The Design Proposal Document – School Learning Management System

Introduction

This document covers the development of a secured learning management system for schools (Se-LMS). The Se-LMS allows teachers, students and parents to view students' assignments and grades securely. Se-LMS have been promoted in the UK over the last decade (Ofsted, 2013) while the Department for Education (DfE) require schools to provide high-quality remote education (DfE, 2024).

System requirements and specification

Users will log-in to the system using a secure password. The system security can be turned on or off. Users will perform role appropriate Create, Read, Update, and Delate (CRUD) functions. Roles will be managed by the administrator. Appropriate legal frameworks, including GDPR will be observed, such as data minimisation. Minimum system requirements are shown in Appendix 2.

Functional requirements of the application

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

- Administrators will be able to add new courses, update existing courses, assign teachers to courses, register, view, and enrol students or manage their profile.
- Teachers will be able to upload content, view assignments, and provide feedback and grades.
- Students will be able to view content, see their own grades, and submit work.
- Users will require secure account log-ins.

Legal requirements

GDPR regulations regarding its processing of user data will be adhered to:

> ‘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 13’ (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
- Users 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, by the administrator.
- When a student leaves the school, their data will be deleted manually by the administrator, on 1st September.
- Security procedures will be followed to secure data processing, as seen below.

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). Users' attempts at inputting their passwords will also be restricted to protect against Brute Force Attacks (Spraul 2015) and user credentials will be confirmed using Multi Factor Authentication (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: 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: Linters – Flake8.

Static analysis is a process of examining the source code without executing it. This determines weakness, including programming errors, violation of coding standards and security vulnerabilities. Flake8 is a popular Python linter. It checks the code against coding style.

IDE: Visual Studio

Integrated Development Environment (IDE) is a software application which includes tools to efficiently develop code. These typically include a source code editor, testing tools, version control system, and packaging tools. Visual Studio has Python extensions including syntax autocorrecting, unit testing and git operation.

Libraries for encryption and testing.

The application will use the ‘cryptography’ library which ensures 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 through 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.

These strategies help protect against common vulnerabilities (OSWAP, 2021):

1. Broken Access Control Access Control.
2. Cryptographic Failures.
3. Injection.
4. Insecure Design.
5. Security Misconfiguration.
6. Vulnerable and Outdated Components.

7. Identification and Authentication Failures.
8. Software and Data Integrity Failures.
9. Security Loggin and Monitoring Failures.
10. Server-Side Request Forgery.

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)
Damage	Destruction of system, data, or application unavailability.
Reproducibility	Attack is very easy to reproduce.
Exploitability	Attack via a web browser.
Affected Users	All users affected.
Discoverability	Vulnerability found in the web address bar.

Threat Level

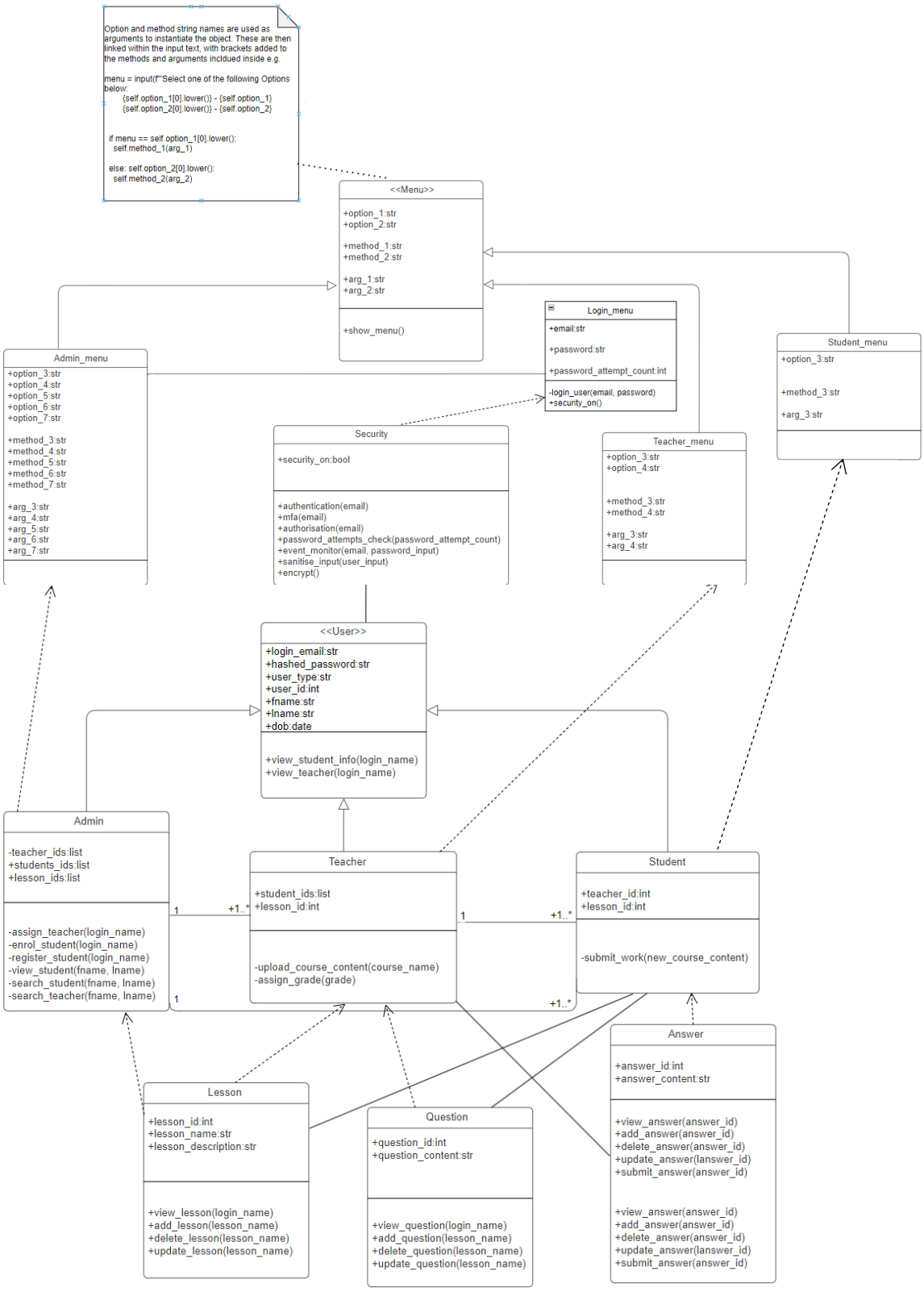
0-10	11-24	25-39	40-50
Low	Medium	High	Critical

Conclusion

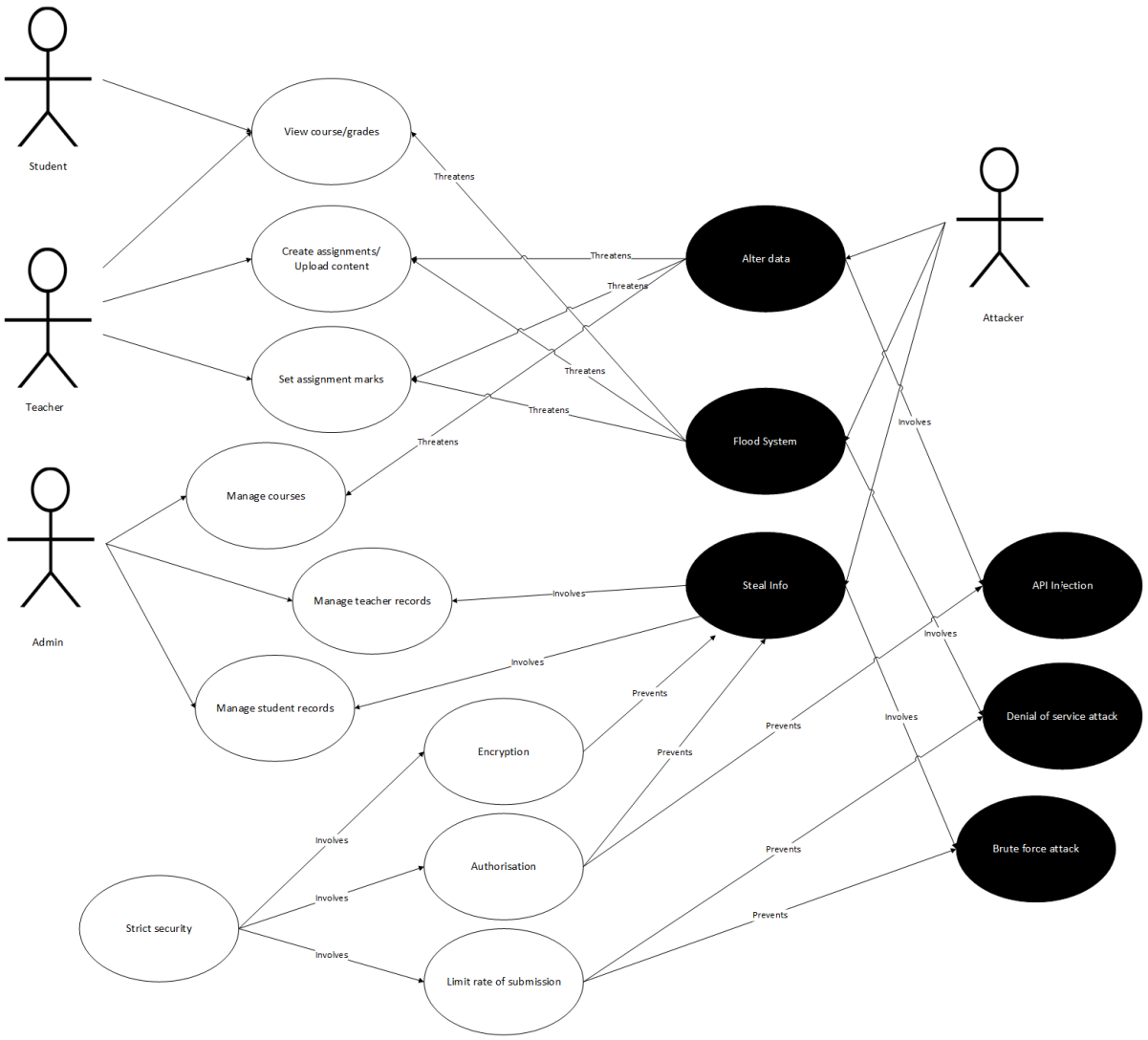
The design proposal will ensure secure development of the application, delivered through a Secure Scrum approach. Threat modelling will ensure security and the legal frameworks, such as GDPR, will be observed.

Appendix 1 – UML Diagrams

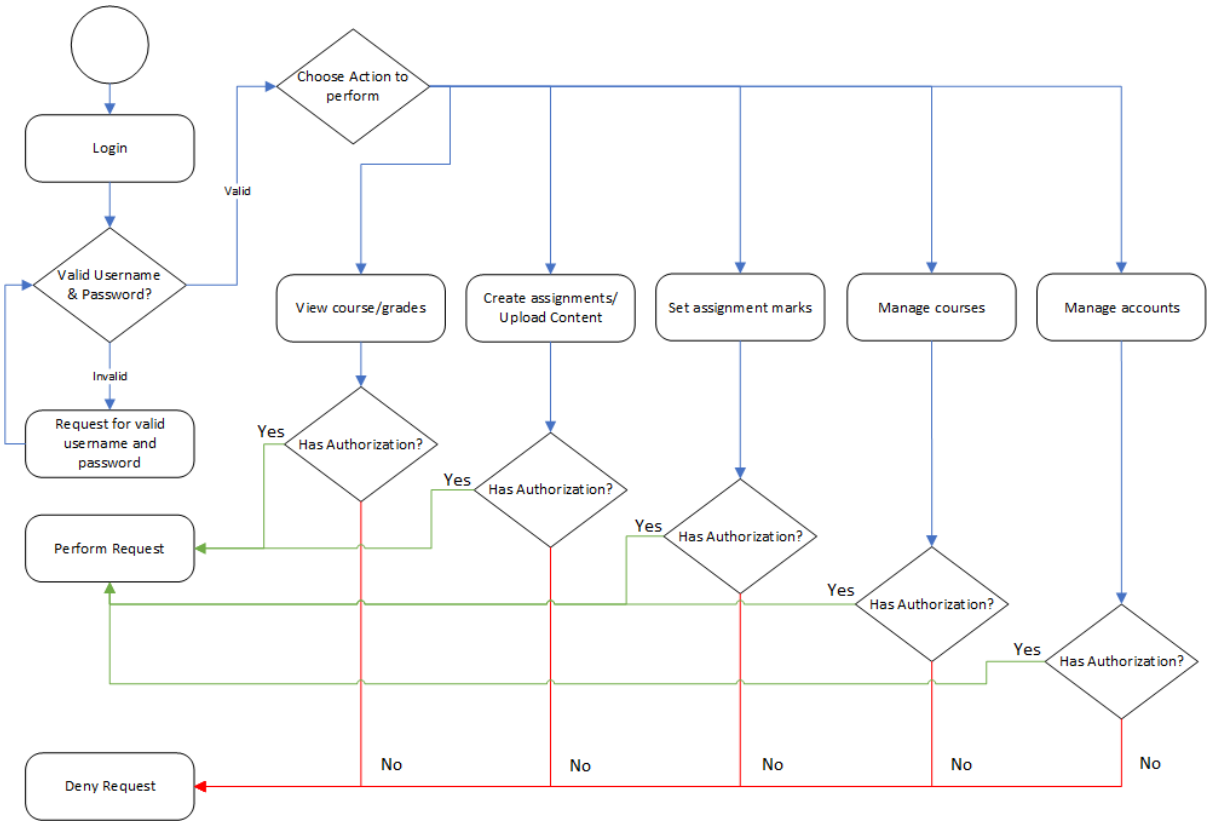
1. Specification Class Diagram



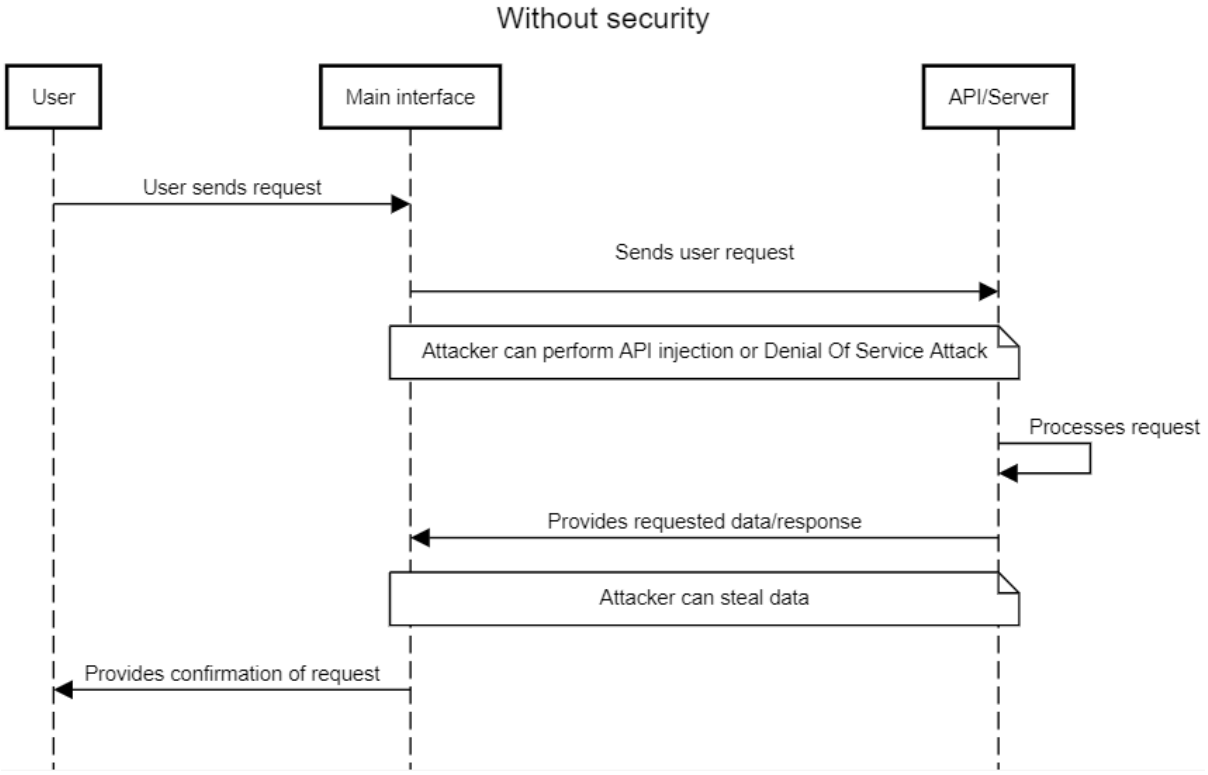
2. Use/Misuse Case Diagram



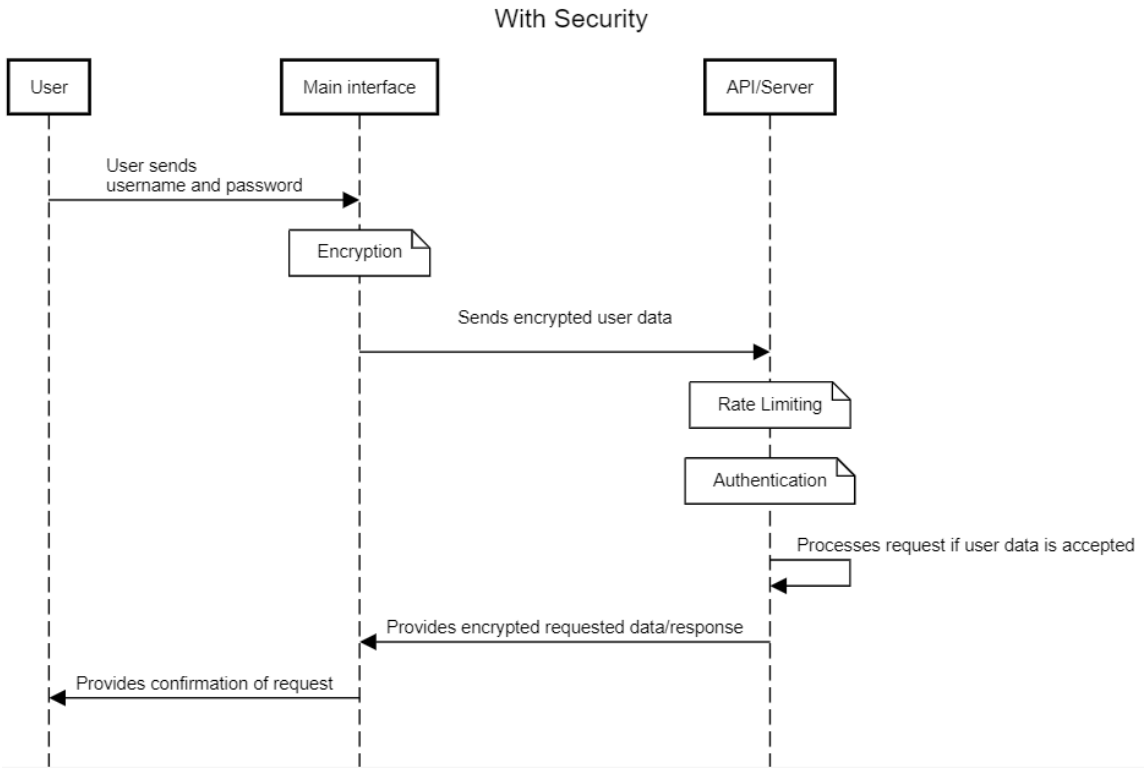
3. Activity Diagram



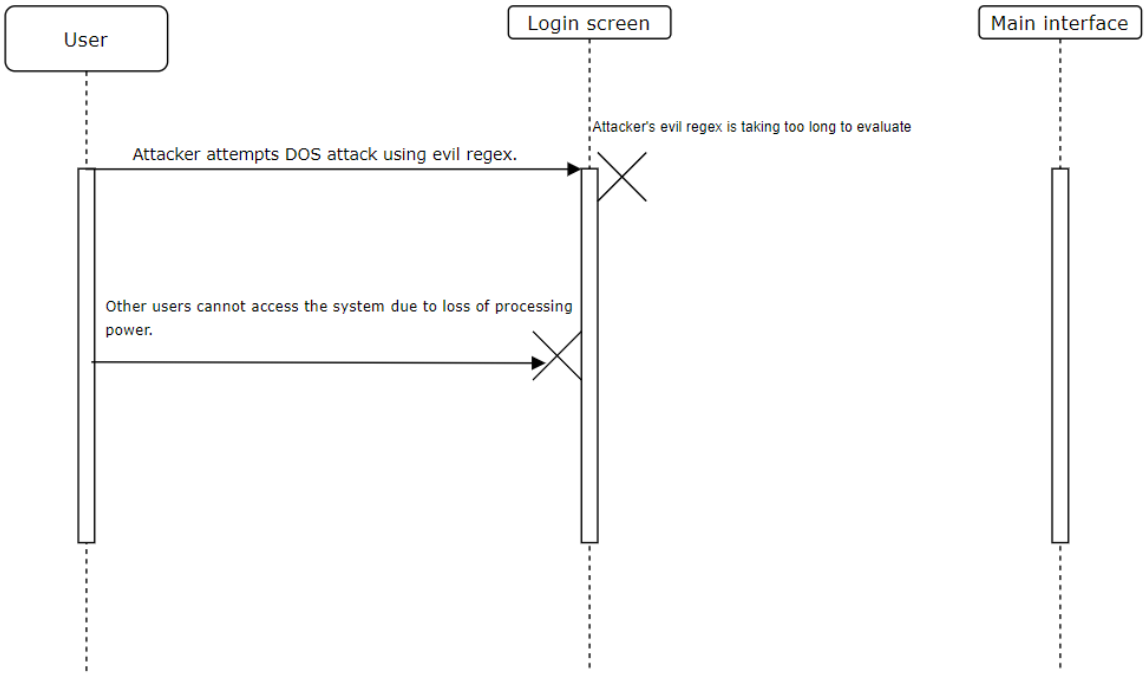
4. Sequence Diagram – without security



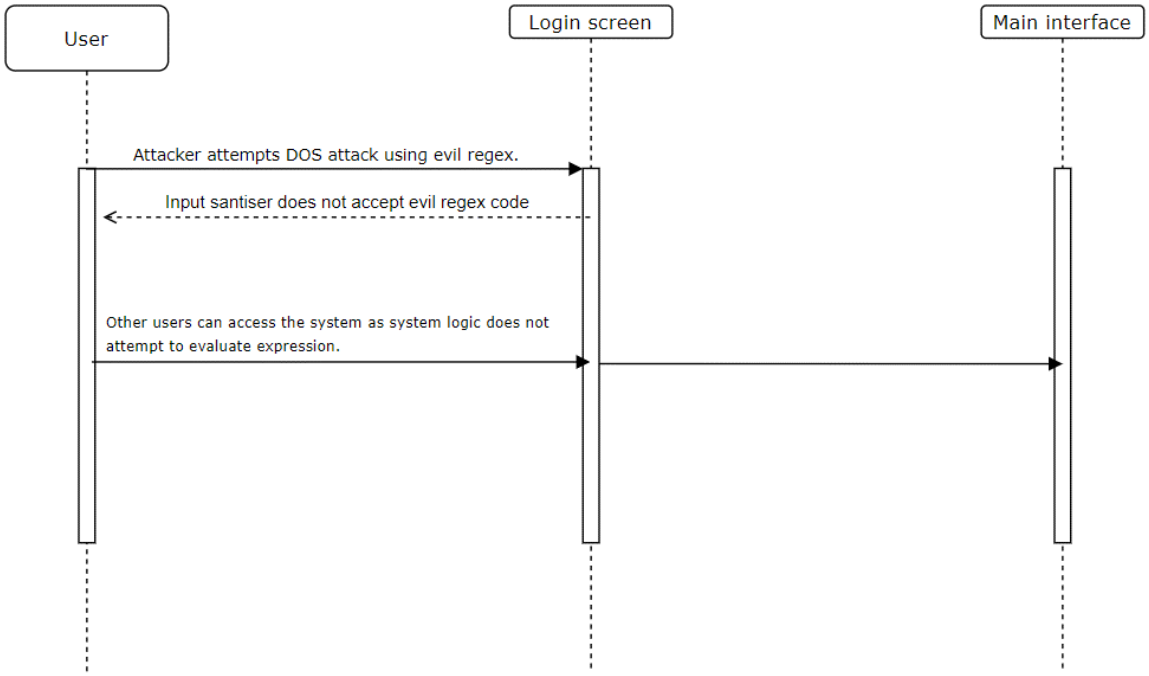
5. Sequence Diagram – with security



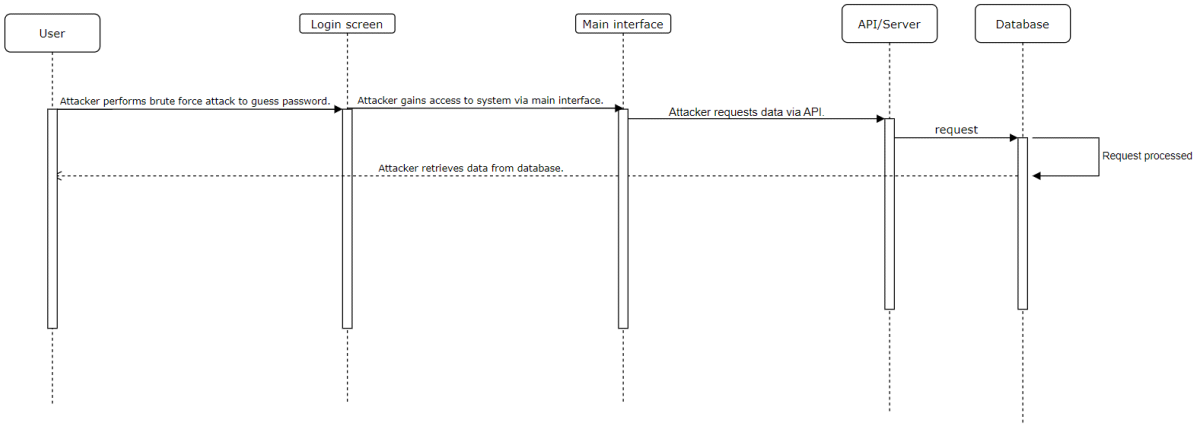
6. Denial of Service (DOS) attack - without security



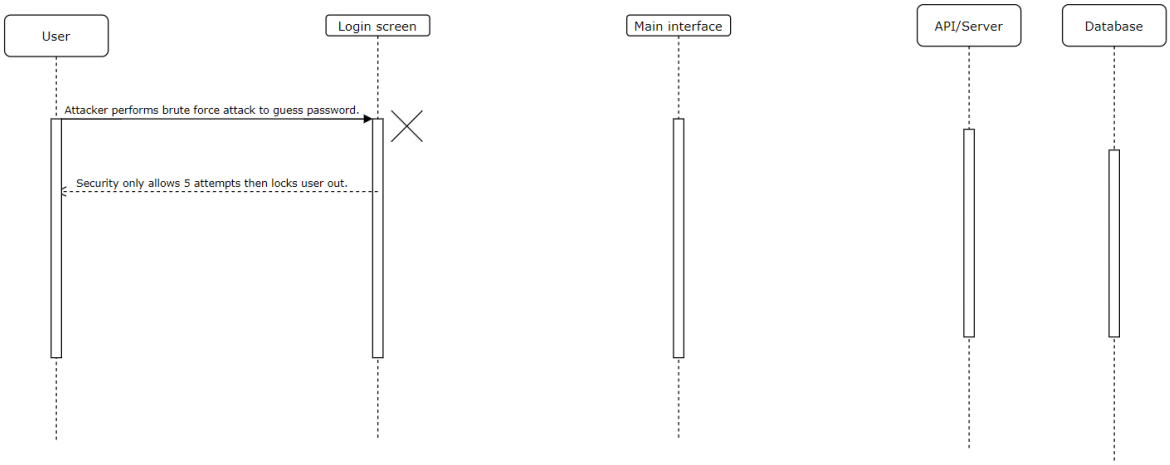
7. Denial of Service (DOS) attack - with security



8. Brute force attack - without security



9. Brute force attack - with security


User
Login screen
Main interface
API/Server
Database
Attacker performs brute force attack to guess password.
Security only allows 5 attempts then locks user out.

Appendix 2 – Minimum System Requirements

The minimum system requirements will be

- Supported Operating System: Windows 10, macOS or Linux.
- Python 3.6 or higher
- Flask
- CPU: Minimum Dual Core
- RAM: Minimum 4GB
- MS Visual Studio

References

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

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

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

Deligeorge, A. and Deligeorge, A. (2023). *Omnivision Design*. [online] OmnivisionDesign.com. Available at: Providing remote education: guidance for schools - GOV.UK (www.gov.uk) [Accessed 28 August 2024]

Department for Education (2024). Providing remote education: guidance for schools. Available from: https://www.gov.uk/government/publications/providing-remote-education-guidance-for-schools/providing-remote-education-guidance-for-schools [Accessed 09 September 2024].

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

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

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.

Larson. E. (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 from: https://cwe.mitre.org/data/definitions/1026.html [Accessed 10 August 2024].

Ofsted (2013). Virtual learning environments: Eleven case studies of effective practice. Available from: https://assets.publishing.service.gov.uk/media/5a756171ed915d7314959844/VLE_e-portfolio_-_case_studies_booklet.pdf [Accessed 9 September 2024].

OSWAP (2021). Top 10 Web Application Security Risks. Available from: OWASP Top Ten | OWASP Foundation [Accessed 9 September 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??

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

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