**The Design Proposal Document – School Learning Management System**

**Introduction:**

This design document covers the development of a secured learning management system for a school (Se-LMS). The Se-LMS will allow teachers, student and parents to view student data and assignment grades securely. The roles will be managed via an administrator (admin) user. Please use a connecting sentence for the below section(s) here.

**Rationale**

The application provides an essential, and expected solution to aid learning (DfE, 2024). Se-LMS, otherwise known as virtual learning environments, have been promoted in the UK over the last decade (Ofsted, 2014). The Department for Education (DfE) recognises the need for all schools to be able to provide high quality remote education, particularly for school closures and where a student is unable to attend, but able to learn (DfE, 2024).

**System requirements and Specification**

Users will log-in to the system using a secure password. The system will support the ability to turn security on and off. Users will be able to perform CRUD functions relevant to their roles. All appropriate legal frameworks, including GDPR will be observed.

**Functional requirements of the application**

The Se-LMS will incorporate a number of components outlined by Patel, Gadhavi & Patel (2013):

* Administrators will be able to add new courses, manage or update existing courses, assign teachers to courses, register, view, and enrol students.
* Teachers will be able to upload course content, view student submissions, and assign grades .
* Students will be able to view course content, their own assignment grades, and submit work.
* All users will require secure account log-ins.

**Legal requirements**

The software will adhere to GDPR regulations regarding its processing of user data, as detailed in below:

‘Consent for the processing of the users’ personal information must be obtained from students aged 13 and over, while consent from parents must be obtained for those under 133’ (Calder, 2018).

To comply with article 5 of the GDPR (Calder, 2018), the following steps will be taken:

* Users’ personal data will not be shared with third parties, and the user will be informed on how their data will be used.
* Personal data will only be used to inform teachers of who their students are and how they are progressing.
* Only relevant student data will be collected to ensure the functionality of the Se-LMS: first and last names, age, assigned teacher, and assignment marks; and for teachers: first and last names, subject speciality.
* The above data will be updated when relevant.
* When the student leaves the school, their data will be deleted.
* Security procedures will be followed to secure data processing.

**Security Requirements**

Secure authentication will be achieved by implementing role permissions. For example, only admin users will be able to register new students (MITRE, 2017). In addition, user passwords are hashed before being compared to the record of hashed user passwords (Spraul, 2015). User’s attempts at inputting their passwords will also be restricted to protect against Brute Force Attacks (Spraul 2015).

Event monitoring will be enforced by logging attempts made by users to log-in as an administrator (Kellezi et al., 2021). SQL injection attacks will be protected against by ensuring that user input is sanitised and never used directly for database queries (Galluccio, et al, 2020). Denial of service attacks will be mitigated against by ensuring that regular expressions are not exploited, with correct sanitisation of user input implemented and input evaluation times handled appropriately (Larson, 2018).

**Tools and Technology**

**Version Control System Used**: GitHub

A version control System is a collaborative platform used by developers to record and track modifications in source code. This helps software developers manage changes to the source code.

**Static Analysis Tools used**: Flake8

Static analysis is a process of examining the source code without executing the code. It helps to find out the weakness including programming errors, violation of coding standards and security vulnerability.Flake8 is one of the popular linter used in python developers.It checks the code against coding style(PEP8)

**IDE used**: Visual Studio

Integrated Development Environment (IDE) is a software application which includes tools for the developers to efficiently develop code. Visual studio comes with python extensions which include syntax autocorrecting,unit testing, git operation etc.

**Libraries for encryption and testing.**

For encryption, the app will use the ‘cryptography’ library which ensures that Flask session data is encrypted and therefore secure (Farmaan, 2024). The testing library will be ‘unit test’, as this allows for the development of tests that match complex, real-life scenarios (Pajankar, 2022).

**Development** **Methodology**

Traditional waterfall models demand thorough documentation, generally leading to a secure but inflexible development. The application will be developed using Secure Scrum which adheres to the Agile Manifesto, but incorporates additional security steps, monitored in the daily scrum meetings.

**UML Models**

UML diagrams can be found in appendix 1.

**OSWAP Proactive Controls**

The application will use OSWAP Level 3 Application Security Verification Standard (ASVS) requirements because sensitive data is held. The OSWAP top 10 secure coding strategies are (Pillai, 2017):

|  |  |
| --- | --- |
| Strategy | Impact |
| Validate inputs | Validating data from untrusted sources eliminates most vulnerabilities. |
| Keep it simple | Complex design increases the probability of security errors. |
| Principal of least privilege | All processes only have the privilege they need. |
| Sanitize data | Reduces the chances of SQL injections |
| Authorise access | Sensitive data is only accessed by those who need it. |
| Perform effective QA | Testing reduces risk. |
| Practice defence in layers | Reduces the impact of a single failure. |
| Define security requirements | Documented security requirements are used in updates. |
| Model threats | Anticipating threats ensures a proactive, secure development. |
| Architect and design for security policies | Consistent approach to security across the system. |

**Threat Modelling**

Threats will be modelled using the STRIDE methodology.

|  |  |
| --- | --- |
| Threat | Counter Measure |
| Spoofing | * Authentication
* Protect secret data
* Do not store secrets
 |
| Tampering with data | * Authorization
* Hashes
* MACs
* Digital signatures
 |
| Repudiation | * Timestamps
* Audit trails
* Digital signatures
 |
| Information Disclosure | * Authorization
* Encryption
* Do not store secrets
 |
| Denial of Service | * Authentication
* Authorization
* Throttling
 |
| Elevation of privilege | * Using lest privilege
 |
| Determine threat profile after mitigations | * Non mitigated
* Partially mitigated
* Fully mitigated
 |

The DREAD model will be used to quantify the risk:

|  |  |
| --- | --- |
| Impact of Attack (0-10) | High Score (10) |
| **D**amage | Destruction of system, data, or application unavailability. |
| **R**eproducibility | Attack is very easy to reproduce. |
| **E**xploitability | Attack via a web browser. |
| **A**ffected Users | All users affected. |
| **D**iscoverability | Vulnerability found in the web address bar. |

Threat Level

|  |  |  |  |
| --- | --- | --- | --- |
| 0-10 | 11-24 | 25-39 | 40-50 |
| Low | Medium | High | Critical |

**Conclusion**

The design proposal will ensure secure development of the application, delivered through a Secure Scrum approach. Threat modelling and mitigation will ensure security and the legal frameworks, such as GDPR, will be observed.

**Appendix 1 – UML Diagrams**

1. Specification Class Diagram

1. Use/Misuse Case Diagram



1. Activity Diagram



1. Sequence Diagram – without security



1. Sequence Diagram – with security



1. Denial of Service (DOS) attack - without security



1. Denial of Service (DOS) attack - with security



1. Brute force attack - without security



1. Brute force attack - with security



**References**

Calder, A. (2018). *EU GDPR: A Pocket Guide, School’s edition*. Cambridge: IT Governance Publishing. DOI: <https://doi.org/10.2307/j.ctvvnfgq>.

Galluccio, E., Caselli, E. and Lombari, G. (2020). SQL Injection Strategies: Practical techniques to secure old vulnerabilities against modern attacks. Packt Publishing Ltd. DOI?

Kellezi, D., Boegelund, C. and Meng, W. (2021) Securing open banking with model-view-controller architecture and OWASP. *Wireless Communications and Mobile Computing*, (1): 1-13. DOI: [**https://doi.org/10.1155/2021/8028073**](https://doi.org/10.1155/2021/8028073).

Farmaan, M. (2004). Secure Flask Sessions with Encryption: A Step-by-Step Guide. Available from: https://rb.gy/z2z9ga [Accessed: August 26, 2024].

E. Larson. (2018). [Research Paper] Automatic Checking of Regular Expressions. *IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, Madrid, Spain, 2018. 225-234. DOI: <https://doi.org/10.1109/SCAM.2018.00034>.

MITRE. (2017). CWE VIEW: Weaknesses in OWASP Top Ten (2017).Available at: <https://cwe.mitre.org/data/definitions/1026.html> [Accessed 10 August 2024].

Pajankar, A. (2022). *Python Unit Test Automation*. Berkeley, CA: Apress. DOI: <https://doi.org/10.1007/978-1-4842-7854-3_3>

Patel, C., Gadhavi, M., & Patel, A. (2013). *A survey paper on e-learning based learning management Systems (LMS)*. *International Journal of Scientific & Engineering Research (IJSER)*, **4**(6), 171.

Pillai, A.B. (2017). Software Architecture with Python. 1st ed. Birmingham: Packt Publishing. DOI??

Spraul, V.A. (2015). How Software Works: The Magic Behind Encryption, CGI, Search Engines, and Other Everyday Technologies. San Francisco: No Starch Press. DOI??

Deligeorge, A. and Deligeorge, A. (2023). *Omnivision Design*. [online] OmnivisionDesign.com. Available at: [**https://www.omnivisiondesign.com/what-is-version-control-system-vcs-and-its-benefits/**](https://www.omnivisiondesign.com/what-is-version-control-system-vcs-and-its-benefits/) [Accessed 28 Aug. 2024]

Soumya (2019). *Version Control Systems*. [online] Geeks for Geeks. Available at: <https://www.geeksforgeeks.org/version-control-systems/>[Accessed 28 Aug. 2024]

Dewhurst, R. (2013). *Static Code Analysis | OWASP*. [online] Owasp.org. Available at: [**https://owasp.org/www-community/controls/Static\_Code\_Analysis**](https://owasp.org/www-community/controls/Static_Code_Analysis) [Accessed 28 Aug. 2024]

Amazon Web Services, Inc. (2024). *What is an IDE? IDE Explained - AWS*. [online] Available at: [**https://aws.amazon.com/what-is/ide/#:~:text=An%20integrated%20development%20environment%20(IDE**](https://aws.amazon.com/what-is/ide/#:~:text=An%20integrated%20development%20environment%20(IDE) [Accessed 28 Aug. 2024]

Codeacdemy. (n.d.). *What is an IDE?* [online] Available at: [**https://www.codecademy.com/article/what-is-an-ide-ios**](https://www.codecademy.com/article/what-is-an-ide-ios) [Accessed 28 Aug. 2024]

*Nat (2023). What is Flake8, and why should we use it? [online]* Available at: *Python Pandemonium. Available at: https://medium.com/python-pandemonium/what-is-flake8-and-why-we-should-use-it-b89bd78073f2.*

‌