THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SOFTWARE TECHNOLOGY

PROJECT TITLE:

SECURITY FOR E-LEARNING WEBSITE

Instructor: **NGUYEN THE XUAN LY M.S**
Student: **BUI QUOC HUY**
Student ID: **102190315**
Class**: 19TCLC_Nhat1**

Da Nang, 1/2025

Bui Quoc Huy PROJECT' S NAME : SECURITY FOR E-LEARNING WEBSITE 2025

THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SOFTWARE TECHNOLOGY

PROJECT TITLE:

SECURITY FOR E-LEARNING WEBSITE

Instructor: **NGUYEN THE XUAN LY M.S**
Student: **BUI QUOC HUY**
Student ID: **102190315**
Class**: 19TCLC_Nhat1**

Da Nang, 1/2025

SUMMARY

Topic title: Security for e-learning website

Student name: Bui Quoc Huy

Student ID: 102190315 Class: 19TCLC_Nhat1

This project aimed to enhance Moodle’s functionality and security by integrating OpenAI’s API for automated quiz generation and addressing critical vulnerabilities, including Stored XSS, Open Redirect, and CSRF. Through simulated attacks and the implementation of robust fixes, the project demonstrated the interplay between innovation and security in educational platforms. By combining modern features with stringent security measures, this project serves as a valuable resource for improving Moodle’s reliability and usability.

DA NANG UNIVERSITY
UNIVERSITY OF SICIENCE AND TECHNOLOGY
FALCUTY
INFORMATION TECHNOLOGY

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name: Bui Quoc Huy Student ID : 102190315

Class: 19TCLC_Nhat1 Faculty: Information Technology Major: Cyber Security

1. *Topic title:*

Security for E-Learning Website

2. *Project topic : ☐has signed intellectual property agreement for final result*
3. *Initial figure and data:*

…...………………………………………………………………………………
…...………………………………………………………………………………
…...………………………………………………………………………………
…...………………………………………………………………………………

Content of the explanations and calculations:

- Introduction
- Overview of Moodle and common CVEss
- Define requirements and design feature
  Development and result evaluation
- Inclusion
  Reference

4. *Drawings, charts (specify the types and sizes of drawings):*

…...……………………………………………………………………………………
…...……………………………………………………………………………………
…...……………………………………………………………………………………
…...……………………………………………………………………………………

5. *Name of instructor:*	*Content parts:*
Nguyen The Xuan Ly M.S	

6. *Date of assignment :* ……../……./202…..
7. *Date of completion :* ……../……./202…..

Đà Nẵng, date month year 202

Head of Division…………………… **Instructor**

PREFACE AND ACKNOWLEDGEMENT

The successful completion of this thesis has been made possible through the guidance, support, and encouragement of numerous individuals and organizations. With heartfelt gratitude and utmost respect, I extend my sincere thanks to all who have contributed to my academic and research journey.

First and foremost, I am deeply grateful to the dedicated faculty members of the Faculty of Information Technology at Da Nang University of Technology. Your unwavering commitment to teaching, mentorship, and nurturing students has been instrumental in shaping my knowledge and skills. Your guidance has been invaluable in helping me complete this thesis, titled *"Security for e-learning website."*

I am especially indebted to my supervisor, MSc. Nguyen The Xuan Ly, for his insightful guidance and persistent support throughout this research. His encouragement, constructive feedback, and expertise have been pivotal in enabling me to navigate challenges and successfully bring this work to completion.

I also extend my gratitude to the leadership and staff of Da Nang University of Technology, as well as the functional departments, for their continuous assistance and facilitation during my studies. Their contributions have provided me with the resources and environment necessary to pursue my academic endeavors.

Despite my best efforts, as a student with limited experience and time, this thesis may still contain shortcomings. I welcome constructive feedback and suggestions from my teachers and peers to help refine my understanding and improve my future work.

Finally, I wish all the educators and staff of Da Nang University good health, inspiration, and continued success in their noble mission of shaping future generations through education.

ASSURRANCE

I hereby affirm the following:

1. The content of this senior project has been independently conducted and developed by myself under the mentorship and guidance of my supervisor, MSc. Nguyen The Xuan Ly.
2. All references, sources, and materials utilized in this thesis have been appropriately cited, with full acknowledgment of the original authors, including the title, publication details, and context.
3. I take full responsibility for the integrity and authenticity of this thesis. Should any instances of plagiarism or dishonesty be identified, I accept all consequences and accountability.

This declaration is made with complete honesty and transparency to uphold the principles of academic integrity and ethics.

Student Performed

TABLE OF CONTENT

{ }

{ }

LIST OF TABLES, PICTURES

LIST OF SYMBOL, ACRONYM

SYMBOL:

…………..

…………..

…………..

…………..

…………..

…………..

ACRONYM:

…………..

…………..

…………..

…………..

INTRODUCTION

Moodle is one of the most widely used learning management systems (LMS) worldwide, serving educational institutions and corporate training environments. However, its popularity and open-source nature make it a prime target for malicious attacks. The primary motivation behind this project is to address Moodle’s security vulnerabilities and demonstrate ways to improve its robustness against cyber threats. By simulating vulnerabilities and applying fixes, this project highlights both the risks and the solutions, ultimately aiming to secure educational platforms and safeguard sensitive information.

The integration of OpenAI APIs introduces a modern feature that enhances Moodle’s utility by enabling automated quiz generation. This feature not only streamlines the teaching process but also demonstrates how advanced technologies can coexist with strong security protocols. The dual focus of this project—feature enhancement and security hardening—ensures a comprehensive understanding of the interplay between functionality and security in modern web applications.

Chapter 1: OVERVIEW OF MOODLE AND CVE VULNERABILITIES

1.1. Moodle setup

1.1.1. *Frontend Technologies:*

Breakdown of programming languages and frameworks used in Moodle for its frontend

HTML5:

- Used for structuring the Moodle interface and providing semantic content.

CSS3:

- Provides styling for themes, ensuring a responsive and visually appealing user interface.
- Supports frameworks like Bootstrap for consistent design and responsiveness.

JavaScript:

- Adds interactivity to the Moodle user interface.
- Uses jQuery and AMD modules for client-side scripting.

Figure 1.1. Frontend familiar Technologies

Mustache Templates:

- Moodle uses Mustache for rendering frontend views dynamically with data from the backend.

YUI Library (Legacy):

- Some parts of Moodle still use Yahoo User Interface (YUI) for frontend components, although newer versions are transitioning away from it.

AJAX:

- Enables dynamic content updates without full-page reloads (e.g., for grading or loading quiz questions).

1.1.2. Backend Technologies

PHP:

- Primary programming language used to develop Moodle's core application logic.
- Supports modular architecture for plugins and custom development.

Moodle Framework:

- Moodle has its own framework, built in PHP, which provides APIs for managing users, courses, quizzes, and other functionality.

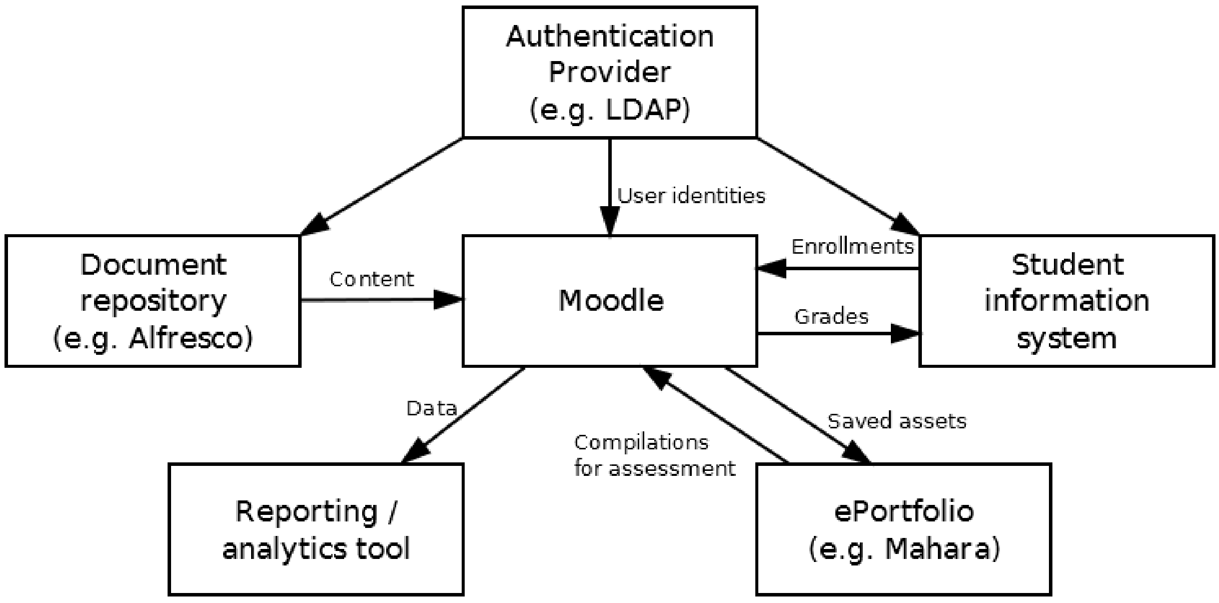

Authentication Provider (e.g. LDAP)
User identities
Document repository (e.g. Alfresco)
Content
Moodle
Enrollments
Grades
Student information system
Data
Compilations for assessment
Saved assets
Reporting / analytics tool
ePortfolio (e.g. Mahara)

Figure 1.2. Diagram of Moodle Architecture

CRON Jobs:

- Moodle relies on PHP scripts executed by CRON jobs to handle background tasks like sending emails, grading, and data cleanup.

1.1.3. Database Technologies

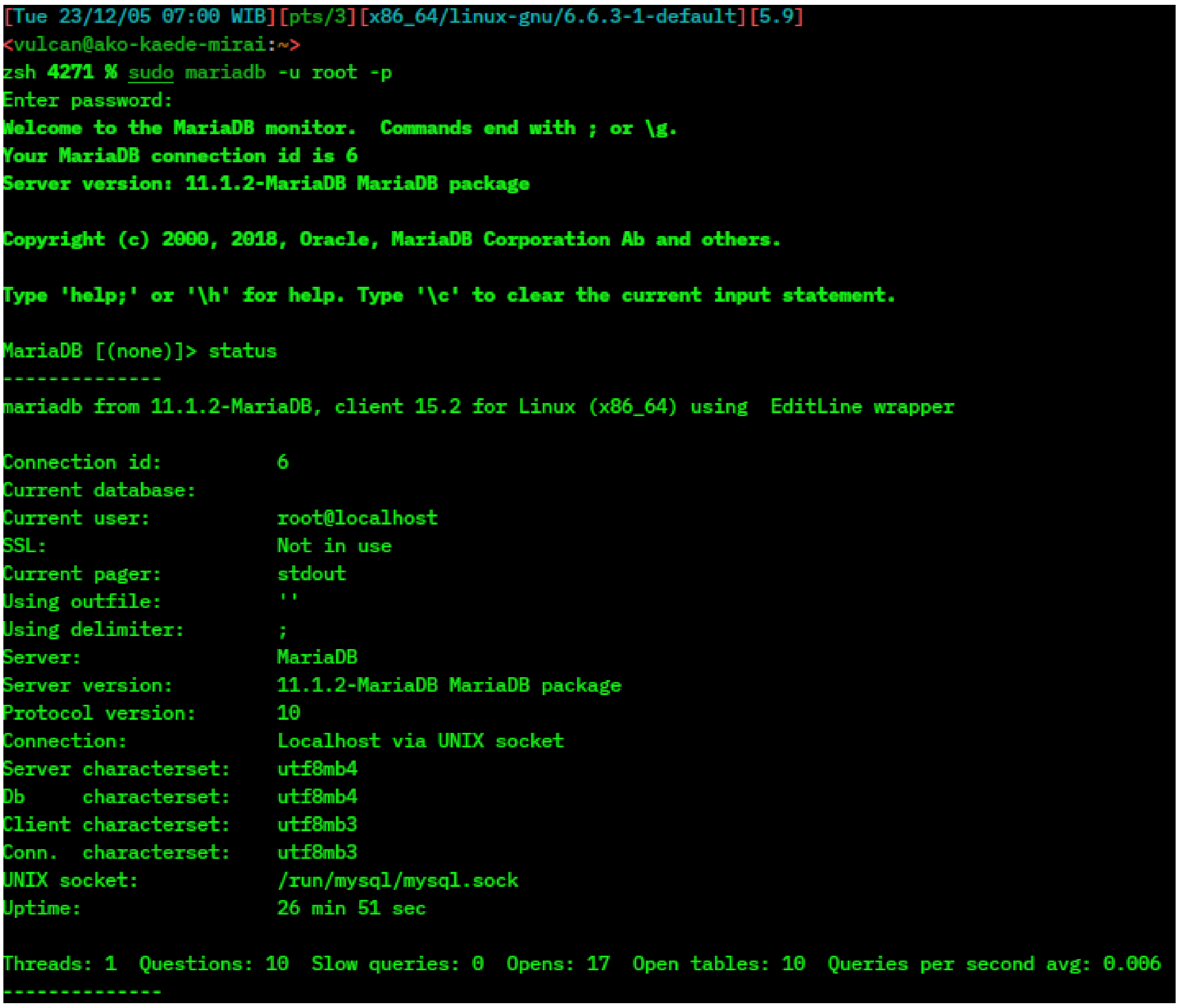
MySQL:

- The most common relational database used with Moodle. It stores user data, course content, logs, and system configurations.
- Reliable, scalable, and well-documented.

Figure 1.3. MariaDB and MySQL

MariaDB:

A fork of MySQL, often used as a drop-in replacement with performance improvements.



```
[Tue 23/12/05 07:00 WIB][pts/3][x86_64/linux-gnu/6.6.3-1-default][5.9]
<vulcan@ako-kaede-mirai:~>
zsh 4271 % sudo mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 6
Server version: 11.1.2-MariaDB MariaDB package

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> status
--------------
mariadb from 11.1.2-MariaDB, client 15.2 for Linux (x86_64) using  EditLine wrapper

Connection id:          6
Current database:
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server:                 MariaDB
Server version:         11.1.2-MariaDB MariaDB package
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    utf8mb4
Db     characterset:    utf8mb4
Client characterset:    utf8mb3
Conn.  characterset:    utf8mb3
UNIX socket:            /run/mysql/mysql.sock
Uptime:                 26 min 51 sec

Threads: 1  Questions: 10  Slow queries: 0  Opens: 17  Open tables: 10  Queries per second avg: 0.006
--------------
```

Figure 1.4. MariaDB set up

For this project I chose MariaDB because in my opinion MariaDB is often better than MySQL for large data sets, performance-critical workloads, and complex queries. MariaDB is a free, open-source clone of MySQL that was created by MySQL's original developers.

Features:

- Scalability: MariaDB is more scalable than MySQL.
- Query speed: MariaDB has faster query speeds than MySQL.
- Storage engines: MariaDB supports row and columnar storage, while MySQL focuses on its InnoDB storage engine.
- Regular expressions: MariaDB uses Perl-compatible regular expressions (PCRE), which are more powerful than MySQL's regex support.

- Thread pool: MariaDB's thread pool mechanism allows it to handle up to 200,000 simultaneous connections.
- KILL command: MariaDB's KILL command can kill all queries for a user or a query ID.

Use cases:

- Large data sets: MariaDB is good for managing large-sized data.
- Performance-critical workloads: MariaDB is a strong choice for performance-critical workloads.
- Complex queries: MariaDB is better at optimizing questions.
- Big, fast applications: MariaDB is a good choice for big, fast applications.

Ease of use:

- MariaDB is easy to install.
- Switching from MySQL to MariaDB is a simple task.

PostgreSQL:

- An alternative relational database supported by Moodle, known for advanced features, scalability, and performance.

Microsoft SQL Server:

- Supported in enterprise environments where Microsoft-based solutions are preferred.

Oracle Database:

- Another supported database, often used in large-scale enterprise setups requiring high performance and security.

1.1.4. Tools support web security

Server-Side Security Measures

- PHP Security Best Practices:
  - Enforcing safe_mode and restricting functions like exec(), system(), etc.
  - Using prepared statements and parameterized queries to prevent SQL injection.
  - Disabling allow_url_include and allow_url_fopen in php.ini.
- Database Security:
  - Support for secure database connections using SSL.
  - Role-based database access to restrict user privileges.

- HTTPS (SSL/TLS Encryption):
    - Ensures secure communication between the server and clients by encrypting all traffic.
    - Uses tools like Let's Encrypt for obtaining free SSL certificates.

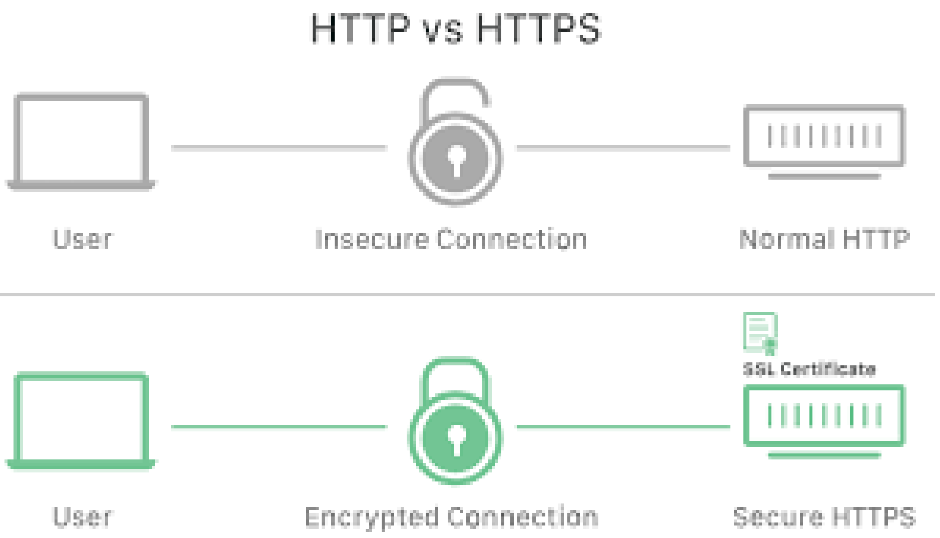


Figure 1.5. HTTPS gives better security

- File Upload Security:
    - Validation of uploaded files to ensure only allowed formats are accepted.
    - Storing files outside the web root to prevent direct access.
    - Using Moodle's File API for secure file management.

Authentication and Authorization

- Password Hashing:
    - Moodle uses secure algorithms like bcrypt for storing passwords.
- Authentication Plugins:
    - Supports OAuth2, LDAP, SAML, and other secure authentication protocols.
    - Integration with multi-factor authentication (MFA) for added security.
- Role and Capability Management:
    - Granular control over user permissions using Moodle's Role and Capability framework.
    - Restricts unauthorized access to sensitive areas.

Built-in Security Features in Moodle

- Input Validation:
    - Moodle's clean_param() function sanitizes user input to prevent injection attacks.
- Cross-Site Scripting (XSS) Prevention:
    - Output encoding with Moodle’s s() function ensures that scripts are not executed unintentionally.
- Cross-Site Request Forgery (CSRF) Tokens:
    - The sesskey parameter and require_sesskey() function help prevent CSRF attacks.
- Session Security:
    - Regenerates session IDs upon login to prevent session fixation.
    - Enforces secure cookies (HttpOnly and Secure flags).
- Captcha Support:
    - Adds CAPTCHA to login and form submissions to prevent automated attacks.

Security Tools for Monitoring and Patching

- Moodle Security Overview Report:
    - Provides a built-in dashboard for administrators to identify and address security issues.
- Updates and Patches:
    - Regular security updates and patches provided by the Moodle community.
    - Integrating tools like Git to automate updates.
- Logging and Auditing:
    - Built-in logging system records all actions for auditing purposes.
    - Integration with external monitoring tools like ELK Stack or Splunk for enhanced visibility.

Network and Infrastructure Security

- Firewall Configuration:
    - Using tools like iptables, UFW, or cloud-based firewalls to restrict unauthorized traffic.

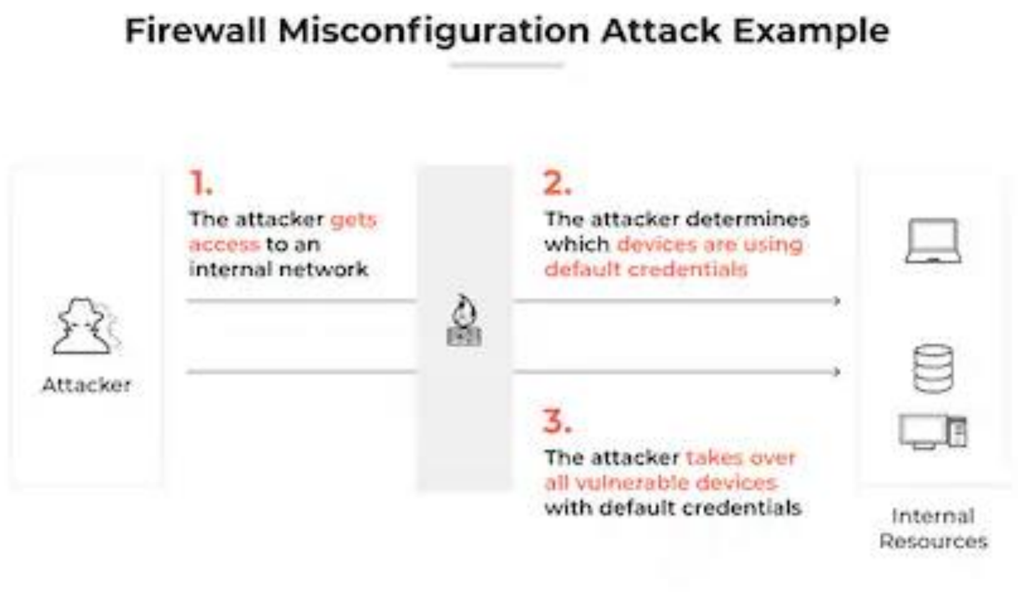

Firewall Misconfiguration Attack Example
Attacker
1. The attacker gets access to an internal network
2. The attacker determines which devices are using default credentials
3. The attacker takes over all vulnerable devices with default credentials
Internal Resources

Figure 1.6. Firewall Misconfiguration Attack

- Content Delivery Network (CDN):
    - Services like Cloudflare or AWS CloudFront to protect against DDoS attacks.
- Web Application Firewall (WAF):
    - Tools like ModSecurity to detect and block malicious traffic.
- Network Isolation:
    - Hosting Moodle on isolated virtual machines or containers (e.g., Docker) to contain potential breaches.

Vulnerability Scanning and Penetration Testing

- Automated Scanning Tools:
    - OWASP ZAP, Burp Suite, or Nessus to identify vulnerabilities like XSS, SQL injection, and CSRF.
- Code Review Tools:
    - Static code analysis tools like SonarQube to ensure secure coding practices.
- Penetration Testing:
    - Conduct regular penetration testing to simulate attacks like stored XSS, SQL injection, and authentication bypass.

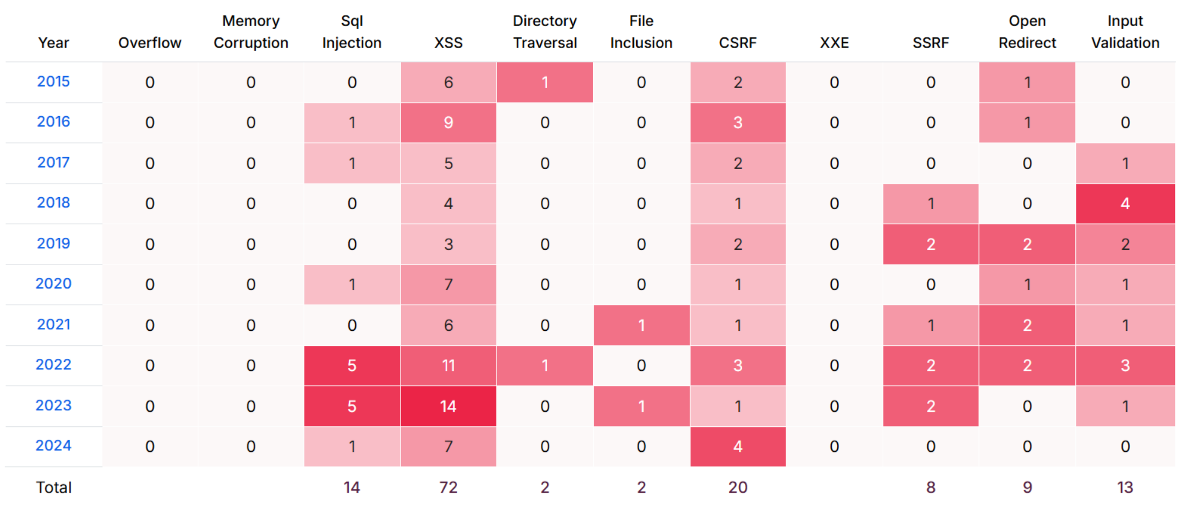
Secure Development Practices

- Framework Security:
  - Adhering to the OWASP Top 10 security practices during development.
  - Utilizing Moodle’s APIs for secure implementation of custom plugins or themes.
- Backup and Recovery:
  - Regular automated backups of the database and user files.
  - Encryption of backup files for additional security.

1.2. Common Moodle CVEs

1.2.1. Overall look

This table provides an analysis of vulnerabilities in Moodle by type and year (2015-2024), highlighting the number of reported vulnerabilities for specific categories such as SQL Injection, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), and others.

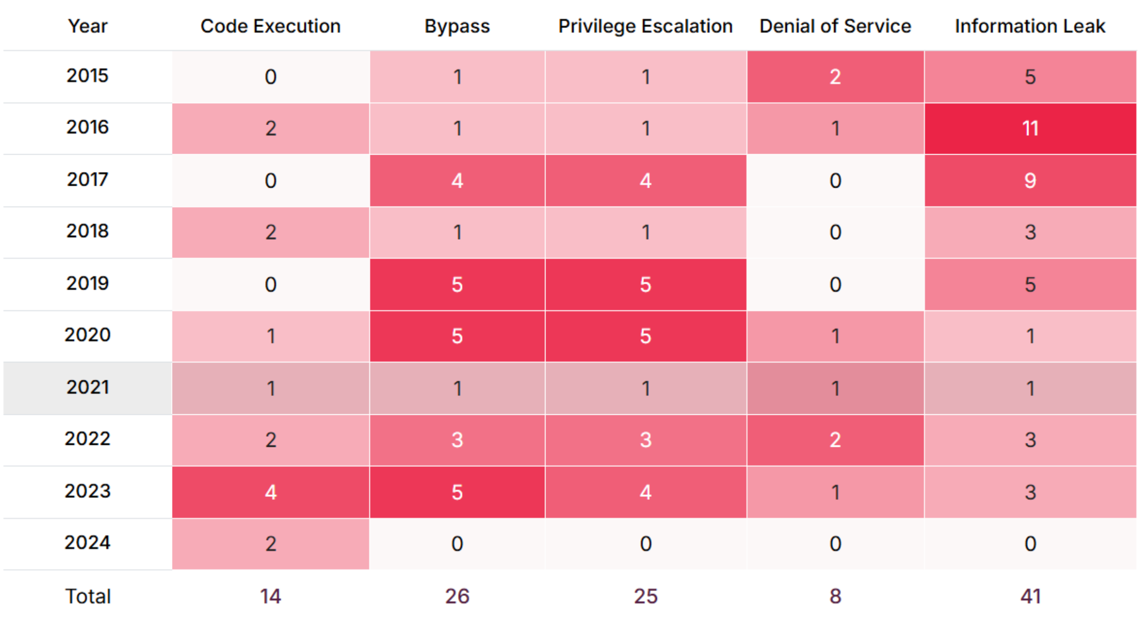


Year	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	File Inclusion	CSRF	XXE	SSRF	Open Redirect	Input Validation
2015	0	0	0	6	1	0	2	0	0	1	0
2016	0	0	1	9	0	0	3	0	0	1	0
2017	0	0	1	5	0	0	2	0	0	0	1
2018	0	0	0	4	0	0	1	0	1	0	4
2019	0	0	0	3	0	0	2	0	2	2	2
2020	0	0	1	7	0	0	1	0	0	1	1
2021	0	0	0	6	0	1	1	0	1	2	1
2022	0	0	5	11	1	0	3	0	2	2	3
2023	0	0	5	14	0	1	1	0	2	0	1
2024	0	0	1	7	0	0	4	0	0	0	0
Total			14	72	2	2	20		8	9	13

Figure 1.7 Vulnerabilities by categories

Cross-Site Scripting (XSS) emerges as the most frequent vulnerability across the years, peaking in 2023 with 14 occurrences. SQL Injection is comparatively rare but saw a spike in 2022 and 2023 with five occurrences each. Other categories, such as CSRF and Open Redirect, show occasional fluctuations, with CSRF reaching a peak of three in 2016 and 2022. Notably, Open Redirect remains consistently low, with occurrences ranging between zero and two annually.

Stored XSS is an increasingly critical issue within Moodle, exacerbated by its potential to persistently affect multiple users. Combined with input validation issues, this underscores the need for heightened attention to sanitizing and validating inputs. Directory Traversal and File Inclusion have minimal occurrences, indicating they are less of a concern for Moodle, but their implications still demand attention in security measures.



Year	Code Execution	Bypass	Privilege Escalation	Denial of Service	Information Leak
2015	0	1	1	2	5
2016	2	1	1	1	11
2017	0	4	4	0	9
2018	2	1	1	0	3
2019	0	5	5	0	5
2020	1	5	5	1	1
2021	1	1	1	1	1
2022	2	3	3	2	3
2023	4	5	4	1	3
2024	2	0	0	0	0
Total	14	26	25	8	41

Figure 1.8 Vulnerabilities by impact types

In terms of impact, Information Leak is a consistent concern, particularly in 2016 with 11 cases. Privilege Escalation and Code Execution also stand out as notable impacts, with a significant rise in Privilege Escalation incidents in 2019 and 2022, both recording five occurrences. Denial of Service (DoS), while not as frequent, had spikes in earlier years, particularly in 2015 and 2016, with two instances each. The data highlights a trend where Information Leak and Privilege Escalation vulnerabilities tend to occur in tandem, emphasizing the criticality of access control and secure data handling.

When comparing the two tables, there is a strong link between specific vulnerability types and their impacts. For example, XSS vulnerabilities directly contribute to Information Leak and Privilege Escalation incidents. The steady rise in XSS vulnerabilities, especially in recent years, correlates with the increasing complexity of web applications and the growing attack surface in Moodle’s ecosystem. SQL Injection, while less frequent, poses a severe threat when exploited, often leading to Code Execution or Information Leak scenarios.

The period between 2019 and 2023 shows a rise in the dive’sity of vulnerability types and their associated impacts. This trend could be attributed to the increased adoption of Moodle in various contexts, leading to more frequent and diverse testing. However, the decline in certain categories such as Directory Traversal and File Inclusion suggests improvements in those areas, potentially due to robust security patches and development practices.

In conclusion, Moodle's security landscape shows both progress and persistent challenges. While certain vulnerabilities like Directory Traversal and File Inclusion have been largely mitigated, XSS and Information Leak remain dominant concerns. The upward trend in XSS vulnerabilities highlights the need for continued focus on secure input handling and validation mechanisms. By analyzing the link between vulnerability types and their impacts, Moodle developers and administrators can better prioritize their security efforts to minimize risks effectively.

1.2.2. Specific Vulnerabilities

Moodle-Specific Vulnerabilities

CVE-2022-45143 (Open Redirect Vulnerability)

- Definition:
  Open Redirect vulnerabilities occur when an application incorrectly processes URLs provided by users, allowing them to redirect victims to malicious websites.
- Cyberattack:
  An attacker embeds a malicious URL as a parameter (e.g., http://example.com/redirect?url=http://malicious.com). When the victim clicks the link, they are redirected to the malicious site.
- Consequences:

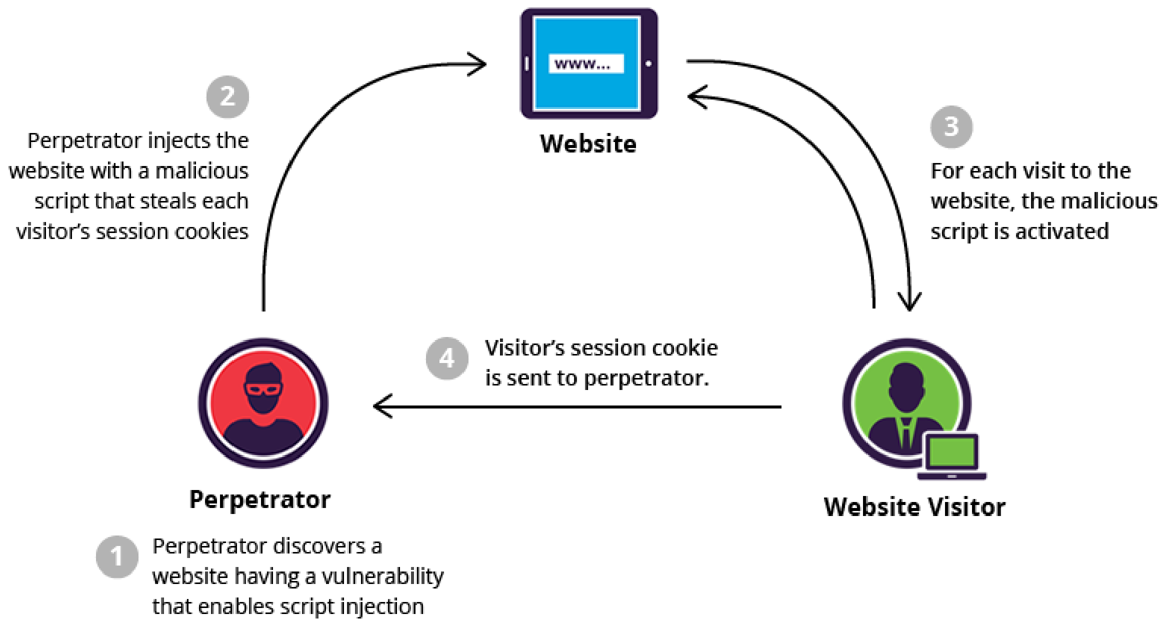
- Phishing attacks become easier, tricking users into giving sensitive information.
- Reduced trust in the Moodle platform.

- Prevention:

- Validate and sanitize all user-supplied URLs.
- Restrict redirects to trusted and whitelisted domains.

CVE-2021-40524 (Stored XSS)

- Definition:
  Stored XSS (Cross-Site Scripting) allows attackers to inject malicious scripts into Moodle's database, which execute when accessed by a user.
- Cyberattack:
  Attackers inject malicious JavaScript code into form fields (e.g., chat messages, forum posts). When another user views the content, the script executes in their browser.



www...
Website
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection

Figure 1.9. XSS Cyberattack

- Consequences:

- Session hijacking.
- Credential theft.
- Unauthorized actions (e.g., changing user roles).

- Prevention:

- Use Moodle's built-in clean_param() function to sanitize input.
- Encode all output using Moodle's s() function to prevent script execution.

CVE-2020-14318 (Authentication Bypass)

- Definition:
  This vulnerability allows attackers to bypass authentication mechanisms due to weak or mismanaged session tokens.
- Cyberattack:
  Attackers exploit poorly validated session tokens or replay a previously captured token to gain unauthorized access.
- Consequences:

- Unauthorized users can access sensitive user accounts or admin pages.
- Complete control of Moodle systems.

- Prevention:

- Use strong token validation methods.
- Enforce HTTPS to prevent token interception.
- Implement token expiration and refresh mechanisms.

CVE-2019-10193 (SQL Injection)

- Definition:
  SQL Injection occurs when attackers manipulate input fields to execute malicious SQL queries on the database.
- Cyberattack:
  An attacker inputs SQL commands into vulnerable fields (e.g., ' OR 1=1 --) to access or modify database records.

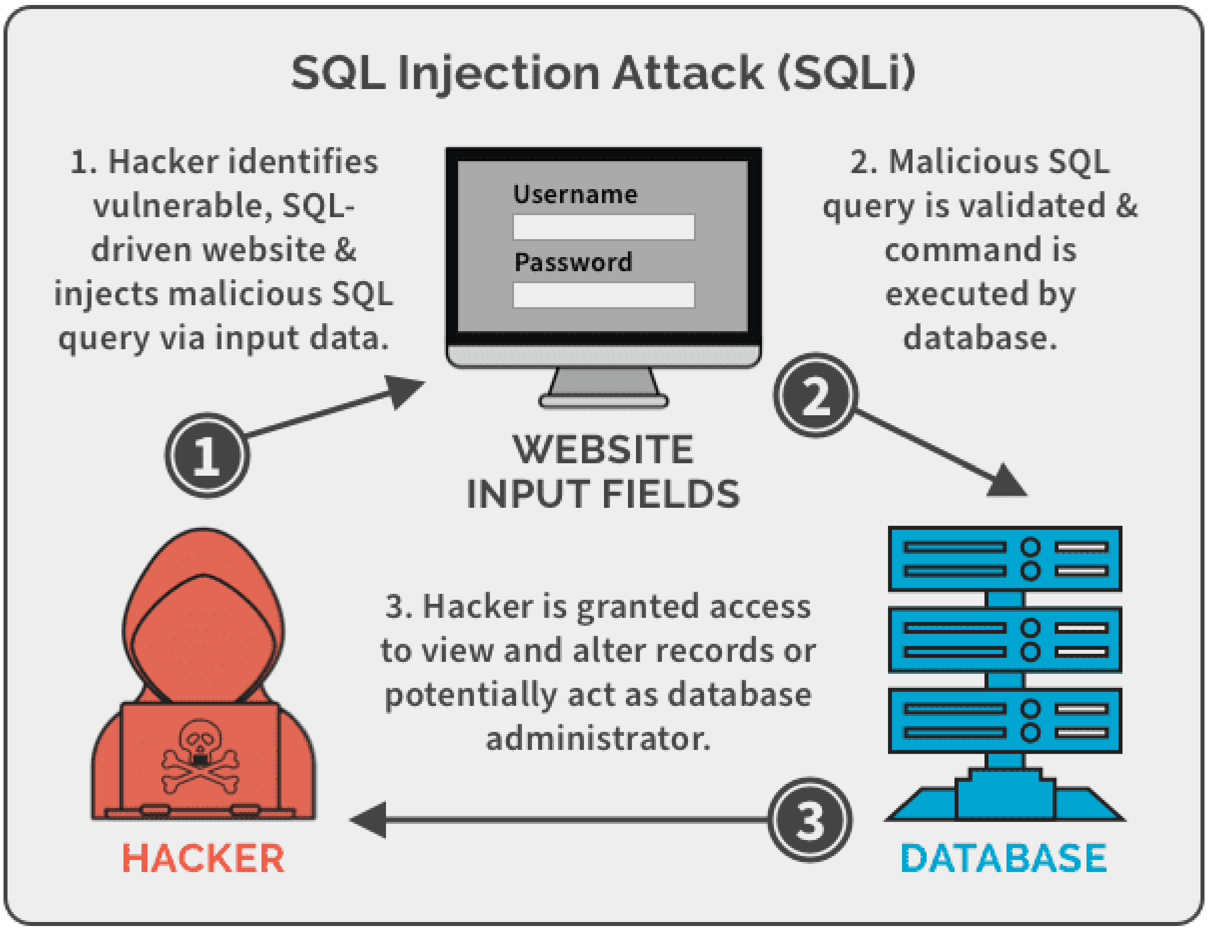

SQL Injection Attack (SQLi)
1. Hacker identifies vulnerable, SQL-driven website & injects malicious SQL query via input data.
Username
Password
2. Malicious SQL query is validated & command is executed by database.
1
2
WEBSITE INPUT FIELDS
3. Hacker is granted access to view and alter records or potentially act as database administrator.
3
HACKER
DATABASE

Figure 1.10. SQL Injection Attack

- Consequences:
- Unauthorized data access or modification.
- Potential deletion of database contents.

- Prevention:
  - Use prepared statements ($DB->get_records_sql()) and parameterized queries.
  - Avoid directly concatenating user input into SQL queries.

CVE-2018-1133 (Sensitive Information Disclosure)

- Definition:
  This vulnerability leaks sensitive information like database credentials or user data via debugging messages.
- Cyberattack:
  Attackers trigger errors or exceptions, revealing debug messages that expose internal system details.
- Consequences:
  - Compromise of database credentials or system configuration.
  - Further attacks (e.g., privilege escalation).
- Prevention:
  - Disable debugging in production ($CFG->debug = DEBUG_NONE).
  - Use custom error pages to prevent raw error outputs.

CVE-2017-2641 (Session Fixation)

- Definition:
  Session fixation allows attackers to set a user's session ID before the user authenticates, enabling them to hijack the session later.
- Cyberattack:
  Attackers craft a malicious link with a predefined session ID and trick users into logging in through it.
- Consequences:

  Attackers gain unauthorized access to user accounts.
- Prevention:
  - Regenerate session IDs upon login (session_regenerate_id() in PHP).
  - Use secure cookie flags (e.g., HttpOnly and Secure).

CVE-2016-3737 (Insufficient Access Control)

- Definition:
  Improper access control allows unauthorized users to access restricted areas of Moodle (e.g., admin pages).
- Cyberattack:
  Attackers manipulate URLs or session data to access admin-level features.

- Consequences:
  - Unauthorized modifications (e.g., adding/removing courses or users).
  - System compromise.
- Prevention:
  - Use Moodle's require_capability() function to enforce permissions.
  - Regularly audit roles and capabilities.

CVE-2015-2266 (File Upload Vulnerability)

- Definition:
  This vulnerability allows attackers to upload malicious files, such as PHP scripts, to the server.
- Cyberattack:
  An attacker uploads a malicious .php file through a vulnerable upload form and executes it.
- Consequences:
  - Remote code execution.
  - Complete control of the server.
- Prevention:
  - Restrict file types to safe formats (e.g., .jpg, .png).
  - Store uploaded files outside the web root.

CVE-2014-7833 (CSRF)

- Definition:
  CSRF (Cross-Site Request Forgery) tricks users into performing unwanted actions on Moodle while authenticated.
- Cyberattack:
  Attackers create a malicious website that sends requests to Moodle on behalf of the logged-in user.

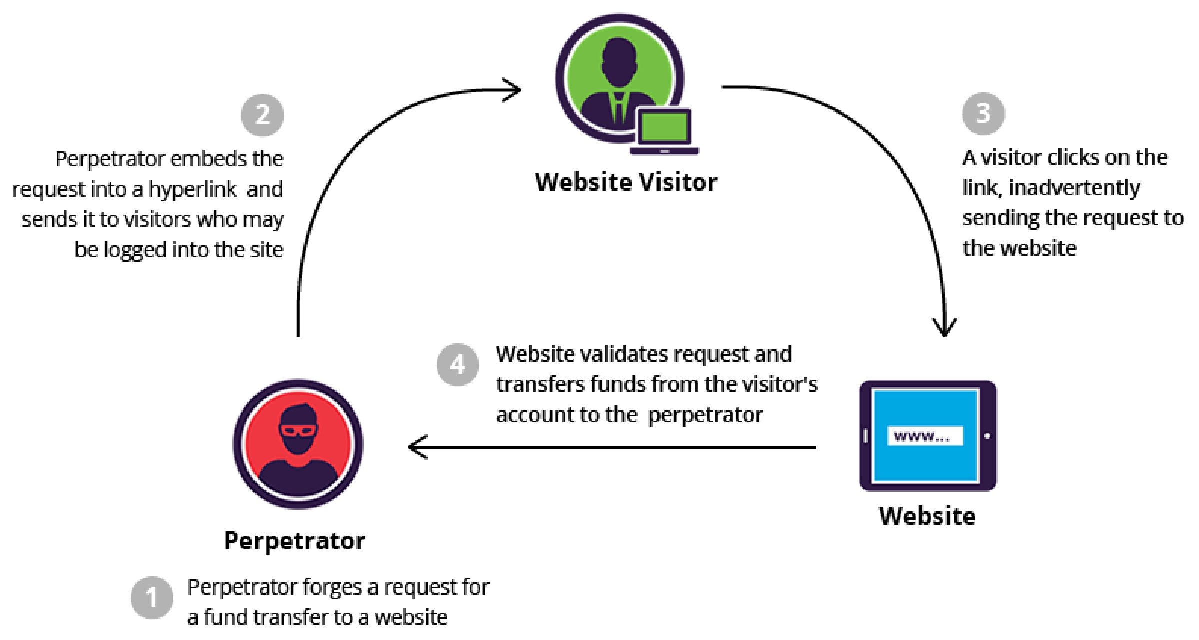

2
Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
Website Visitor
3
A visitor clicks on the link, inadvertently sending the request to the website
4
Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1
Perpetrator forges a request for a fund transfer to a website

Figure 1.11. CSRF Attack

- Consequences:

- Unintended actions (e.g., changing user roles or deleting content).
- Compromised system integrity.

- Prevention:

- Use Moodle's built-in CSRF tokens (sesskey parameter).
- Validate requests using require_sesskey().

PHP-Related Vulnerabilities

CVE-2012-3394 (Arbitrary File Inclusion)

- Definition:
  Improper use of PHP functions (include(), require()) allows attackers to include unauthorized files.
- Cyberattack:
  Attackers manipulate file paths in the URL or form fields to include files like /etc/passwd.
- Consequences:

- Exposure of sensitive server files.
- Execution of malicious code.

- Prevention:

- Validate and sanitize file paths.
- Use realpath() to resolve file paths and restrict to specific directories.

Misconfigured PHP File Uploads (Linked to CVE-2015-2266)

- Definition:
  Improper PHP configuration allows malicious file uploads to execute.
- Cyberattack:
  Upload a .php file as a script through a vulnerable Moodle form.
- Consequences:

- Remote code execution.
- Server takeover.

- Prevention:

- Set file_uploads = Off in php.ini if file uploads aren't required.
- Use move_uploaded_file() to store files in a safe location.

Chapter 2: DEFINE REQUIREMENTS AND DESIGN FEATURES

2.1. Project Requirements:

2.1.1. Security requirements:

- In this project, identifying and exploiting vulnerabilities in a controlled environment was paramount to understanding Moodle's security landscape. The controlled environment simulated realistic attack scenarios, such as SQL Injection, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), and privilege escalation, allowing for a comprehensive analysis of the platform's weaknesses. By safely mimicking potential threats, I ensured that no real harm or data compromise occurred during testing, preserving the integrity of the project and the environment. This approach facilitated an in-depth study of the root causes of vulnerabilities, such as insufficient input validation or insecure communication protocols..
- Post-identification, the project shifted to implementing robust fixes aimed at neutralizing these vulnerabilities. For instance, applying strong input validation mechanisms helped sanitize data inputs to prevent malicious payloads. Token-based CSRF protections were deployed to secure user sessions from unauthorized actions, while encoding practices mitigated XSS risks by ensuring that web browsers interpreted user

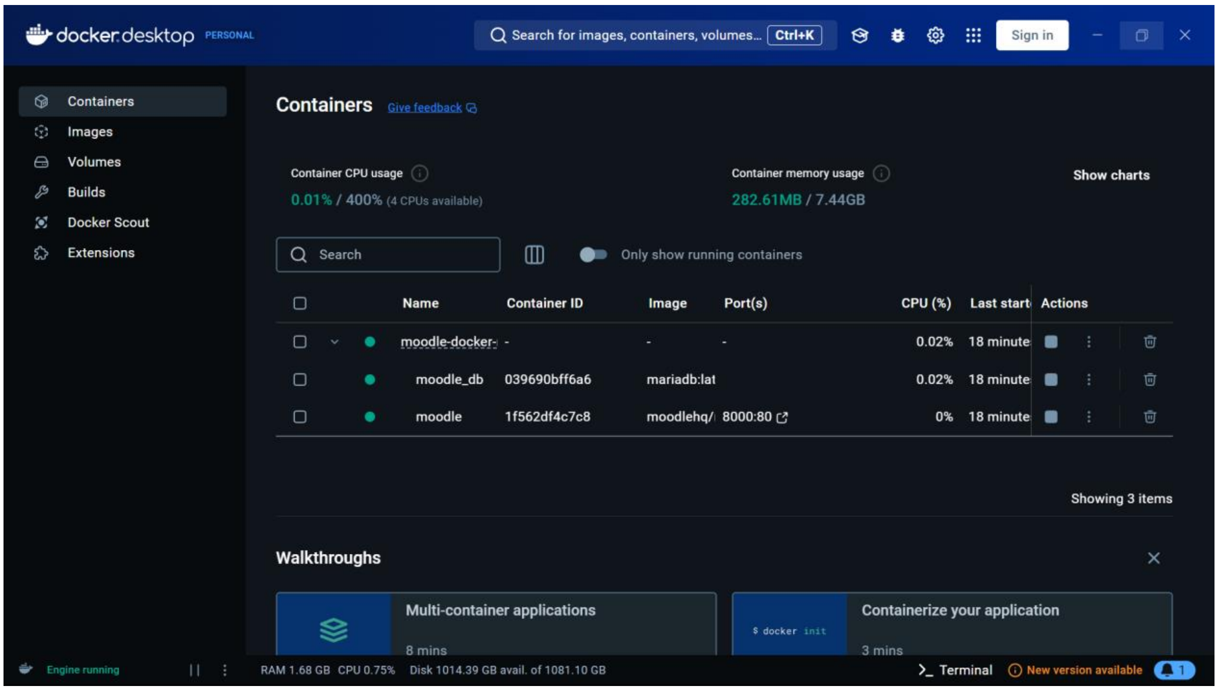
inputs safely. This cycle of exploitation and resolution not only fortified the Moodle platform against known vulnerabilities but also provided a replicable methodology for security testing in similar web applications.

2.1.2. Feature Requirements:

Integrating OpenAI APIs into Moodle necessitated striking a delicate balance between usability and security. The primary feature requirement was to empower teachers with an automated quiz-generation tool, leveraging OpenAI's advanced capabilities to create meaningful and customizable assessments. The integration was designed to ensure a seamless user experience, emphasizing simplicity in accessing and utilizing the feature while maintaining stringent security protocols. The project implemented secure API communication over HTTPS to prevent man-in-the-middle attacks, and API keys were handled securely within the plugin code to protect against unauthorized access. These measures were coupled with user-centric design principles to enable educators to interact effortlessly with the feature.

2.1.3. Development Requirements:

The project emphasized adhering to well-defined development requirements to maintain consistency, modularity, and scalability in its implementation. Docker was employed to create consistent development environments.



Containers

Container CPU usage
0.01% / 400% (4 CPUs available)

Container memory usage
282.61MB / 7.44GB

Name	Container ID	Image	Port(s)	CPU (%)	Last start	Actions
moodle-docker-	-	-	-	0.02%	18 minute	
moodle_db	039690bff6a6	mariadb:lat		0.02%	18 minute	
moodle	1f562df4c7c8	moodlehq/	8000:80	0%	18 minute	

Showing 3 items

RAM 1.68 GB CPU 0.75% Disk 1014.39 GB avail. of 1081.10 GB

Figure 2.1. Using Docker as container

The team also adopted PHP best practices, including using object-oriented programming and adhering to Moodle’s plugin development architecture, which is essential for integrating seamlessly into the existing ecosystem. The plugin was built as a modular and extendable component, adhering to Moodle's API guidelines to ensure compatibility with future platform updates. Moreover, security was deeply embedded into the development process by conducting regular code reviews and employing static analysis tools to identify vulnerabilities in the codebase. The development practices also included version control using Git, allowing for effective collaboration and tracking of changes, further ensuring the project’s stability and scalability. These practices, combined with Docker’s containerization, enabled a smooth and efficient development workflow, laying a strong foundation for the project’s success.

2.2. Degisn Features

2.2.1. Security Improvements

The project prioritized security improvements as a critical aspect of its design features, focusing on preventing common vulnerabilities while strengthening Moodle’s defenses against emerging threats. Input validation and encoding were implemented to mitigate XSS attacks, ensuring that user inputs were sanitized and processed safely before rendering in the browser. This prevented malicious scripts from being executed, safeguarding both the platform and its users. Token-based CSRF protection mechanisms were incorporated to enhance session security by requiring unique, session-bound tokens for sensitive user actions. This measure effectively prevented unauthorized requests from being executed on behalf of legitimate users

Together, these improvements elevated the security posture of the Moodle platform, making it more resilient to a wide range of attacks while maintaining user trust and system integrity.

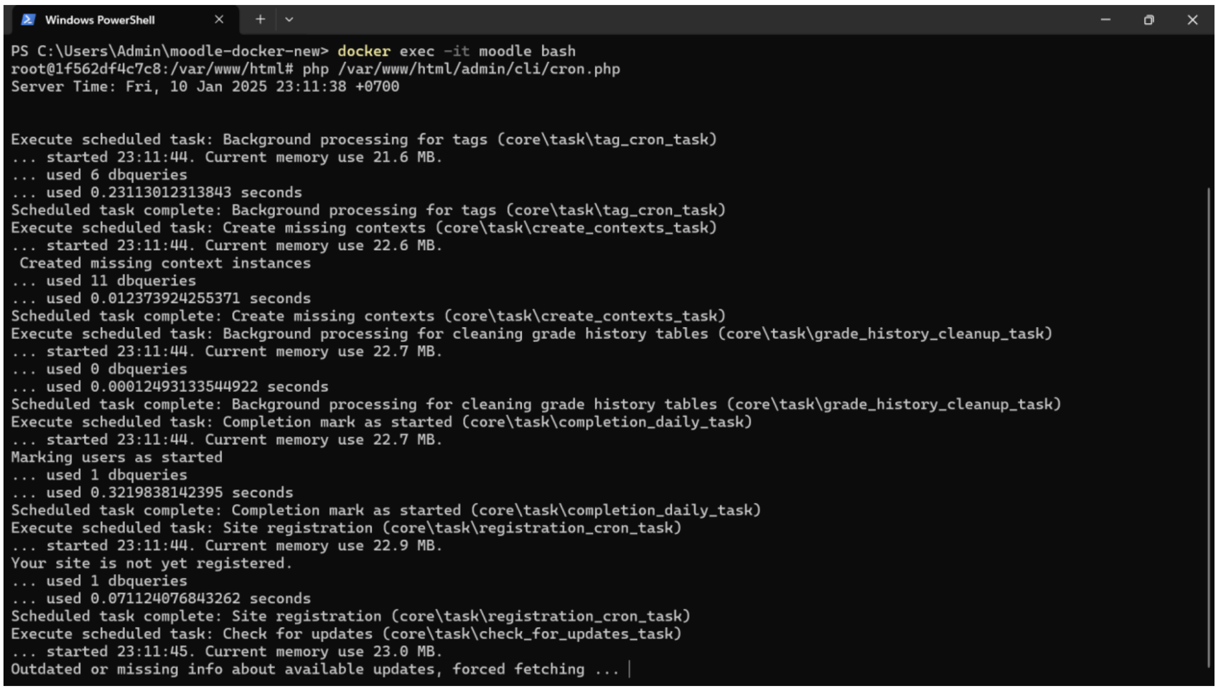
2.2.2. OpenAI Integration

The integration of OpenAI into Moodle represented a groundbreaking advancement in automating quiz generation. A custom plugin was developed using PHP, adhering to Moodle’s plugin architecture to ensure seamless integration into the platform. This plugin featured an intuitive user interface designed specifically for educators, enabling them to define quiz parameters such as topics, difficulty levels, and question types. The plugin communicated securely with the OpenAI API via HTTPS, ensuring that data exchanges remained encrypted and protected from interception.

2.2.3. Monitoring and Automation

Monitoring and automation were integral design features aimed at maintaining the project’s efficiency and security over time. Automated cron jobs were configured to handle repetitive tasks such as scheduled quiz generation and vulnerability scanning.

These jobs ensured that the system remained proactive in identifying and addressing potential issues, reducing manual overhead and the risk of oversight.

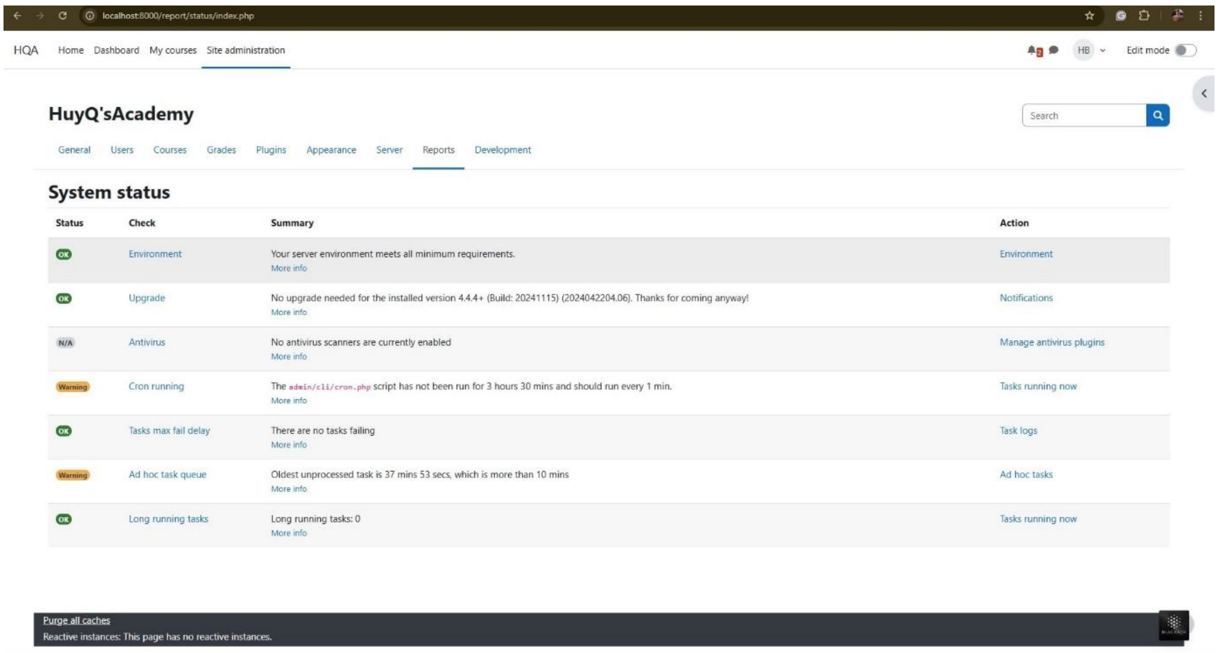


```
PS C:\Users\Admin\moodle-docker-new> docker exec -it moodle bash
root@1f562df4c7c8:/var/www/html# php /var/www/html/admin/cli/cron.php
Server Time: Fri, 10 Jan 2025 23:11:38 +0700


Execute scheduled task: Background processing for tags (core\task\tag_cron_task)
... started 23:11:44. Current memory use 21.6 MB.
... used 6 dbqueries
... used 0.23113012313843 seconds
Scheduled task complete: Background processing for tags (core\task\tag_cron_task)
Execute scheduled task: Create missing contexts (core\task\create_contexts_task)
... started 23:11:44. Current memory use 22.6 MB.
 Created missing context instances
... used 11 dbqueries
... used 0.012373924255371 seconds
Scheduled task complete: Create missing contexts (core\task\create_contexts_task)
Execute scheduled task: Background processing for cleaning grade history tables (core\task\grade_history_cleanup_task)
... started 23:11:44. Current memory use 22.7 MB.
... used 0 dbqueries
... used 0.00012493133544922 seconds
Scheduled task complete: Background processing for cleaning grade history tables (core\task\grade_history_cleanup_task)
Execute scheduled task: Completion mark as started (core\task\completion_daily_task)
... started 23:11:44. Current memory use 22.7 MB.
Marking users as started
... used 1 dbqueries
... used 0.3219838142395 seconds
Scheduled task complete: Completion mark as started (core\task\completion_daily_task)
Execute scheduled task: Site registration (core\task\registration_cron_task)
... started 23:11:44. Current memory use 22.9 MB.
Your site is not yet registered.
... used 1 dbqueries
... used 0.071124076843262 seconds
Scheduled task complete: Site registration (core\task\registration_cron_task)
Execute scheduled task: Check for updates (core\task\check_for_updates_task)
... started 23:11:45. Current memory use 23.0 MB.
Outdated or missing info about available updates, forced fetching ...
```

Figure 2.2. Using Cron for automatically handling tasks

For instance, regular vulnerability scans were automated to identify any emerging security threats, ensuring that the platform remained compliant with best practices. Additionally, the automated quiz generation system utilized pre-defined schedules to create quizzes, allowing educators to plan assessments well in advance without manual intervention. This level of automation enhanced the platform’s usability while ensuring that security monitoring remained continuous and effective. By embedding automation into the design, the project achieved a balance between operational efficiency and robust security, laying the groundwork for sustainable use and further development.



HuyQ'sAcademy

System status

Status	Check	Summary	Action
OK	Environment	Your server environment meets all minimum requirements. More info	Environment
OK	Upgrade	No upgrade needed for the installed version 4.4.4+ (Build: 20241115) (2024042204.06). Thanks for coming anyway! More info	Notifications
N/A	Antivirus	No antivirus scanners are currently enabled More info	Manage antivirus plugins
Warning	Cron running	The `admin/cli/cron.php` script has not been run for 3 hours 30 mins and should run every 1 min. More info	Tasks running now
OK	Tasks max fail delay	There are no tasks failing More info	Task logs
Warning	Ad hoc task queue	Oldest unprocessed task is 37 mins 53 secs, which is more than 10 mins More info	Ad hoc tasks
OK	Long running tasks	Long running tasks: 0 More info	Tasks running now

Purge all caches
Reactive instances: This page has no reactive instances.

Figure 2.3. Cron has to be run all time

Chapter 3: OPENAI API INTEGRATION, SECURITY TESTING AND RESULTS EVALUATION

3.1. Quiz Generator Integration

3.1.1. Plugin Development

The following diagram shows the database structure for the quiz and question bank in Moodle

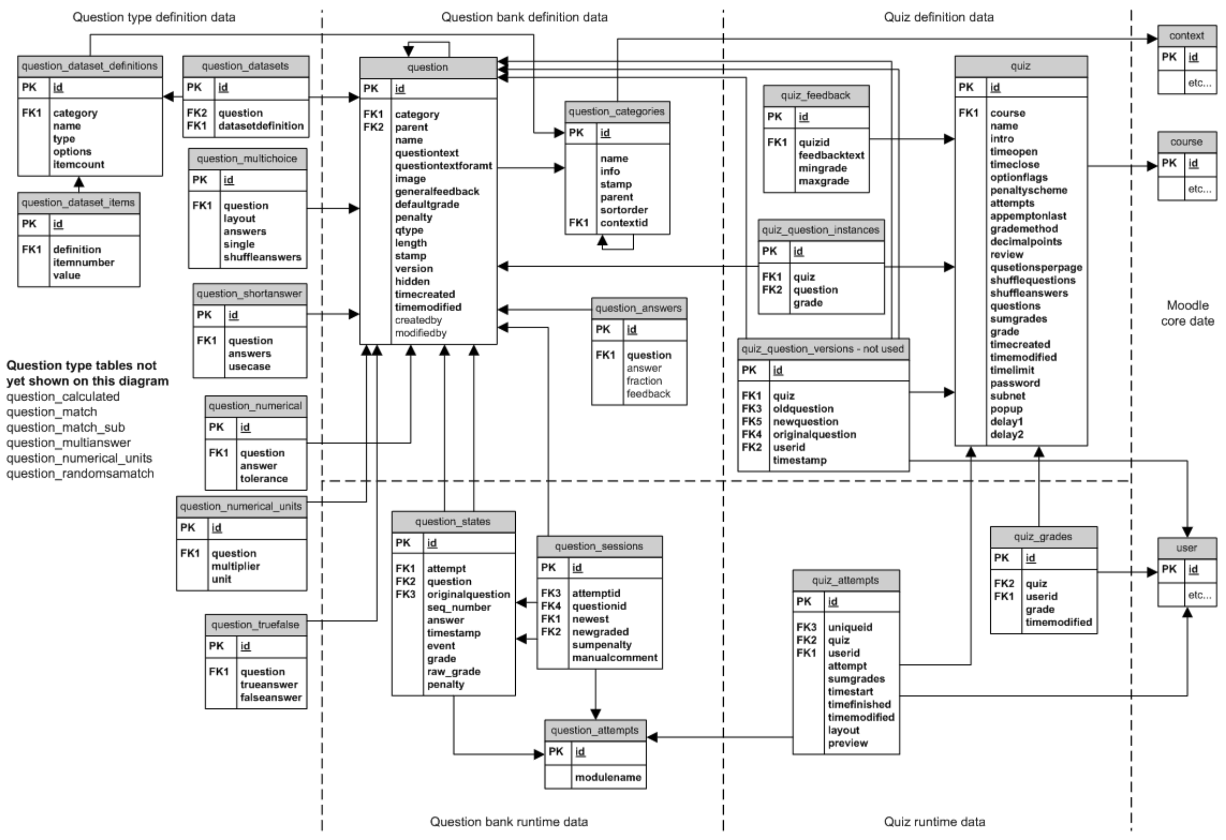

Question type definition data
Question bank definition data
Quiz definition data
question_dataset_definitions
PK id
FK1 category name type options itemcount
question_datasets
PK id
FK2 question
FK1 datasetdefinition
question_dataset_items
PK id
FK1 definition itemnumber value
question_multichoice
PK id
FK1 question layout answers single shuffleanswers
question_shortanswer
PK id
FK1 question answers usecase
question_numerical
PK id
FK1 question answer tolerance
question_numerical_units
PK id
FK1 question multiplier unit
question_truefalse
PK id
FK1 question trueanswer falseanswer
Question type tables not yet shown on this diagram
question_calculated
question_match
question_match_sub
question_multianswer
question_numerical_units
question_randomsamatch
question
PK id
FK1 category
FK2 parent
name questiontext questiontextforamt image generalfeedback defaultgrade penalty qtype length stamp version hidden timecreated timemodified createdby modifiedby
question_categories
PK id
name info stamp parent sortorder
FK1 contextid
question_answers
PK id
FK1 question answer fraction feedback
quiz_feedback
PK id
FK1 quizid feedbacktext mingrade maxgrade
quiz_question_instances
PK id
FK1 quiz
FK2 question grade
quiz_question_versions - not used
PK id
FK1 quiz
FK3 oldquestion
FK5 newquestion
FK4 originalquestion
FK2 userid timestamp
quiz
PK id
FK1 course name intro timeopen timeclose optionflags penaltyscheme attempts appemptonlast grademethod decimalpoints review qusetionsperpage shufflequestions shuffleanswers questions sumgrades grade timecreated timemodified timelimit password subnet popup delay1 delay2
context
PK id
etc...
course
PK id
etc...
Moodle core date
question_states
PK id
FK1 attempt
FK2 question
FK3 originalquestion seq_number answer timestamp event grade raw_grade penalty
question_sessions
PK id
FK3 attemptid
FK4 questionid
FK1 newest
FK2 newgraded sumpenalty manualcomment
question_attempts
PK id
modulename
quiz_attempts
PK id
FK3 uniqueid
FK2 quiz
FK1 userid attempt sumgrades timestart timefinished timemodified layout preview
quiz_grades
PK id
FK2 quiz
FK1 userid grade timemodified
user
PK id
etc...
Question bank runtime data
Quiz runtime data

Figure 3.1. Database structure for the quiz and questions bank

The quiz data model has a fairly large pool of database tables, so the first step in explaining them is to provide some order. Conceptually it is possible to distinguish between the tables holding the teacher-supplied data, defining quizzes and questions, and the tables storing all the data that is generated when students interact with the quizzes and questions.

A further simplification is possible by separating out the questiontype specific tables. They are logical extensions to other tables and therefore are not necessary for understanding the general basic model. They are therefore explained on the developer documentation page for the individual question types.

The diagram below shows how the most important tables are linked to one another.

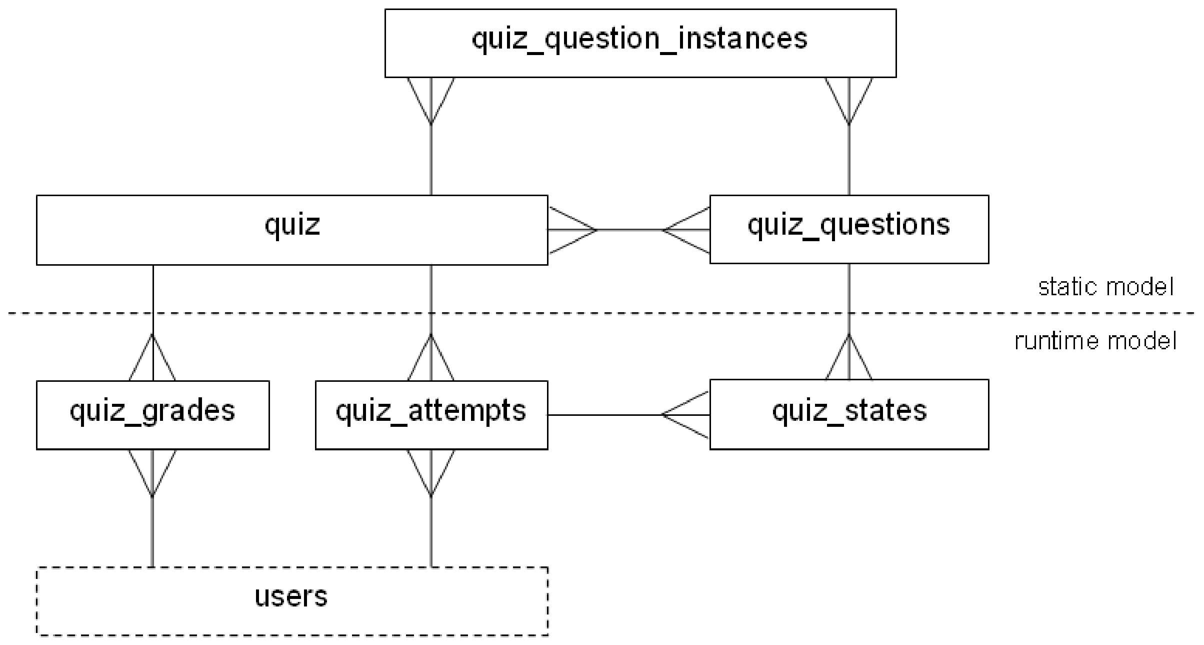


Figure 3.2. Quiz related tables links

While this one show how the newly add quiz plugin works and its flows

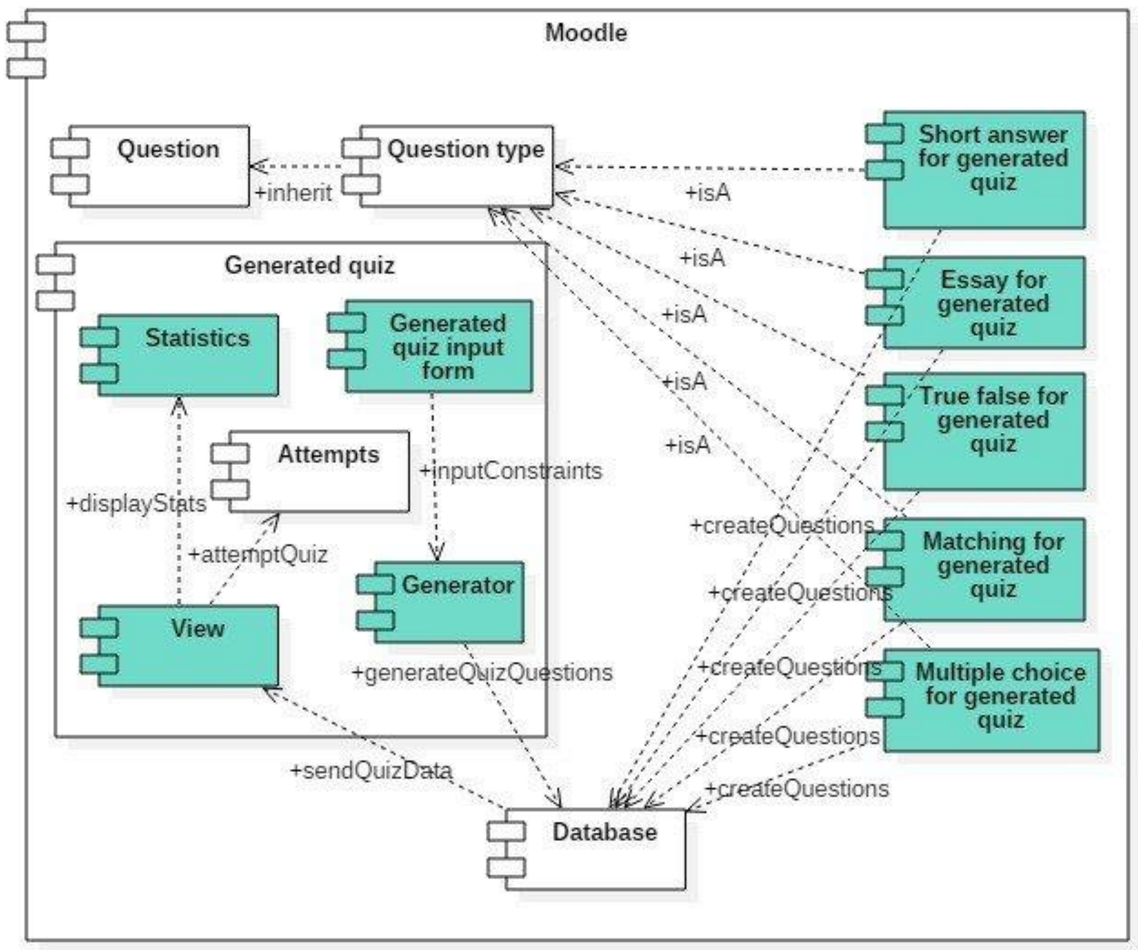


Figure 3.3. Quiz generator work flows

To integrate OpenAI with Moodle, a plugin folder was created inside the local directory of Moodle's file structure. This plugin acts as a bridge between Moodle’s user interface and OpenAI’s API. The folder was uploaded as a ZIP file through Moodle's Site Administration > Plugins > Install Plugins interface, allowing administrators to seamlessly add this functionality without manual file handling.

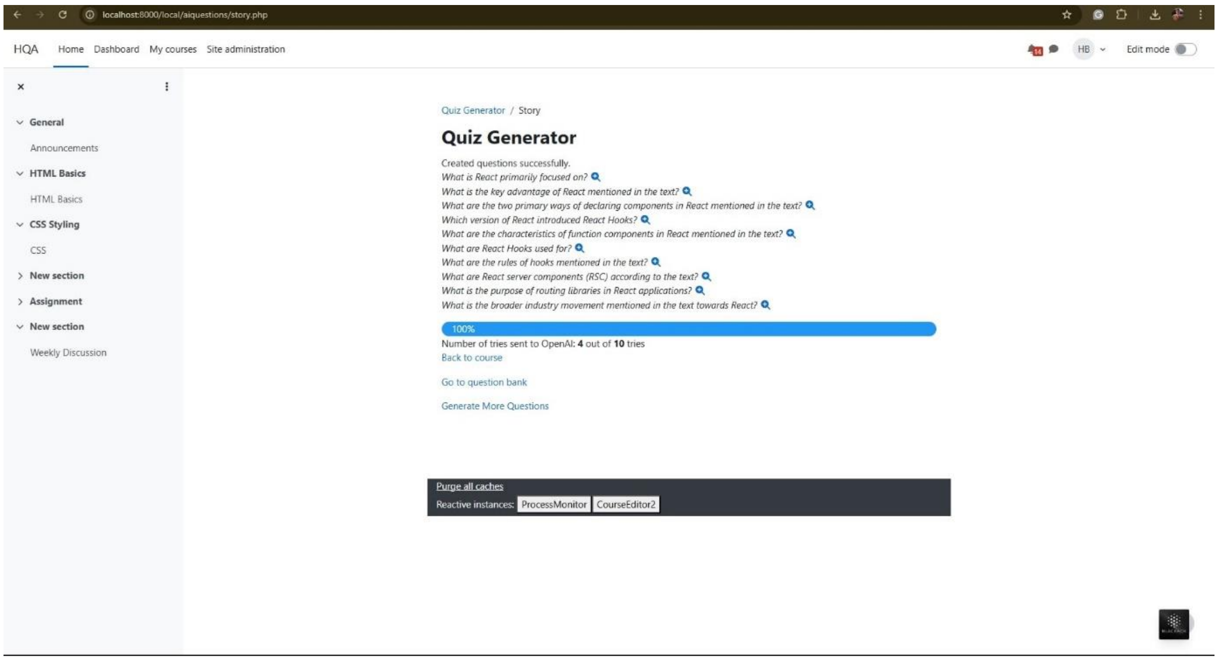
Key elements inside the plugin folder include:

db/install.xml: Defines the database schema for storing plugin-specific configurations or generated quiz metadata.

settings.php: Provides a user interface in Moodle for configuring API keys and other parameters.

lib.php and index.php: Implements the core functionality, handling API communication and user interactions.

The modular approach ensures the plugin is isolated, easily maintained, and adheres to Moodle’s architecture, allowing it to coexist with other extensions.



Quiz Generator / Story

Quiz Generator

Created questions successfully.
What is React primarily focused on?
What is the key advantage of React mentioned in the text?
What are the two primary ways of declaring components in React mentioned in the text?
Which version of React introduced React Hooks?
What are the characteristics of function components in React mentioned in the text?
What are React Hooks used for?
What are the rules of hooks mentioned in the text?
What are React server components (RSC) according to the text?
What is the purpose of routing libraries in React applications?
What is the broader industry movement mentioned in the text towards React?

100%

Number of tries sent to OpenAI: **4** out of **10** tries
Back to course

Go to question bank

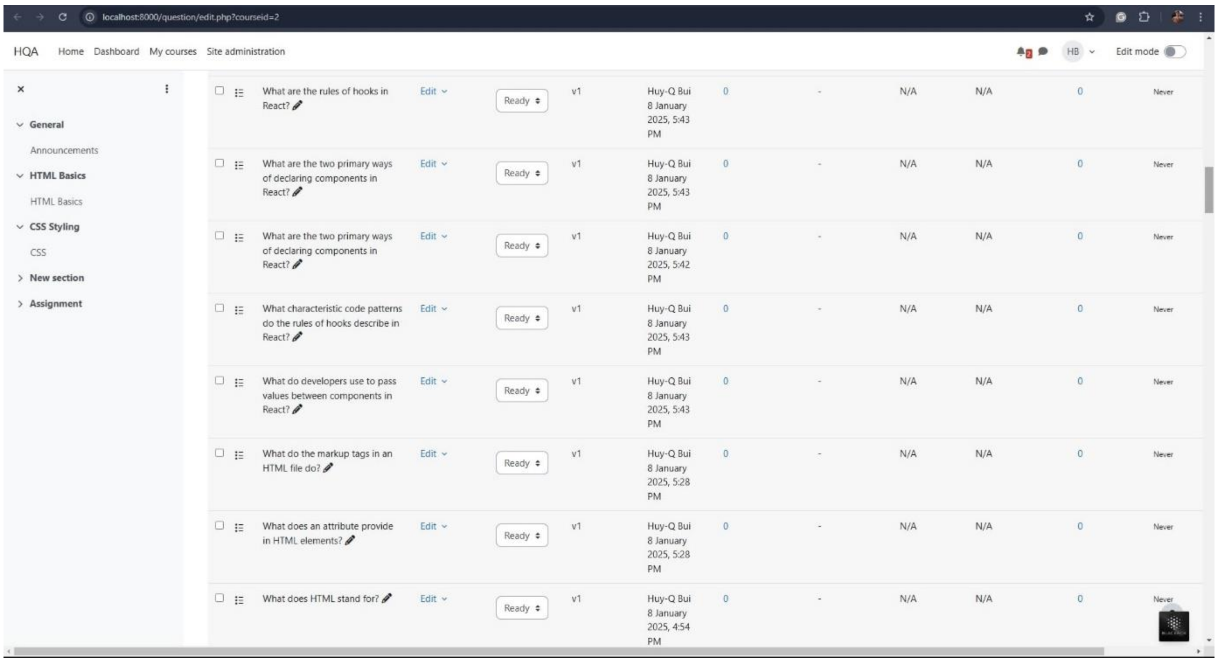
Generate More Questions

Purge all caches
Reactive instances: ProcessMonitor CourseEditor2

Figure 3.4. Questions generated by Quiz Generator by OpenAI

3.1.2 API Communication:

- Utilized secure HTTPS communication with the OpenAI API.
- The plugin sends an HTTP request to OpenAI’s API endpoint
- Handled API keys securely to prevent unauthorized usage.
- The API processes the request and generates a response containing structured text or JSON data representing quiz questions and answers.
- The plugin parses the response, formats the questions into Moodle-compatible structures, and optionally stores them in the database or question bank.



What are the rules of hooks in React?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 5:43 PM	0	-	N/A	N/A	0	Never
What are the two primary ways of declaring components in React?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 5:43 PM	0	-	N/A	N/A	0	Never
What are the two primary ways of declaring components in React?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 5:42 PM	0	-	N/A	N/A	0	Never
What characteristic code patterns do the rules of hooks describe in React?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 5:43 PM	0	-	N/A	N/A	0	Never
What do developers use to pass values between components in React?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 5:43 PM	0	-	N/A	N/A	0	Never
What do the markup tags in an HTML file do?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 5:28 PM	0	-	N/A	N/A	0	Never
What does an attribute provide in HTML elements?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 5:28 PM	0	-	N/A	N/A	0	Never
What does HTML stand for?	Edit	Ready	v1	Huy-Q Bui 8 January 2025, 4:54 PM	0	-	N/A	N/A	0	Never

Figure 3.5. Questions go to question bank

After being stored in questions bank, these questions can be used for good all up to the teachers/instructor. They view them on the Moodle interface and review or modify them before deploying them in quizzes or assignments.

3.2. Fake Attacks Simulated

3.2.1. Open Redirect Vulnerability (CVE-2022-45143)

Step 1: Find a Redirect Endpoint:

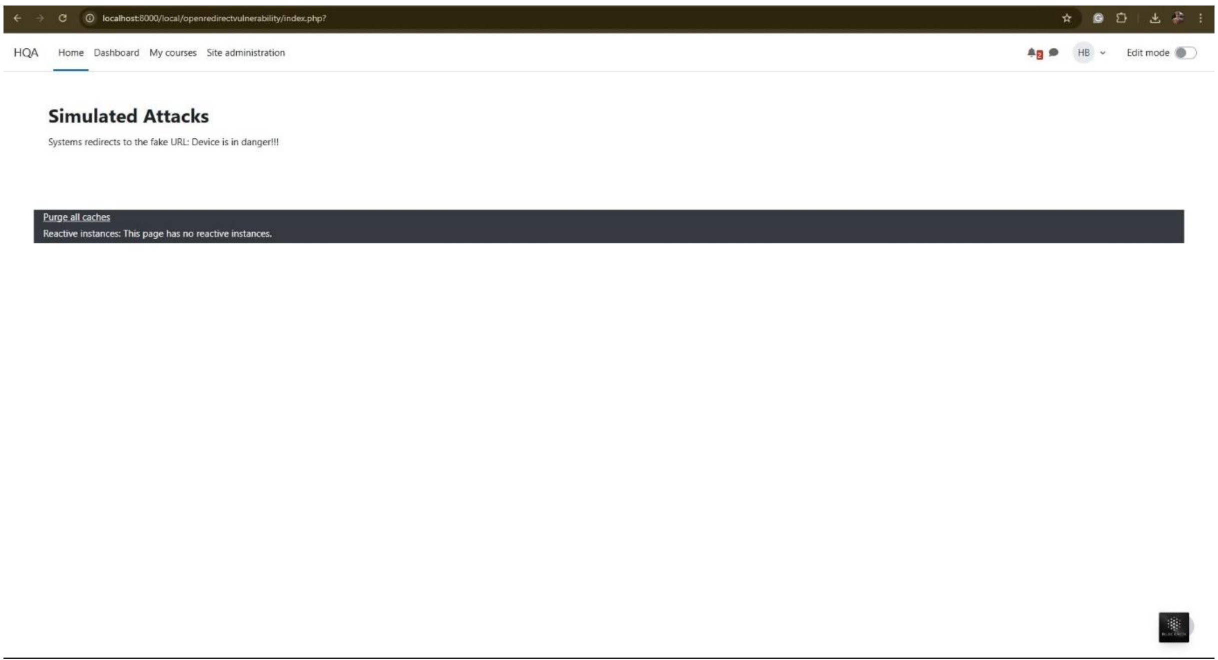
- Look for a URL in Moodle platform that includes a redirect or URL parameter. Example:

bash

http://example.com/redirect?url=http://malicious.com

Step 2: Test the Redirect:

- Replace the url parameter with a link to a fake or safe malicious site (e.g., http://test-attack.com).
- Click the link or share it to simulate how an attacker might trick a user.



localhost:8000/local/openredirectvulnerability/index.php?

HQA Home Dashboard My courses Site administration

Simulated Attacks

Systems redirects to the fake URL: Device is in danger!!!

Purge all caches

Reactive instances: This page has no reactive instances.

Figurue 3.6. Attack using Open Redirect Vulnerability

Step 3: Check the Behavior:

- If the system redirects to the fake URL, the vulnerability is confirmed.

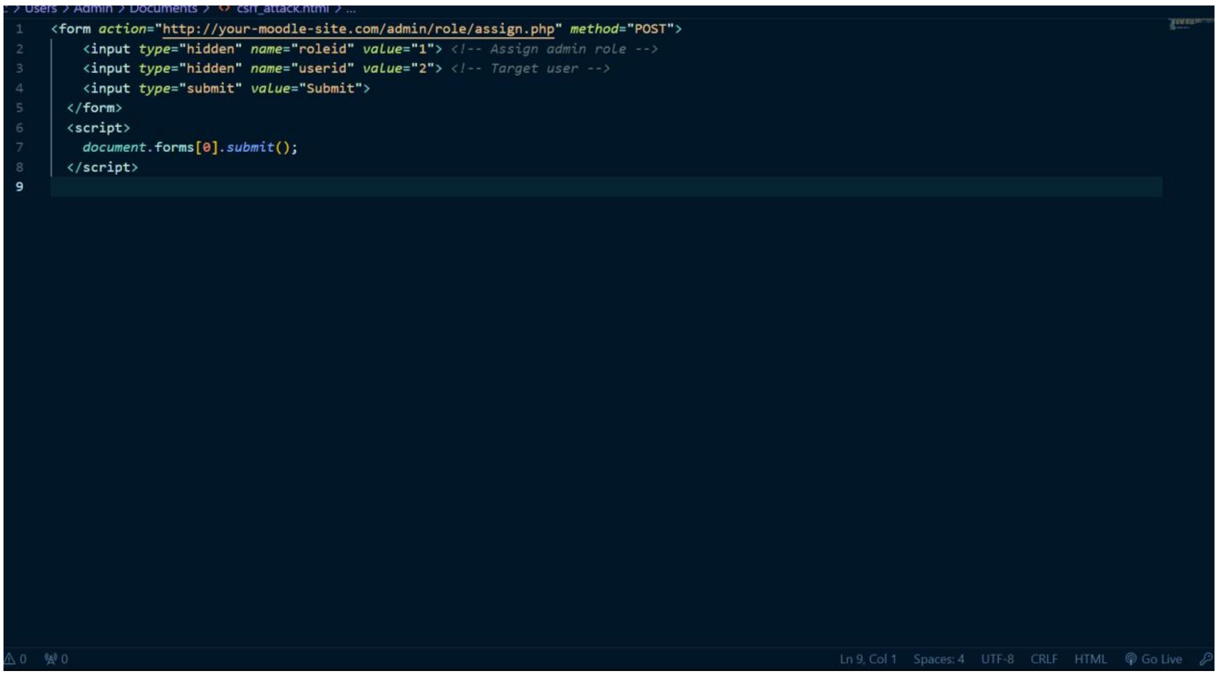
Prevention (to simulate a fix):

- Add URL validation to only allow redirection to trusted domains.
- Implement whitelists in Moodle’s code.

3.2.2. CSRF (Cross-Site Request Forgery) (CVE-2014-7833)

Step 1: Create a Malicious HTML Form:

Step2: Create a file called csrf_attack.html with the following code:



```html
<form action="http://your-moodle-site.com/admin/role/assign.php" method="POST">
    <input type="hidden" name="roleid" value="1"> <!-- Assign admin role -->
    <input type="hidden" name="userid" value="2"> <!-- Target user -->
    <input type="submit" value="Submit">
  </form>
  <script>
    document.forms[0].submit();
  </script>
```

Figure 3.7. Malicious file to exploit CSRF

Step 3: Host the File:

- Serve the csrf_attack.html file using a simple web server (e.g., Python):

bash

python3 -m http.server 8080

- Access it at http://localhost:8080/csrf_attack.html.

Step 4: Trigger the Attack

- While logged into Moodle as an admin, open the malicious webpage.
- Observe if it performs the unintended action (e.g., assigns admin role to a user).

Prevention (to simulate a fix):

- Use Moodle's built-in CSRF tokens (sesskey parameter).
- Validate requests using require_sesskey() in Moodle code.

3.2.3. Stored XSS (CVE-2021-40524)

Step 1: Understand the Stored XSS Attack

Stored XSS occurs when a malicious script is injected into a form or input field (e.g., a forum post or comment). The script is stored in the database and executed whenever the affected page is loaded by a user.

Step 2: Identify a Vulnerable Input Field in Moodle

Moodle has various input fields where users can submit content. For example:

- Forum Posts: Moodle allows users to create and reply to forum posts.
- Comment Boxes: Moodle supports comments in activities or resources.
- Profile Fields: Users can edit their profile and submit text in certain fields.

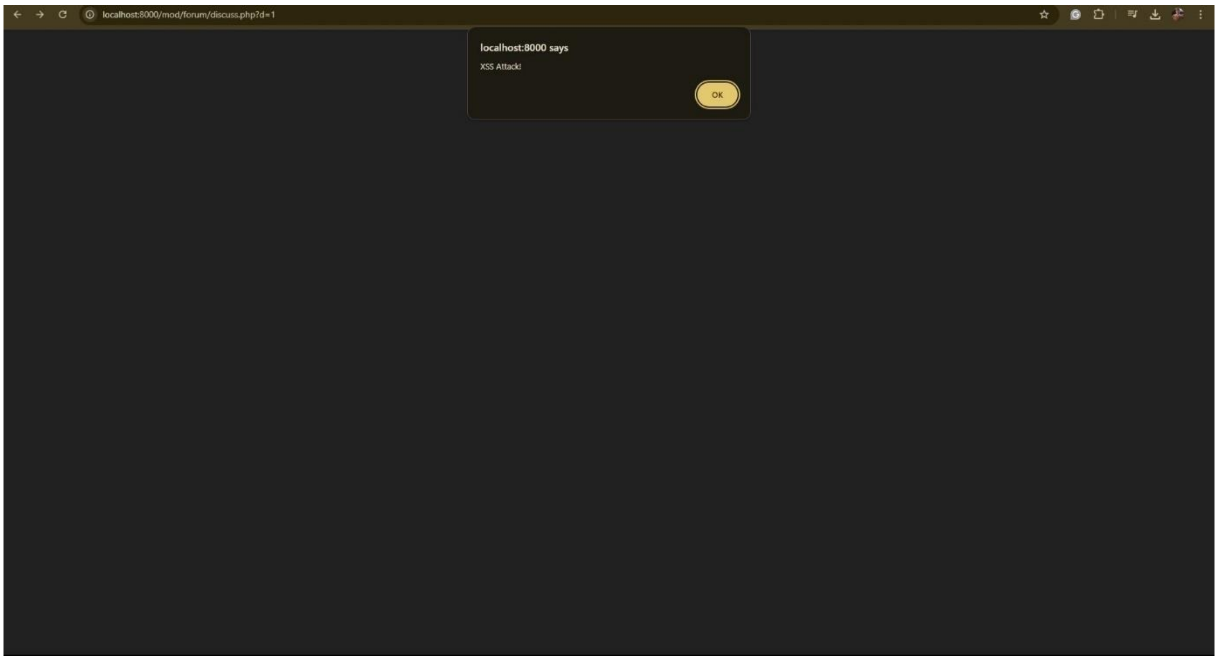
Step 3: Inject a Malicious Script

- Log in to Moodle:
- Use an account with permissions to post or comment (e.g., a teacher or student role).
- Navigate to a Comment or Post Section:
- Go to a forum, activity, or profile section that accepts user input.
- Submit Malicious Input:
- In the input field, submit a script: <script>alert('XSS Attack!');</script> in html

If the system is vulnerable, this script will be stored in the database and executed whenever a user visits the page.

Step 4: Verify the Attack

- Visit the Affected Page:
- Log in as another user (e.g., a student) and visit the page where the malicious content was posted.



localhost:8000 says
XSS Attack!

Figure 3.8. Cyberattack using XSS vulnerability

- Check for Script Execution: If the site is vulnerable, a pop-up alert with the message XSS Attack! will appear.

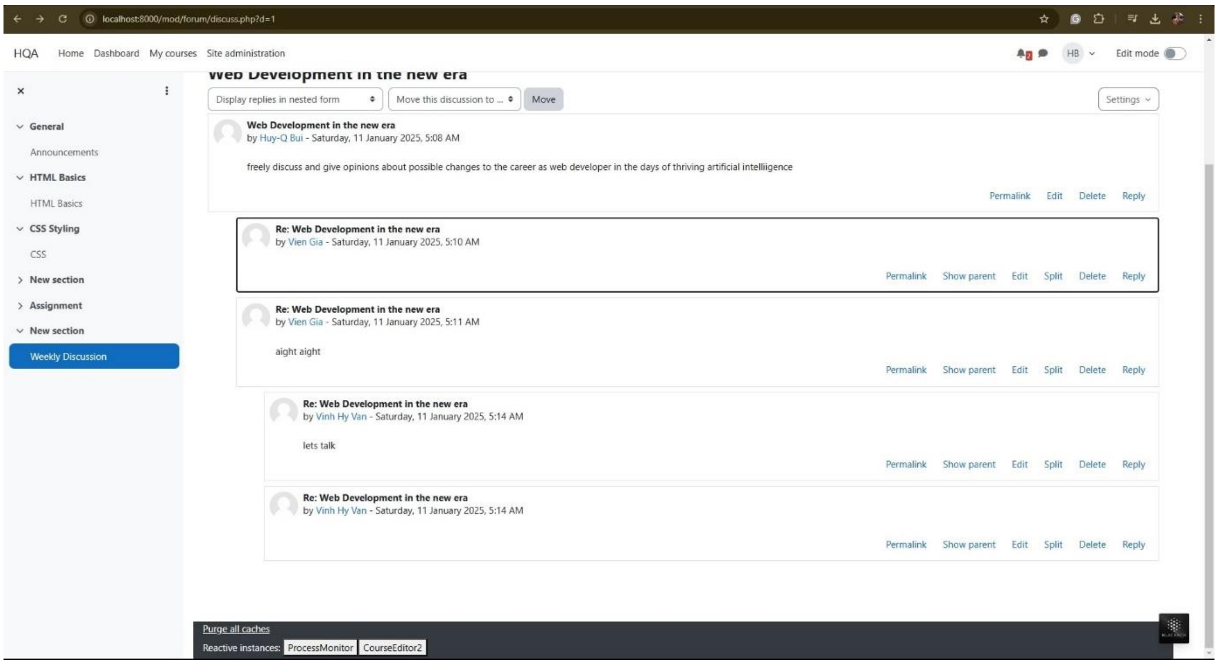
Prevention:

- Sanitize Input:

Use Moodle's clean_param() function to validate and sanitize inputs before storing them in the database.

- Sanitize Output:

Use Moodle's s() function to encode output safely before rendering it on the page


localhost:8000/mod/forum/discuss.php?d=1

HQA Home Dashboard My courses Site administration

Web Development in the new era

Web Development in the new era
by Huy-Q Bui - Saturday, 11 January 2025, 5:08 AM

freely discuss and give opinions about possible changes to the career as web developer in the days of thriving artificial intelliigence

Re: Web Development in the new era
by Vien Gia - Saturday, 11 January 2025, 5:10 AM

Re: Web Development in the new era
by Vien Gia - Saturday, 11 January 2025, 5:11 AM

aight aight

Re: Web Development in the new era
by Vinh Hy Van - Saturday, 11 January 2025, 5:14 AM

lets talk

Re: Web Development in the new era
by Vinh Hy Van - Saturday, 11 January 2025, 5:14 AM

General
Announcements
HTML Basics
HTML Basics
CSS Styling
CSS
New section
Assignment
New section
Weekly Discussion
Display replies in nested form
Move this discussion to ...
Move
Settings
Permalink Show parent Edit Split Delete Reply
Edit mode
Purge all caches
Reactive instances: ProcessMonitor CourseEditor2

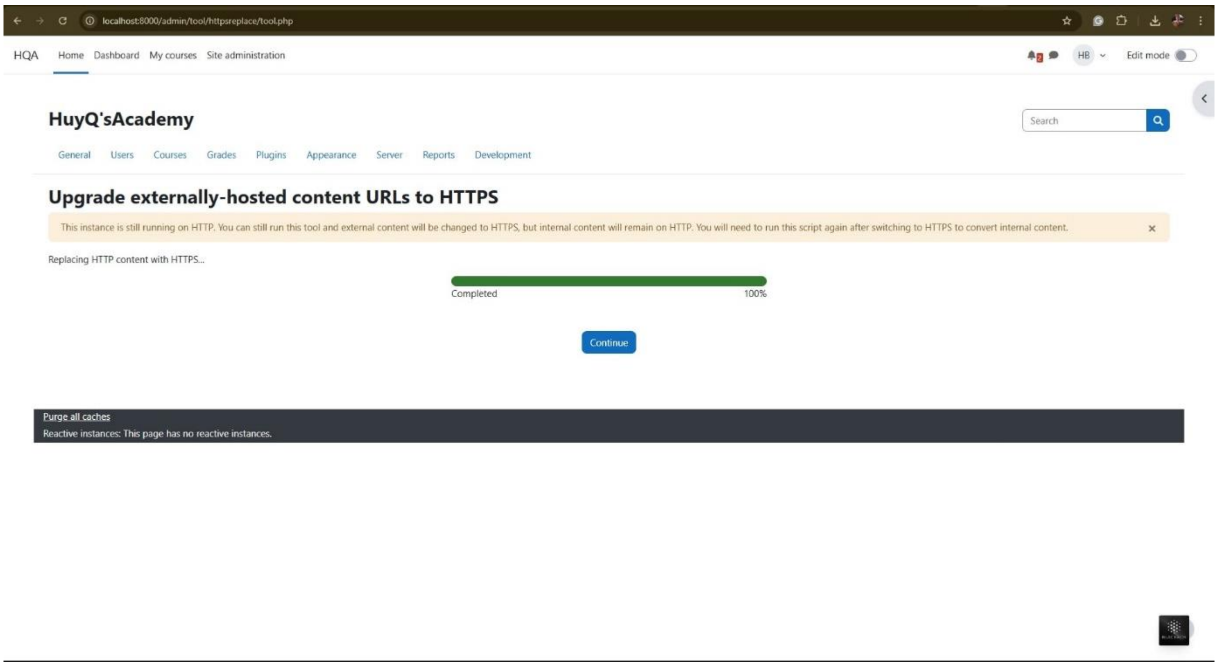
Figure 3.9. Site after being secured

As figure above, after applying both prevention and adjustments in the code, page came back to normal. The macious content also disappeared

I also properly secured the installation to guarantee the site as safe as possible

Secure Configuration:

- Enable HTTPS/SSL



HuyQ'sAcademy

Upgrade externally-hosted content URLs to HTTPS

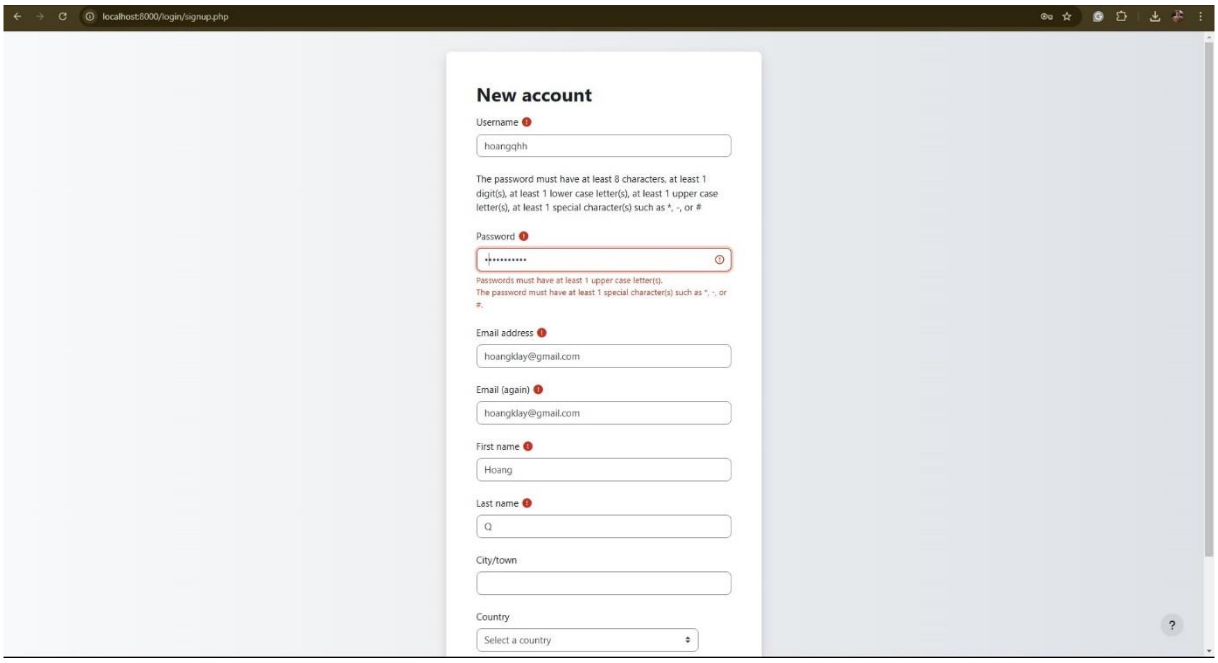
This instance is still running on HTTP. You can still run this tool and external content will be changed to HTTPS, but internal content will remain on HTTP. You will need to run this script again after switching to HTTPS to convert internal content.

Replacing HTTP content with HTTPS...

Reactive instances: This page has no reactive instances.

Figure 3.10. HTTPS enabled

- Set strong password policies



New account

Username: hoangqhh

The password must have at least 8 characters, at least 1 digit(s), at least 1 lower case letter(s), at least 1 upper case letter(s), at least 1 special character(s) such as *, -, or #

Password: •••••••••••

Passwords must have at least 1 upper case letter(s).
The password must have at least 1 special character(s) such as *, -, or #.

Email address: hoangklay@gmail.com

Email (again): hoangklay@gmail.com

First name: Hoang

Last name: Q

City/town:

Country: Select a country

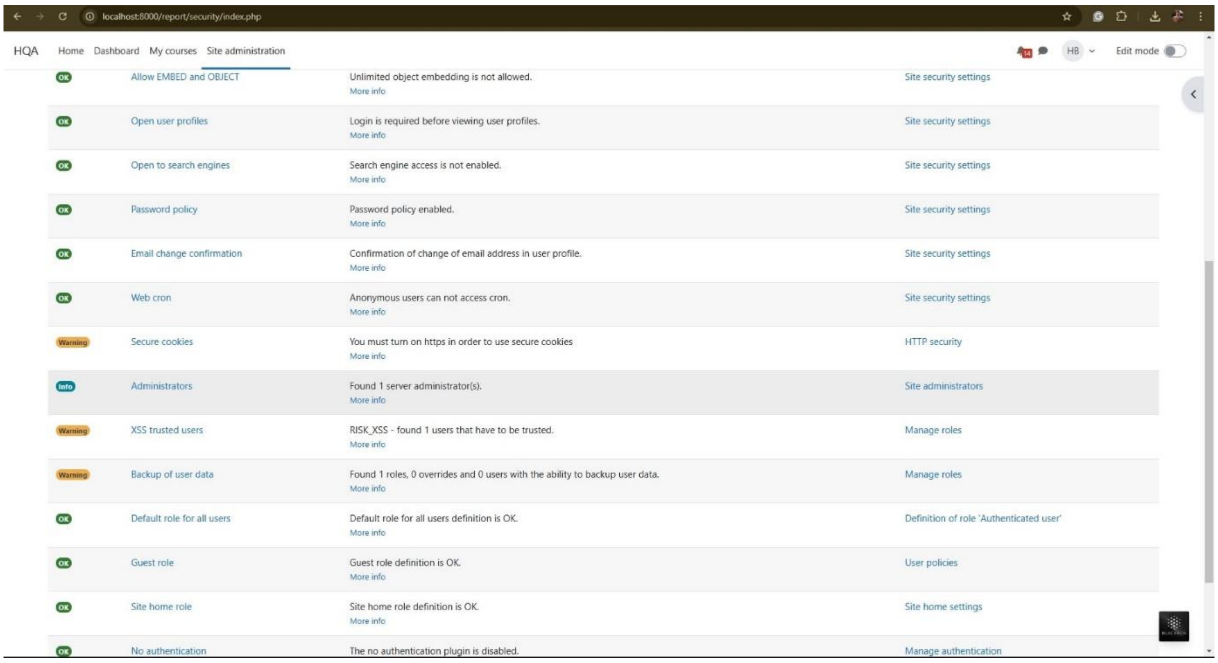
Figure 3.11. Strong password policy applied

- Configure session timeouts

- Limit login attempts
- Disable guest access if not needed
- Keep Moodle and all plugins updated
- Use secure file upload settings

Legitimate Security Testing:

- Run Moodle's built-in security overview report



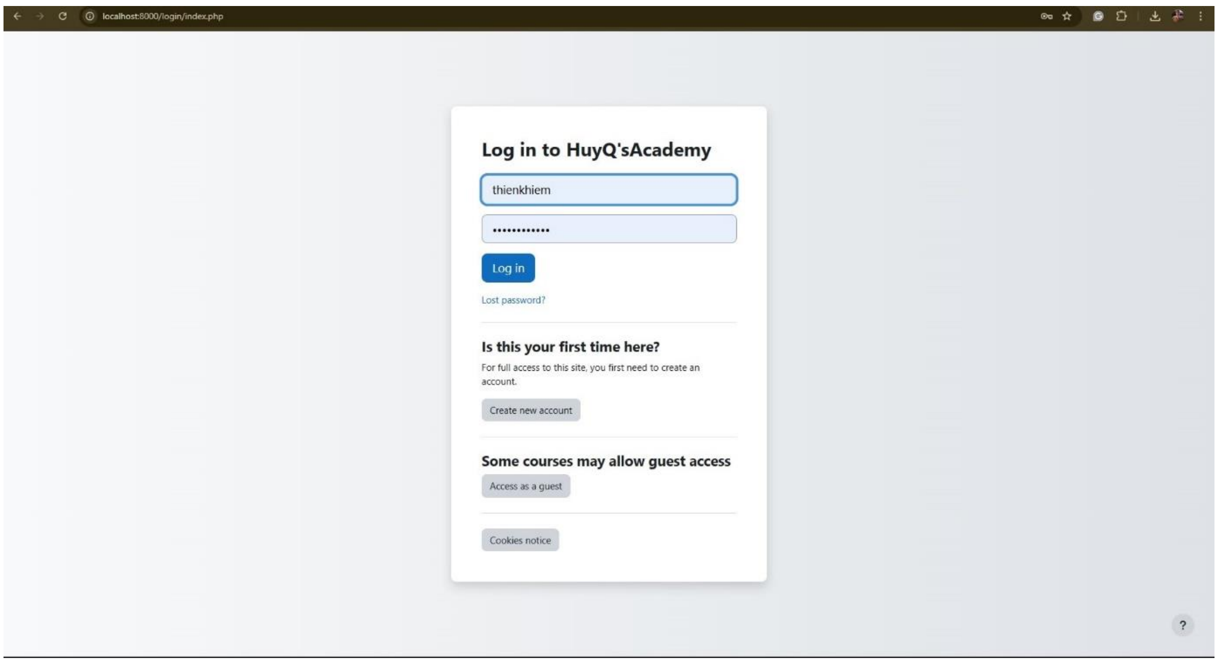
localhost:8000/report/security/index.php

OK	Allow EMBED and OBJECT	Unlimited object embedding is not allowed. More info	Site security settings
OK	Open user profiles	Login is required before viewing user profiles. More info	Site security settings
OK	Open to search engines	Search engine access is not enabled. More info	Site security settings
OK	Password policy	Password policy enabled. More info	Site security settings
OK	Email change confirmation	Confirmation of change of email address in user profile. More info	Site security settings
OK	Web cron	Anonymous users can not access cron. More info	Site security settings
Warning	Secure cookies	You must turn on https in order to use secure cookies More info	HTTP security
Info	Administrators	Found 1 server administrator(s). More info	Site administrators
Warning	XSS trusted users	RISK_XSS - found 1 users that have to be trusted. More info	Manage roles
Warning	Backup of user data	Found 1 roles, 0 overrides and 0 users with the ability to backup user data. More info	Manage roles
OK	Default role for all users	Default role for all users definition is OK. More info	Definition of role 'Authenticated user'
OK	Guest role	Guest role definition is OK. More info	User policies
OK	Site home role	Site home role definition is OK. More info	Site home settings
OK	No authentication	The no authentication plugin is disabled.	Manage authentication

Figure 3.12. Security report

- Check permissions settings
- Verify user role configurations
- Test backup/restore procedures
- Audit plugin security settings
- Monitor system logs
- Review authentication methods

3.3.3. Result Evaluation:


localhost:8000/login/index.php

Log in to HuyQ'sAcademy

thienkhiem

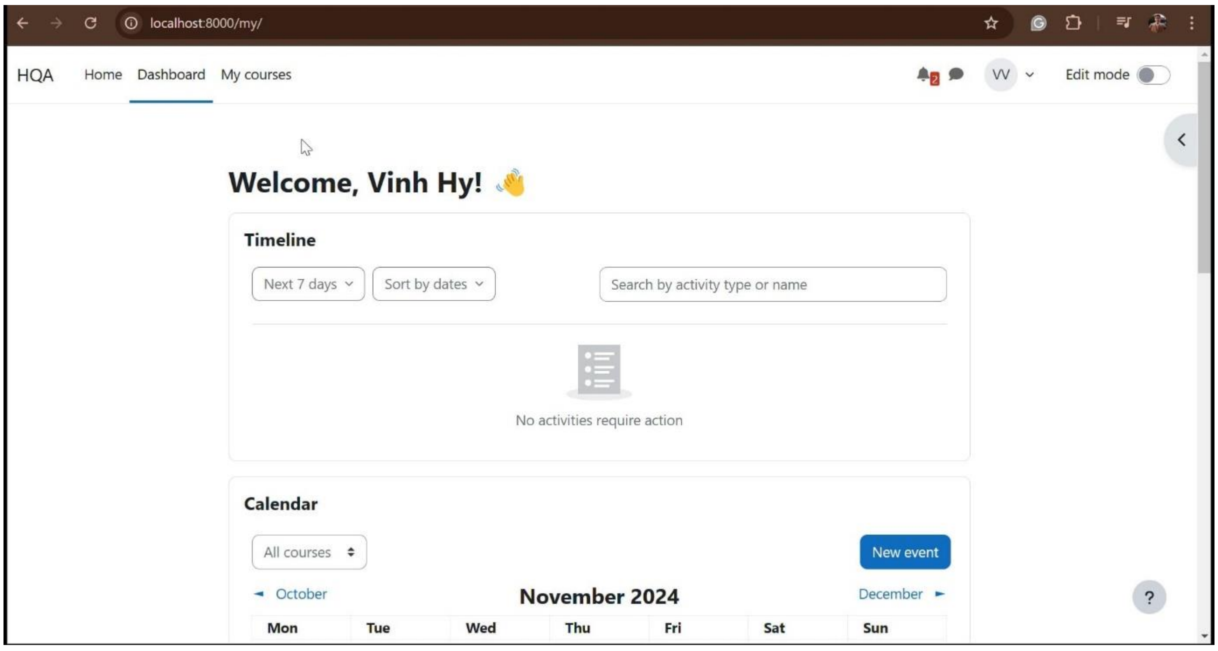
••••••••••••

Is this your first time here?

For full access to this site, you first need to create an account.

Some courses may allow guest access

Figure 3.13. Log in/Sign up Interface


localhost:8000/my/

Welcome, Vinh Hy! 👋

Timeline

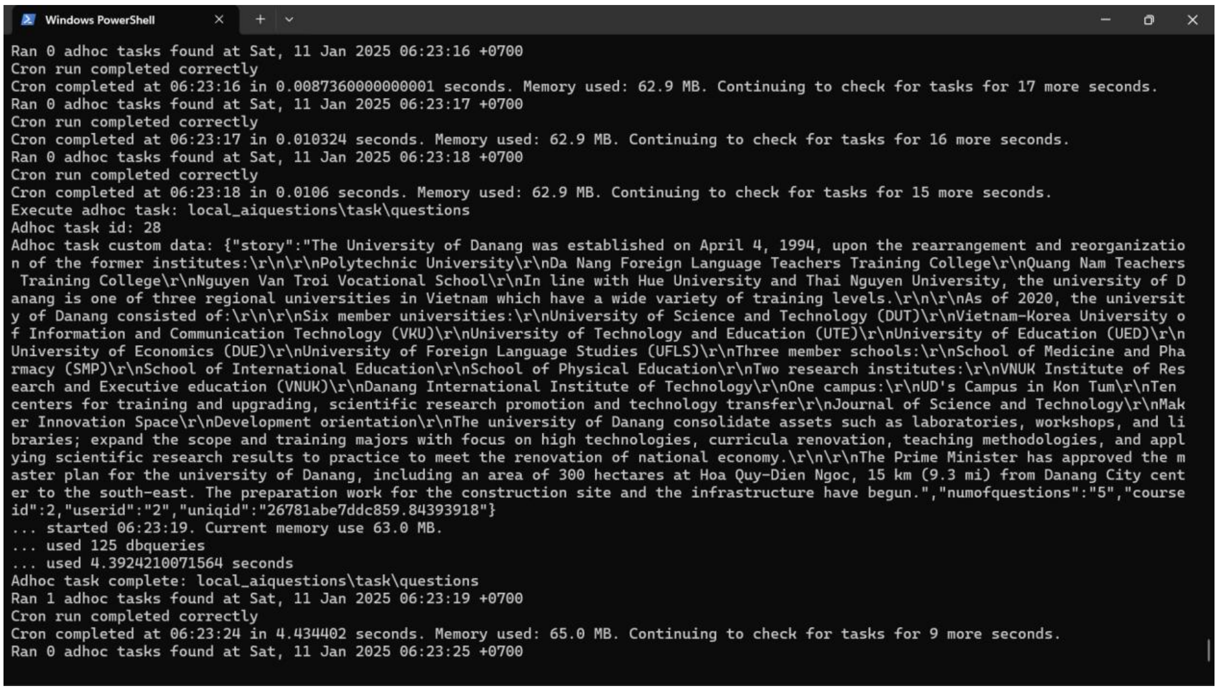
No activities require action

Calendar

November 2024

Mon	Tue	Wed	Thu	Fri	Sat	Sun

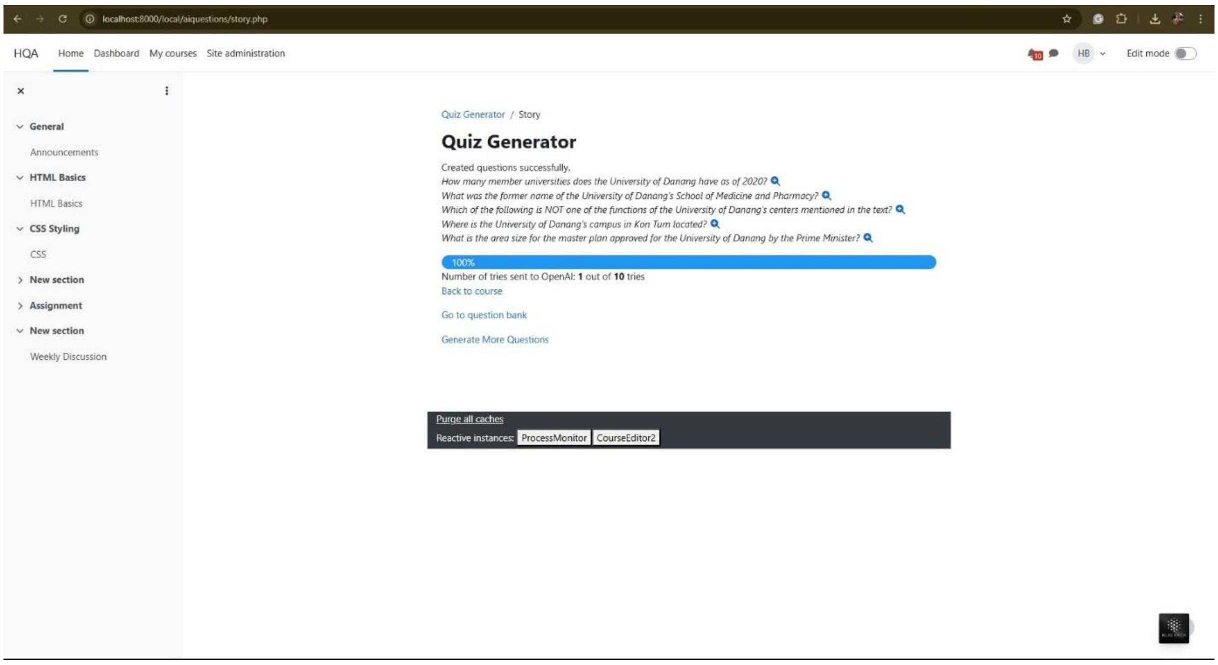
Figure 3.14. Homepage/Dashboard of the site



```
Ran 0 adhoc tasks found at Sat, 11 Jan 2025 06:23:16 +0700
Cron run completed correctly
Cron completed at 06:23:16 in 0.0087360000000001 seconds. Memory used: 62.9 MB. Continuing to check for tasks for 17 more seconds.
Ran 0 adhoc tasks found at Sat, 11 Jan 2025 06:23:17 +0700
Cron run completed correctly
Cron completed at 06:23:17 in 0.010324 seconds. Memory used: 62.9 MB. Continuing to check for tasks for 16 more seconds.
Ran 0 adhoc tasks found at Sat, 11 Jan 2025 06:23:18 +0700
Cron run completed correctly
Cron completed at 06:23:18 in 0.0106 seconds. Memory used: 62.9 MB. Continuing to check for tasks for 15 more seconds.
Execute adhoc task: local_aiquestions\task\questions
Adhoc task id: 28
Adhoc task custom data: {"story":"The University of Danang was established on April 4, 1994, upon the rearrangement and reorganizatio
n of the former institutes:\r\n\r\nPolytechnic University\r\nDa Nang Foreign Language Teachers Training College\r\nQuang Nam Teachers
 Training College\r\nNguyen Van Troi Vocational School\r\nIn line with Hue University and Thai Nguyen University, the university of D
anang is one of three regional universities in Vietnam which have a wide variety of training levels.\r\n\r\nAs of 2020, the universit
y of Danang consisted of:\r\n\r\nSix member universities:\r\nUniversity of Science and Technology (DUT)\r\nVietnam-Korea University o
f Information and Communication Technology (VKU)\r\nUniversity of Technology and Education (UTE)\r\nUniversity of Education (UED)\r\n
University of Economics (DUE)\r\nUniversity of Foreign Language Studies (UFLS)\r\nThree member schools:\r\nSchool of Medicine and Pha
rmacy (SMP)\r\nSchool of International Education\r\nSchool of Physical Education\r\nTwo research institutes:\r\nVNUK Institute of Res
earch and Executive education (VNUK)\r\nDanang International Institute of Technology\r\nOne campus:\r\nUD's Campus in Kon Tum\r\nTen
centers for training and upgrading, scientific research promotion and technology transfer\r\nJournal of Science and Technology\r\nMak
er Innovation Space\r\nDevelopment orientation\r\nThe university of Danang consolidate assets such as laboratories, workshops, and li
braries; expand the scope and training majors with focus on high technologies, curricula renovation, teaching methodologies, and appl
ying scientific research results to practice to meet the renovation of national economy.\r\n\r\nThe Prime Minister has approved the m
aster plan for the university of Danang, including an area of 300 hectares at Hoa Quy-Dien Ngoc, 15 km (9.3 mi) from Danang City cent
er to the south-east. The preparation work for the construction site and the infrastructure have begun.","numofquestions":"5","course
id":2,"userid":"2","uniqid":"26781abe7ddc859.84393918"}
... started 06:23:19. Current memory use 63.0 MB.
... used 125 dbqueries
... used 4.3924210071564 seconds
Adhoc task complete: local_aiquestions\task\questions
Ran 1 adhoc tasks found at Sat, 11 Jan 2025 06:23:19 +0700
Cron run completed correctly
Cron completed at 06:23:24 in 4.434402 seconds. Memory used: 65.0 MB. Continuing to check for tasks for 9 more seconds.
Ran 0 adhoc tasks found at Sat, 11 Jan 2025 06:23:25 +0700
```

Figure 3.15. Backend handling and processing quiz generate request

Initially, the quiz generator was designed to accept single words or short keywords as input, which worked well for generating broad and generalized questions. However, shifting to long-text inputs, such as paragraphs or detailed material, significantly enhances the power and relevance of the generated quizzes. By providing a comprehensive context, the AI can craft more specific, nuanced, and accurate questions tailored to the actual teaching content. This approach aligns better with educational practices, as it directly leverages the material being taught, ensuring the generated quizzes are more meaningful and closely related to the course objectives. The switch also minimizes the need for manual adjustments by teachers, streamlining the process and making the feature far more effective in real-world use.



Quiz Generator / Story

Quiz Generator

Created questions successfully.
How many member universities does the University of Danang have as of 2020?
What was the former name of the University of Danang's School of Medicine and Pharmacy?
Which of the following is NOT one of the functions of the University of Danang's centers mentioned in the text?
Where is the University of Danang's campus in Kon Tum located?
What is the area size for the master plan approved for the University of Danang by the Prime Minister?

100%

Number of tries sent to OpenAI: **1** out of **10** tries

Back to course

Go to question bank

Generate More Questions

Purge all caches

Reactive instances: ProcessMonitor CourseEditor2

Figure 3.16. Questions generated about Danang University

………………………………………………………………………………………..

CONCLUSION

Achievements

The project successfully demonstrated the integration of OpenAI's API within Moodle to generate quizzes, providing an innovative feature for enhancing the educational experience. Additionally, the implementation of security testing strategies, including simulated attacks like XSS, SQL Injection, and other vulnerabilities, highlighted areas for improvement in Moodle’s defense mechanisms. The focus on patching vulnerabilities and improving input validation showcases significant progress in bolstering Moodle's resilience against common threats.

Limitations

Despite the successes, the project faced several limitations. The integration of the OpenAI API was designed as a demonstration and is not fully optimized for large-scale deployments. Furthermore, while the project covered several attack simulations, it could not encompass all potential vulnerabilities due to resource and time constraints. The security testing relied heavily on manually crafted scenarios, limiting the exploration of more advanced or rare attack techniques. Lastly, the absence of in-depth deployment testing for diverse environments restricts the broader applicability of the findings.

Development Orientation

Future development will focus on optimizing the OpenAI quiz generator to support larger datasets and more complex interactions. Enhancing Moodle’s defenses by integrating real-time monitoring tools, such as Burp Suite and SQLMap, will be prioritized to ensure proactive vulnerability management. Additionally, the implementation of automated testing frameworks will facilitate a broader analysis of security flaws. Efforts will also include exploring advanced authentication mechanisms and encryption strategies to further secure user data and prevent privilege escalation or information leaks. This ongoing development aims to make Moodle a more robust and secure platform for educational institutions worldwide.

REFERENCES

{bold, size 14}

{Để 2 dòng trống}

Moodle Documentation and Forums

Moodle Plugin Development Guide: https://docs.moodle.org/dev/Plugin_development

Moodle Security Guidelines: https://docs.moodle.org/secure/

Moodle Community Forums: https://moodle.org/mod/forum/

OpenAI Documentation and Resources

OpenAI API Documentation: https://platform.openai.com/docs/

Examples of GPT-3 Use Cases: https://openai.com/blog/gpt-3-apps/

Best Practices for OpenAI API Usage: https://platform.openai.com/docs/best-practices

PHP and Plugin Development

PHP Official Documentation: https://www.php.net/docs.php

Secure PHP Development: https://owasp.org/www-project-secure-php-development/

AI in Education

AI in Education Overview: https://www.educationdive.com/news/how-ai-is-shaping-education/

Benefits of AI-Powered Learning Tools: https://edtechmagazine.com/k12/article/2021/07/how-ai-changing-classrooms

Security and Vulnerability Testing

OWASP Testing Guide: https://owasp.org/www-project-web-security-testing-guide/

Tools for Web Security Testing: https://owasp.org/www-community/Testing_Tools

Docker Resources

Docker Documentation: https://docs.docker.com/

Using Docker for Moodle: https://moodledev.io/general/releases/docker

Miscellaneous

RESTful API Security Guidelines: https://restfulapi.net/security-essentials/

AI Ethics in Education: https://www.unesco.org/en/digital-learning/ai-education

APPENDIX1

APPENDIX 2

{bold, size 14}