Development Team Project: Design Document

Group 1: Lauren Pechey, Craig Bourne, Ioannis Maragkos, Timothy Brayshaw

Background and Rationale

The COVID-19 pandemic and global digitalisation have accelerated technology integration in education, emphasising the need for efficient data management systems in schools (Singh et al., 2021). Studies show that integrating technology into schools positively impacts student performance by making learning more accessible, efficient and engaging (Timotheou et al., 2023). Furthermore, increased transparency, parental involvement and reduced administrative workload are shown to contribute to academic success (Bonina et al., 2023). Despite these advantages, many institutions still use fragmented systems or show poor technical literacy, causing inefficiencies (Hillman et al., 2020).

This document proposes a school learning management system application for viewing timetables and grades and allowing administrators to manage student records. The system aims to improve efficiency and student accountability while adhering to GDPR, implementing security requirements and adopting user-friendly principles.

System Requirements and Specifications

The application will be developed in Python, leveraging built-in libraries, cryptographic functions and cross-platform compatibility. It will employ a responsive web interface and client-server architecture for secure data handling. Minimum client requirements include a modern computer with a dual-core processor, 4GB RAM and a web browser with a stable internet connection.

Server requirements are more robust, requiring a modern computer with web server and data storage capabilities, a quad-core processor or higher, a minimum of 8GB RAM and a reliable high-speed internet connection. The server must manage multiple requests, complex processing and be securely located.

Functional Requirements

The system allows users to create accounts with usernames and passwords, assigning specific permissions to administrators, parents, or students. Users can create, read, update and delete (CRUD) records based on their access rights, with administrators having the highest access following the least privilege principle (Ince, 2019).

Sensitive user data must be encrypted before storage to prevent exploitation in breaches, leading to privacy violations and legal issues. The system should allow for deleting sensitive data upon request, or after a certain period, adhering to the right to be forgotten (GDPR, 2018). For security and debugging purposes, the system will log key events and include a toggle to enable or disable security features for testing.

Legal and GDPR

The application will adhere to legal requirements and GDPR guidelines, focusing on three important considerations: consent management, data minimisation and data security (GDPR, 2018). Students, parents, or guardians must provide explicit consent before data collection. To protect children's right to data privacy, they will be informed about how their data is collected, processed and stored (Milkaite & Lievens, 2020).

The application will only process essential personal data, such as name and date of birth, to reduce misuse risks (Delgado-von-Eitzen et al., 2021). Strong security measures, including encryption and access controls will be implemented to prevent data breaches and protect personal information from unauthorised access, maintaining data integrity and confidentiality (Delgado-von-Eitzen et al., 2021).

These measures ensure compliance with legal and GDPR requirements. Detailed security measures will be discussed further to demonstrate how the application safeguards user data and maintains the highest data protection standards throughout its operation.

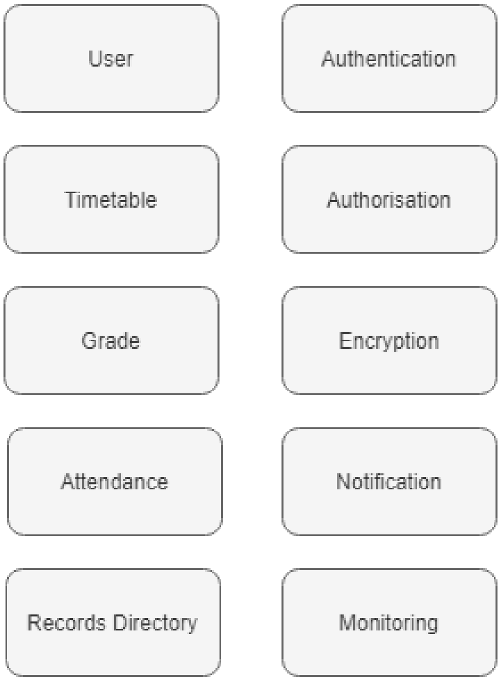
Security Requirements

To ensure robust security in the application, the development process will follow security requirements, which outlines the most critical web application security risks: OWASP Top Ten 2021 (OWASP, 2021).

OWASP Top 10	**Threat**	**Mitigation**
A01:Broken Access Control	Insider Threats	Implement role-based access control to limit access to sensitive data and functions. Monitor user activity and maintain logs to detect suspicious behaviour.
A02:Cryptographic Failures	Data Breach	Encrypt sensitive data such as passwords using a hashing function.
A03:Injection	SQL Injection	Validate and sanitise all user inputs to ensure they meet expected patterns.
A06:Vulnerable and Outdated Components	Software Vulnerabilities	Keep the application, libraries, and server software up to date with the latest security patches. Follow secure coding guidelines to prevent introducing vulnerabilities.
A08:Software and Data Integrity Failures	Social Engineering	Train users and staff on recognising and resisting social engineering tactics. Develop a plan to respond to incidents caused by social engineering attacks.
A09:Security Logging and Monitoring Failures	DOS/DDOS Attack	Monitor external requests and maintain logs to detect suspicious patterns.

Solution Proposal and Methodology

The proposed solution employs several mechanisms to reduce the attack surface, as outlined in the security requirements and misuse case diagram. The software architecture is built on Microservices, with each service addressing a specific need within a defined scope (Tai Ramirez, 2023). The system generates detailed logs for centralised monitoring, enabling performance tracking and security auditing (Tai Ramirez, 2023).



This approach allows for data isolation between services, promoting decoupling. The system encrypts data at-rest in databases at both disk and application levels, focusing on critical operations to balance security and performance (Tai Ramirez, 2023).

For data in-transit, all communication uses Transport Layer Security to ensure confidentiality and integrity, with each microservice having its own private/public key pair and signed certificate (de Almeida & Canedo, 2022; Tai Ramirez, 2023).

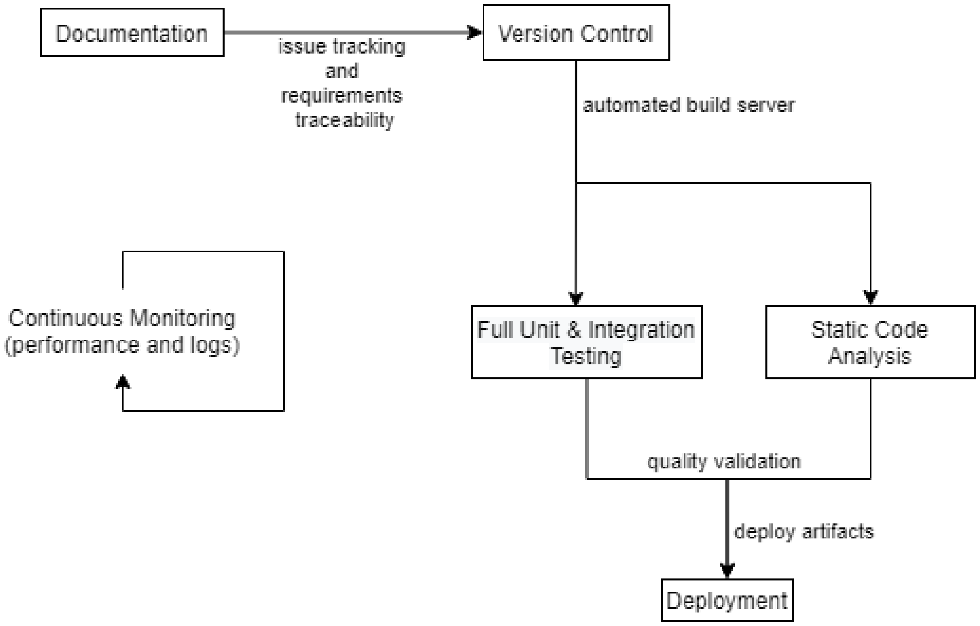
User authentication relies on OAuth2.0 with multi-factor authentication, while authorisation uses role-based access control based on user group permissions (de Almeida & Canedo, 2022; Tai Ramirez, 2023). An API Gateway serves as the initial point for authentication and authorisation, blocking unauthorised access attempts (Tai Ramirez, 2023).

Development follows Agile methodologies with a focus on secure design and Test-Driven Development. A security threat analysis and automated incident response plan are also in place to manage potential security issues.

Tools and Technologies

All tools and libraries used will be the latest stable releases available and feature long term support.

Programming Language	Python 3.12.6 (Python Software Foundation, 2024a) with an emphasis on the Python Standard Library (Python Software Foundation, 2024b) and built-in libraries/modules.
IDEs	VS Code (Microsoft, 2024), PyCharm (JetBrains, 2024).
Version Control	Git (Software Freedom Conservancy, 2024).
Linter	Pylint (Python Software Foundation, 2024c).
CI/CD Pipeline	Jenkins (Jenkins Infrastructure, 2024), Artifactory (JFrog, 2024).
Issue Tracking with Requirements Traceability	GitHub Issues (GitHub, 2024), Jira (Atlassian, 2024).
Development Platform	Platform-agnostic.


Documentation
issue tracking and requirements traceability
Version Control
automated build server
Continuous Monitoring (performance and logs)
Full Unit & Integration Testing
Static Code Analysis
quality validation
deploy artifacts
Deployment

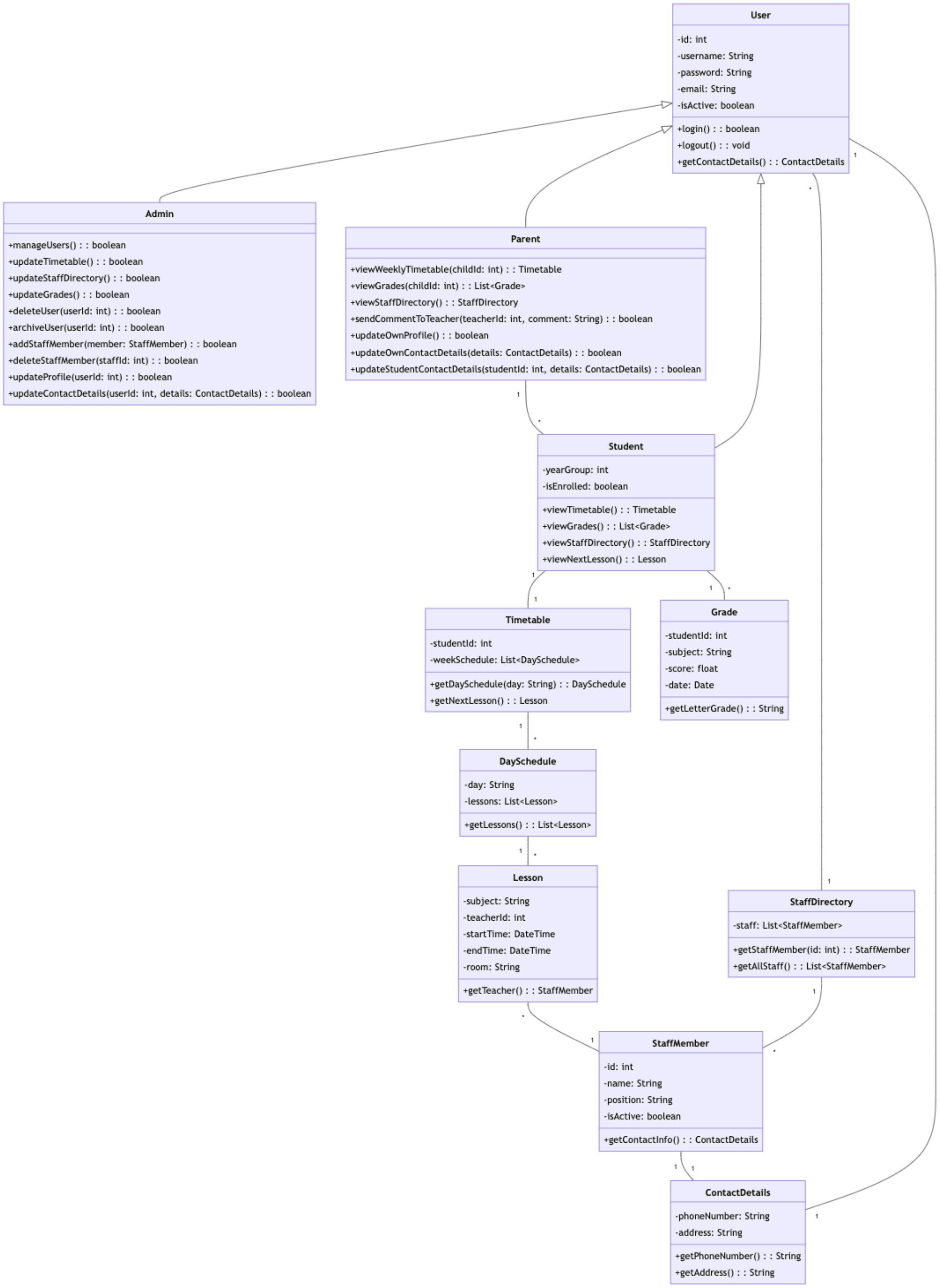
Encryption and Testing Libraries

The application will store user data securely using Python's OS library for database access rights and encryption libraries. bcrypt will be used for password hashing, offering protection against brute force attacks through its computationally slow design and automatic salt management (LeBlanc, J, Messerschmidt, T. 2016). Pylint will serve as the linter during development, ensuring code quality and PEP 8 compliance (Pylint Contributors, ND).

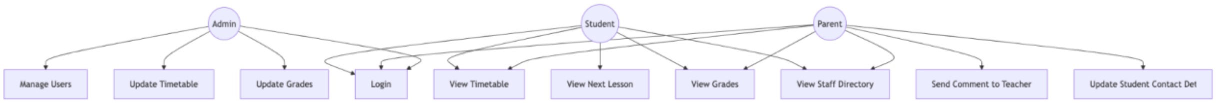
The development process will include unit and integration testing to verify individual components and modules, ensuring the overall reliability and security of the application.

UML Diagrams

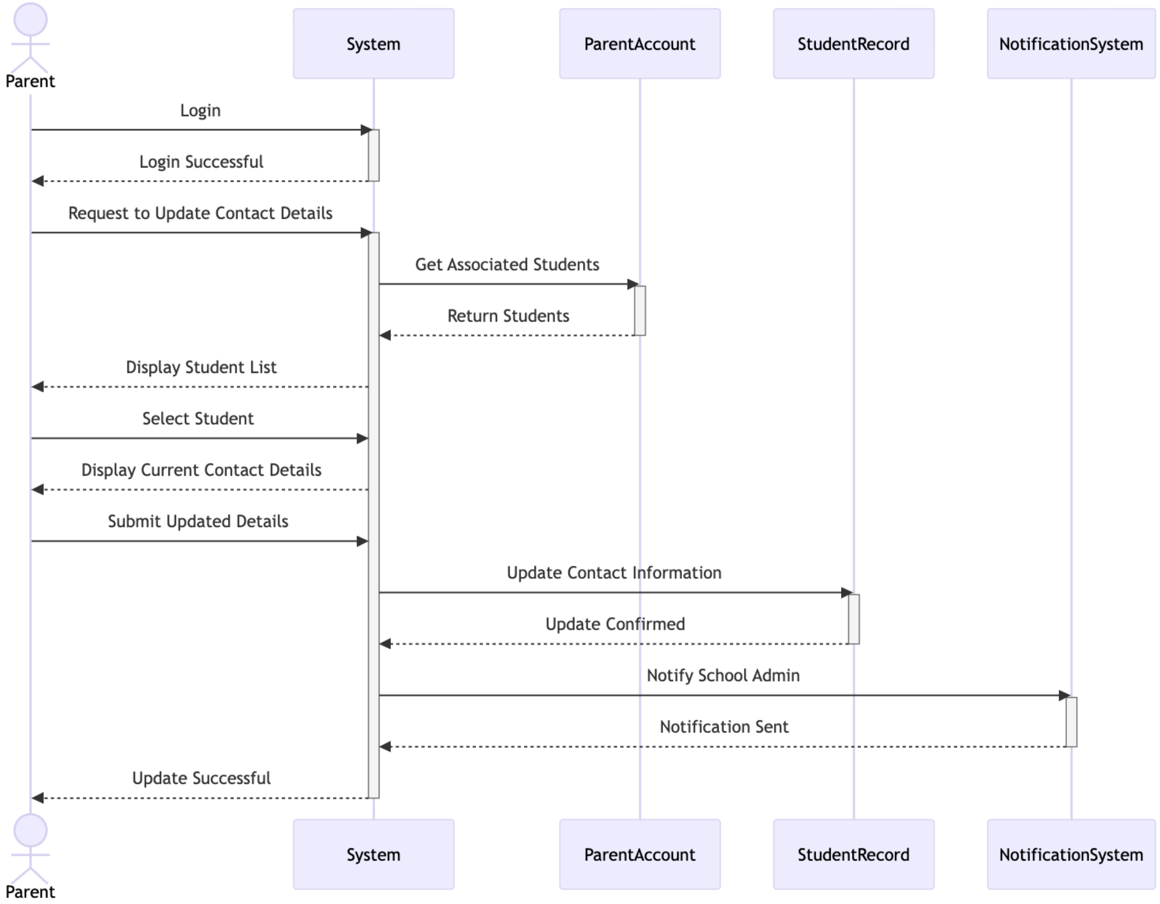
Class Diagram


User
-id: int
-username: String
-password: String
-email: String
-isActive: boolean
+login() : : boolean
+logout() : : void
+getContactDetails() : : ContactDetails
Admin
+manageUsers() : : boolean
+updateTimetable() : : boolean
+updateStaffDirectory() : : boolean
+updateGrades() : : boolean
+deleteUser(userId: int) : : boolean
+archiveUser(userId: int) : : boolean
+addStaffMember(member: StaffMember) : : boolean
+deleteStaffMember(staffId: int) : : boolean
+updateProfile(userId: int) : : boolean
+updateContactDetails(userId: int, details: ContactDetails) : : boolean
Parent
+viewWeeklyTimetable(childId: int) : : Timetable
+viewGrades(childId: int) : : List<Grade>
+viewStaffDirectory() : : StaffDirectory
+sendCommentToTeacher(teacherId: int, comment: String) : : boolean
+updateOwnProfile() : : boolean
+updateOwnContactDetails(details: ContactDetails) : : boolean
+updateStudentContactDetails(studentId: int, details: ContactDetails) : : boolean
Student
-yearGroup: int
-isEnrolled: boolean
+viewTimetable() : : Timetable
+viewGrades() : : List<Grade>
+viewStaffDirectory() : : StaffDirectory
+viewNextLesson() : : Lesson
Timetable
-studentId: int
-weekSchedule: List<DaySchedule>
+getDaySchedule(day: String) : : DaySchedule
+getNextLesson() : : Lesson
Grade
-studentId: int
-subject: String
-score: float
-date: Date
+getLetterGrade() : : String
DaySchedule
-day: String
-lessons: List<Lesson>
+getLessons() : : List<Lesson>
Lesson
-subject: String
-teacherId: int
-startTime: DateTime
-endTime: DateTime
-room: String
+getTeacher() : : StaffMember
StaffDirectory
-staff: List<StaffMember>
+getStaffMember(id: int) : : StaffMember
+getAllStaff() : : List<StaffMember>
StaffMember
-id: int
-name: String
-position: String
-isActive: boolean
+getContactInfo() : : ContactDetails
ContactDetails
-phoneNumber: String
-address: String
+getPhoneNumber() : : String
+getAddress() : : String

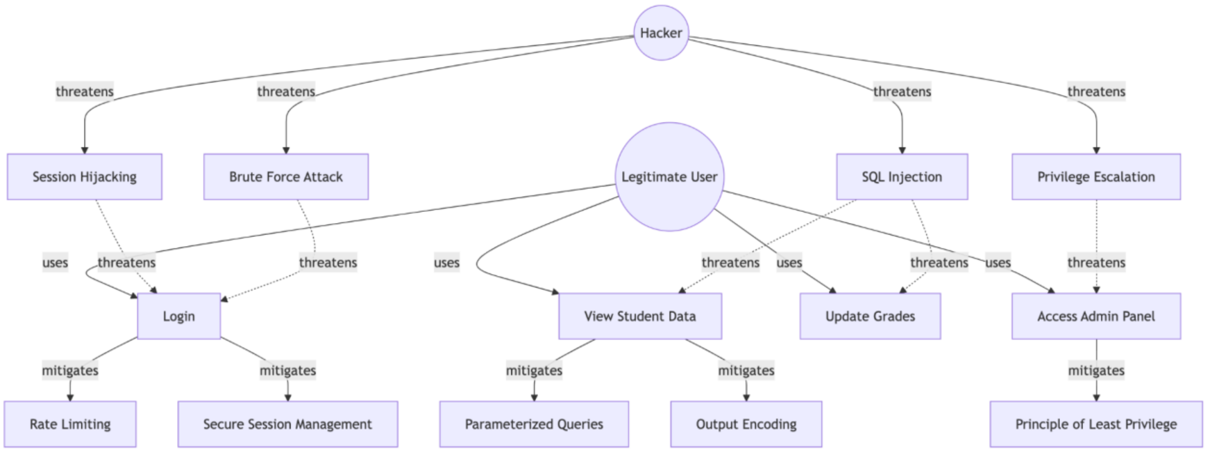
Use Case Diagram



Sequence Diagram for Parent Updating Student Contact Details



Misuse Case Diagram



References

Atlassian (2024) Jira. Available from: https://www.atlassian.com/software/jira [Accessed 26 August 2024].

Bonina, C., Koskinen, K., & Gawer, A. (2023) Digital platforms for development: Foundations and research agenda. *Information Systems Journal* 31(5): 869–902. DOI: https://doi.org/10.1111/isj.12326

de Almeida, M.G., & Canedo, E.D (2022) Authentication and Authorization in Microservices Architecture: A Systematic Literature Review. *Appl. Sci.* 12(6): 3023. DOI: https://doi.org/10.3390/app12063023

Delgado-von-Eitzen, C., Anido-Rifón, A., & Fernández-Iglesias, M. (2021) Application of Blockchain in Education: GDPR-Compliant and Scalable Certification and Verification of Academic Information. *Applied Sciences* 11(10): 1-24. DOI: https://doi.org/10.3390/app11104537

General Data Protection Regulation (GDPR) 2018, c. 2. Available from: https://gdpr-info.eu/ [Accessed 7 September 2024].

GitHub (2024) Issues. Available from: https://github.com/features/issues [Accessed 26 August 2024].

Hillman, T., Rensfeldt, A., & Ivarsson, J. (2020) Brave new platforms: a possible platform future for highly decentralised schooling. *Learning, Media and Technology* 45(1): 7–16. DOI: https://doi.org/10.1080/17439884.2020.1683748

Ince, D. (2019) *A Dictionary of the Internet*. 4th ed. Oxford: Oxford University Press. Available from: https://www-oxfordreference-com.uniessexlib.idm.oclc.org/display/10.1093/acref/9780191884276.001.0001/acref-9780191884276-e-4529 [Accessed 7 September 2024].

Jenkins Infrastructure (2024) Jenkins. Available from: https://www.jenkins.io/ [Accessed 26 August 2024].

JetBrains (2024) PyCharm. Available from: https://www.jetbrains.com/pycharm/ [Accessed 26 August 2024].

JFrog (2024) JFROG ARTIFACTORY. Available from: https://jfrog.com/artifactory/ [Accessed 26 August 2024].

LeBlanc, J. Messerschmidt, T. (2016) *Identity and Data Security for Web Development.* O'Reilly Media Inc. Available from: https://learning.oreilly.com/library/view/identity-and-data/9781491937006/ch02.html#idm139804447923120 [Accessed 1 September 2024].

Microsoft (2024). Visual Studio Code. Available from: https://code.visualstudio.com/ [Accessed 26 August 2024].

Milkaite, I., & Lievens, E. (2020) Child-friendly transparency of data processing in the EU: from legal requirements to platform policies. *Journal of Children and Media* 14(1): 5-21. DOI: https://doi.org/10.1080/17482798.2019.1701055

OWASP. (2021) OWASP Top Ten Web Application Security Risks. Available from: https://owasp.org/www-project-top-ten/ [Accessed 4 September 2024].

Pylint Contributors. (n.d.) *Pylint documentation*. Available from: https://pylint.pycqa.org/en/latest/index.html# [Accessed 7 September 2024].

Python Software Foundation. (n.d.) Frequently Asked Questions about Python. Available from: https://docs.python.org/3/faq/general.html [Accessed 1 September 2024].

Python Software Foundation (2024a) python. Available from: https://www.python.org/ [Accessed 26 August 2024].

Python Software Foundation (2024b) The Python Standard Library. Available from: https://docs.python.org/3/library/index.html [Accessed 26 August 2024].

Python Software Foundation (2024c) pylint. Available from: https://pypi.org/project/pylint/ [Accessed 26 August 2024].

Red Hat (2024) What is CI/CD? Available from: https://www.redhat.com/en/topics/devops/what-is-ci-cd [Accessed 26 August 2024].

Singh, A., Sharma, S., & Paliwal, M. (2021) Adoption intention and effectiveness of digital collaboration platforms for online learning: The Indian students' perspective. *Interactive Technology and Smart Education* 18(4): 493-514. DOI: https://doi.org/10.1108/ITSE-05-2020-0070

Software Freedom Conservancy (2024) git. Available from: https://git-scm.com/ [Accessed 26 August 2024].

Tai Ramirez, E. (2023) *A Framework to Build Secure Microservice Architecture*. Doctor of Philosophy. The University of Texas at El Paso. Available from: https://login.uniessexlib.idm.oclc.org/login?url=https://www.proquest.com/dissertations-theses/framework-build-secure-microservice-architecture/docview/2825753125/se-2 [Accessed 31 August 2024].

Timotheou, S., Miliou, O., Dimitriadis, Y., Sobrino, S., Giannoutsou, N., Cachia, R., Monés, A., & Ioannou, A. (2023) Impacts of digital technologies on education and factors influencing schools' digital capacity and transformation: A literature review. *Education and Information Technologies* 28(1): 6695–6726. DOI: https://doi.org/10.1007/s10639-022-11431-8