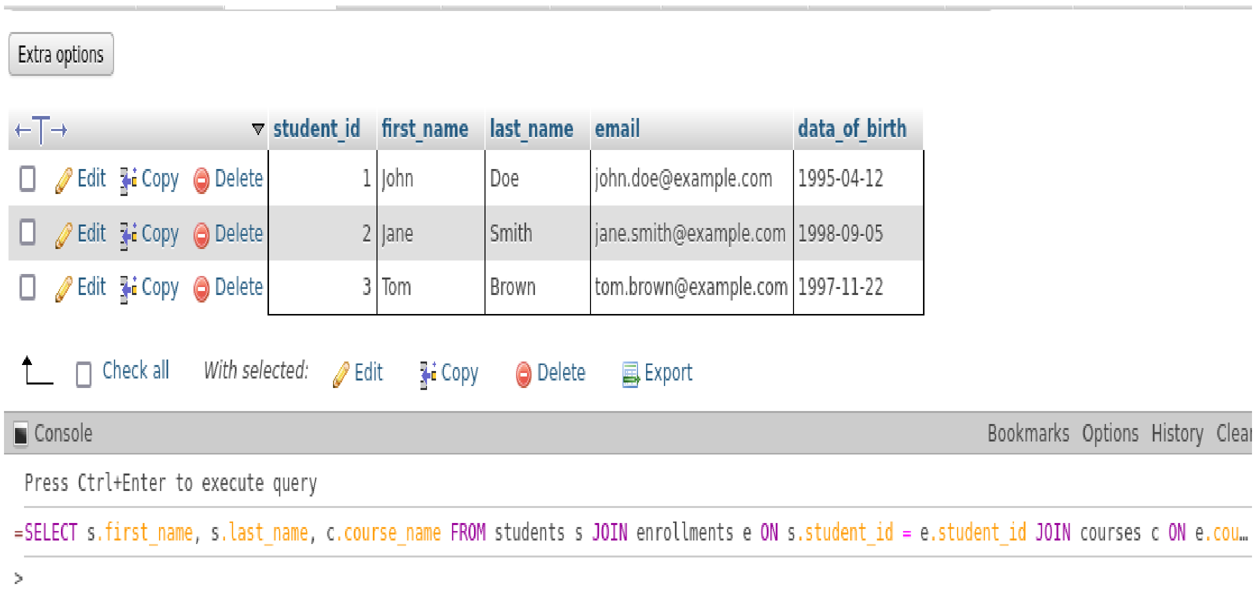
WHERE c.course_name = 'Web Development'



Search results for "*SELECT s.first_name, s.last_name, c.course_name FROM students s JOIN enrollments e ON s.student_id = e.student_id JOIN courses c ON e.course_id = c.course_id WHERE c.course_name = 'Web Development';*" at least one of the words:

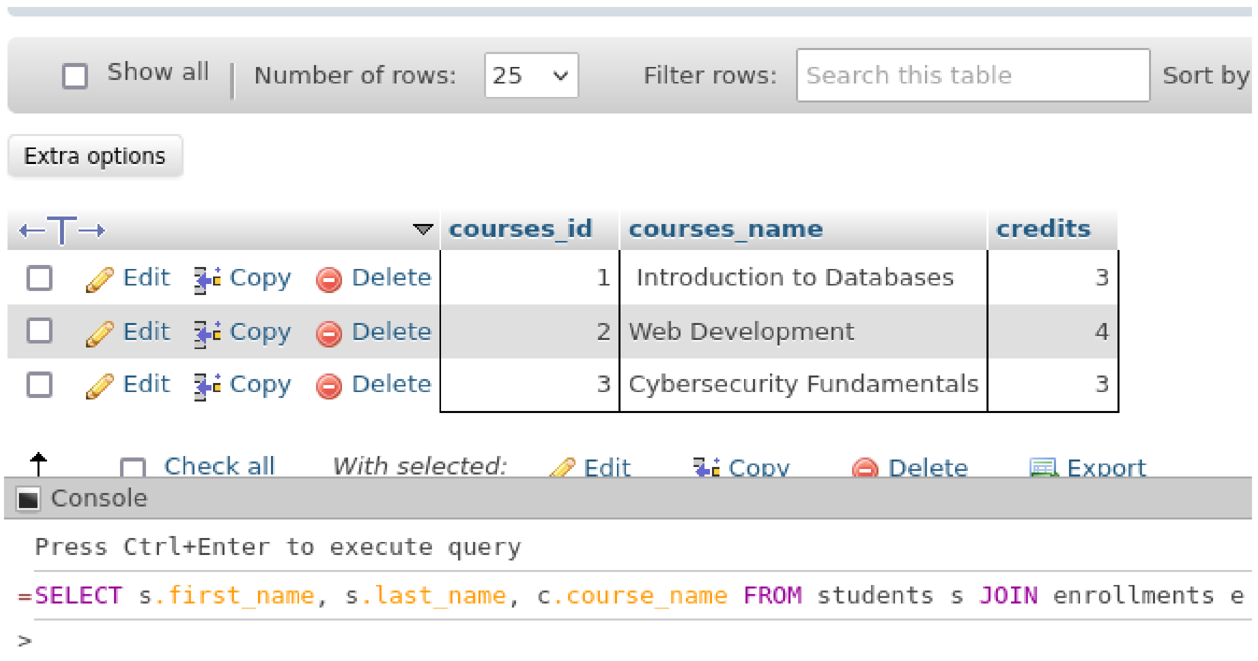
3 matches in **courses**	Browse Delete
0 matches in **enrollments**	
3 matches in **Student**	Browse Delete

Total: *6* matches



	student_id	first_name	last_name	email	data_of_birth
Edit Copy Delete	1	John	Doe	john.doe@example.com	1995-04-12
Edit Copy Delete	2	Jane	Smith	jane.smith@example.com	1998-09-05
Edit Copy Delete	3	Tom	Brown	tom.brown@example.com	1997-11-22

```
Press Ctrl+Enter to execute query
=SELECT s.first_name, s.last_name, c.course_name FROM students s JOIN enrollments e ON s.student_id = e.student_id JOIN courses c ON e.cou…
>
```


Show all
Number of rows: 25
Filter rows: Search this table
Sort by
Extra options

	courses_id	courses_name	credits
Edit Copy Delete	1	Introduction to Databases	3
Edit Copy Delete	2	Web Development	4
Edit Copy Delete	3	Cybersecurity Fundamentals	3

Check all
With selected: Edit Copy Delete Export
Console

```
Press Ctrl+Enter to execute query
=SELECT s.first_name, s.last_name, c.course_name FROM students s JOIN enrollments e
>
```

Update a Student's Email Address:

- Update the email address for John Doe using the following SQL statement:
- UPDATE students
- SET email = 'john.newemail@example.com'

WHERE first_name = 'John' AND last_name = 'Doe';

Delete an Enrollment Record:

- Delete the enrollment record for Jane Smith from the enrollments table:
- DELETE FROM enrollments

WHERE student_id = 2 AND course_id = 2;

Reflection:

- Discuss how SQL allows you to query and manipulate data in a relational database. How do **JOINs** facilitate retrieving related information from multiple tables?

ANS

SQL allows us to manipulate and query data in the database by using primary key or foreign key, which is common identity that relates two or more table together, the relational database can also be query by using SQL structure statement such as WHERE, FROM, SELECT, JOIN, AND, OR etc.

JOINs facilitate retrieve of related data by checking the relationship between two tables. This can be done in different ways such as:

INNER JOIN: this will join only what common or overlap to the two table

OUTER JOIN: this will join what is not common to the two tables

FULL JOIN: this will join two tables irrespective of what common to bot.

LEFT AND RIGHT JOIN: this will only consider the left table and right will consider only the right table to join the table.
