|  |  |  |
| --- | --- | --- |
|  | Design Document | |
|  |  | Team:   * Arianna Poverini * Imraan Meyer * Mobeen Ali * Yanani Yasotharan |
|  |  |
|  | Secure Software Development |

**Table of Contents**

[Introduction 2](#_Toc200306740)

[Aims and Objectives 2](#_Toc200306741)

[Requirements and Assumptions 2](#_Toc200306742)

[Functional Requirements 2](#_Toc200306743)

[Non-Functional Requirements 3](#_Toc200306744)

[Assumptions 3](#_Toc200306745)

[Security Risks and Mitigation (OWASP Framework) 3](#_Toc200306746)

[A10:2021 Insufficient Logging & Monitoring. 3](#_Toc200306747)

[A03:2021 Data Injection. 4](#_Toc200306748)

[A07:2021 Identification and Authentication Failures. 4](#_Toc200306749)

[A05:2021 Security Misconfiguration. 4](#_Toc200306750)

[UML Diagrams 6](#_Toc200306751)

[Misuse Activity Diagram 7](#_Toc200306752)

[Tools and Libraries 8](#_Toc200306753)

[Project Management and Solo Development Approach 8](#_Toc200306754)

[Project Timeline 10](#_Toc200306755)

[References 11](#_Toc200306756)

Introduction

This platform, built in Python and accessed via a Command Line Interface (CLI), will be a secure e-commerce. This type of online retail platform reflects the evolving digital marketplace where consumers purchase goods, services, and financial products online (D’Adamo et al., 2021). Incorporating secure coding, object-oriented design, and General Data Protection Regulations (GDPR) compliance, it will enable user registration, authentication, and full CRUD - Create, Read, Update, and Delete - operations on product data. E-commerce now spans national and international sellers, collaborative services, and financial transactions, reshaping global consumer behaviour. It offers businesses market expansion without physical storefronts however, it introduces challenges notably in cybersecurity, privacy, and sustainability (D’Adamo et al., 2021).

Aims and Objectives

The aim is to engineer a secure e-commerce capable of resisting common OWASP-identified threats. Addressing e-commerce security issues requires a multi-faceted approach: as pointed out by D’Adamo (2021), customers are primarily concerned with privacy loss, sharing personal data, and payment security. For e-commerce companies, it's crucial to prioritise data protection when setting up an application, designing secure products and services, implementing systematic risk management.

The primary objective is then to address security concerns through robust data handling and secure interactions, ensuring trust and efficiency in a growing digital economy.

One way of meeting the objectives is to identify threats via the OWASP framework. This will ensure the project uses the best practices for security.

Requirements and Assumptions

Functional Requirements

* Enable authenticated users to perform CRUD operations on system entities through a command-line interface using JSON files as the primary data storage.
* Secure account registration and login functionality must be implemented using password-based authentication, adhering to modern security standards, with credentials stored in a hashed format within JSON documents.
* A command-line toggle should enable or disable system security features for testing purposes.

Non-Functional Requirements

* The system should comply with GDPR to ensure data protection and user privacy through the pseudonymization of personally identifiable information in records.
* Object-orientated programming and a modular design pattern should be used for maintainability and scalability.
* Data should be encrypted both at rest and in transit to reduce vulnerability.
* The command-line interface should be intuitive, providing clear instructions and feedback to promote usability and minimize user errors.

Assumptions

* Users will interact with the system locally through a terminal with Python pre-installed.
* bcrypt alone is sufficient for password security (no session management is needed in the CLI).

Security Risks and Mitigation

In alignment with the OWASP Top Ten 2021 framework (OWASP, 2021), we have identified and addressed four significant security threats to protect the integrity and confidentiality of our system.

A10:2021 Insufficient Logging & Monitoring.

* Risk of undetected breaches and delayed incident response.
* Failed login attempts may indicate brute force attacks.
* Mitigation: Use real-time monitoring to detect anomalies and potential attacks.
* Mitigation: Implement account lockout mechanisms for repeated failures.
* Regularly review logs to ensure accountability and security.
* Ensure log data is protected to maintain user privacy.

A03:2021 Data Injection.

* All third-party data is validated using strict schema validation to prevent injection of untrusted input
* Output is encoded and sanitised to block the execution of malicious scripts.
* External communication is handled securely via a centralized APIManager module to reduce injection risks.
* Secure transport Hypertext Transfer Protocol Secure (HTTPS), API keys, and rate limiting are enforced to safeguard API endpoints.

A07:2021 Identification and Authentication Failures.

* Enforce robust password policies.
* Use bcrypt for secure password hashing.
* Mandate passwords of at least 12 characters.
* Integrate Two-Factor Authentication (2FA) via PyOTP.
* Implement mechanisms to limit login attempts.

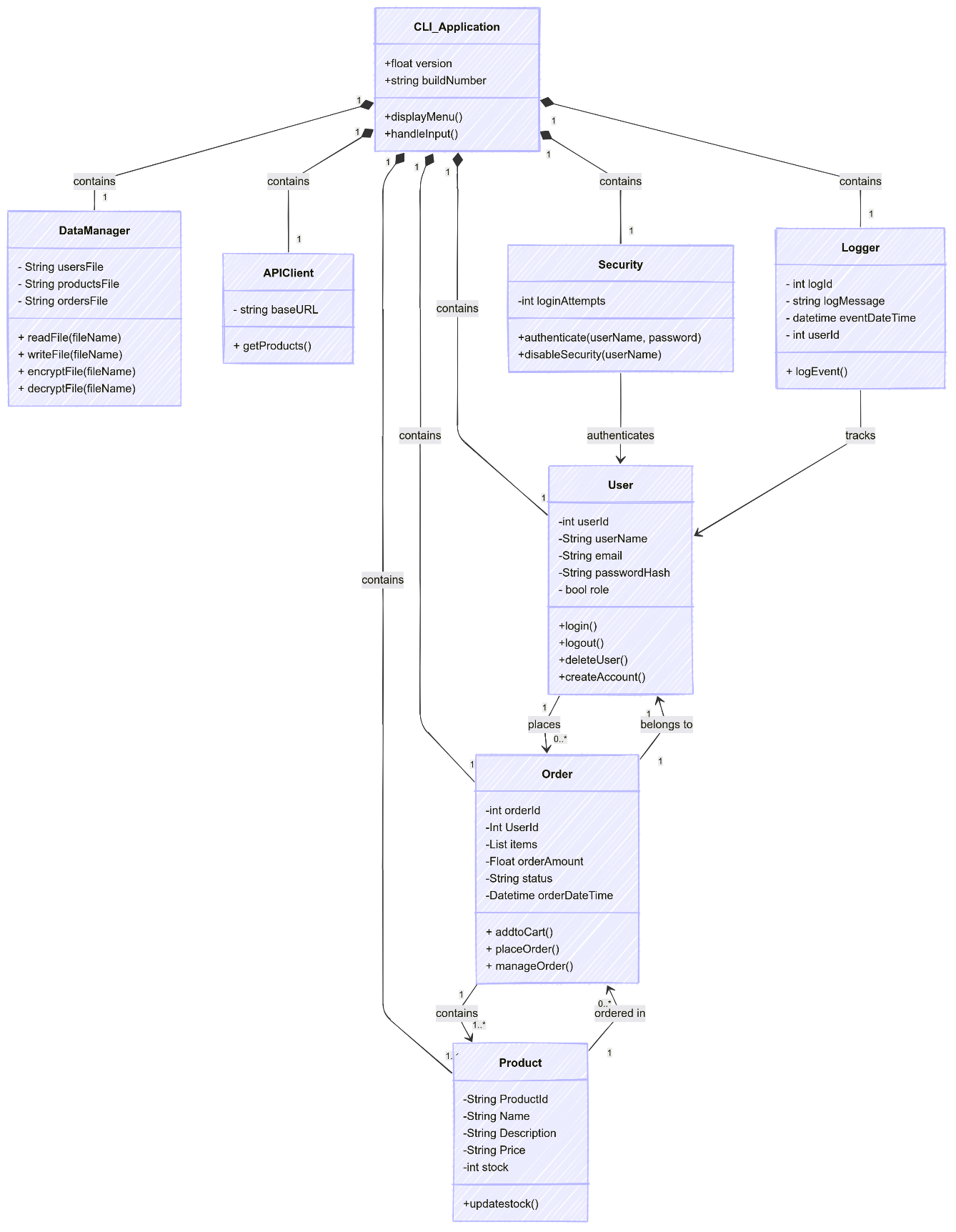
A05:2021 Security Misconfiguration.

* Avoid default credentials and verbose errors through secure defaults.
* Disable debugging features in production.
* Restricting admin-only commands by using role-based access control.
* Sanitizing error messages.
* A secure configuration file and role-based access will be implemented.
* A ‘SecurityManager’ class controlling operational security states and limiting the security toggle to admin users only.

UML Diagrams

**Class Diagram**

Figure 1 - Shows modular OOP design including CLI controller, user/product/order entities, and support modules for logging, security, and JSON-based persistence.



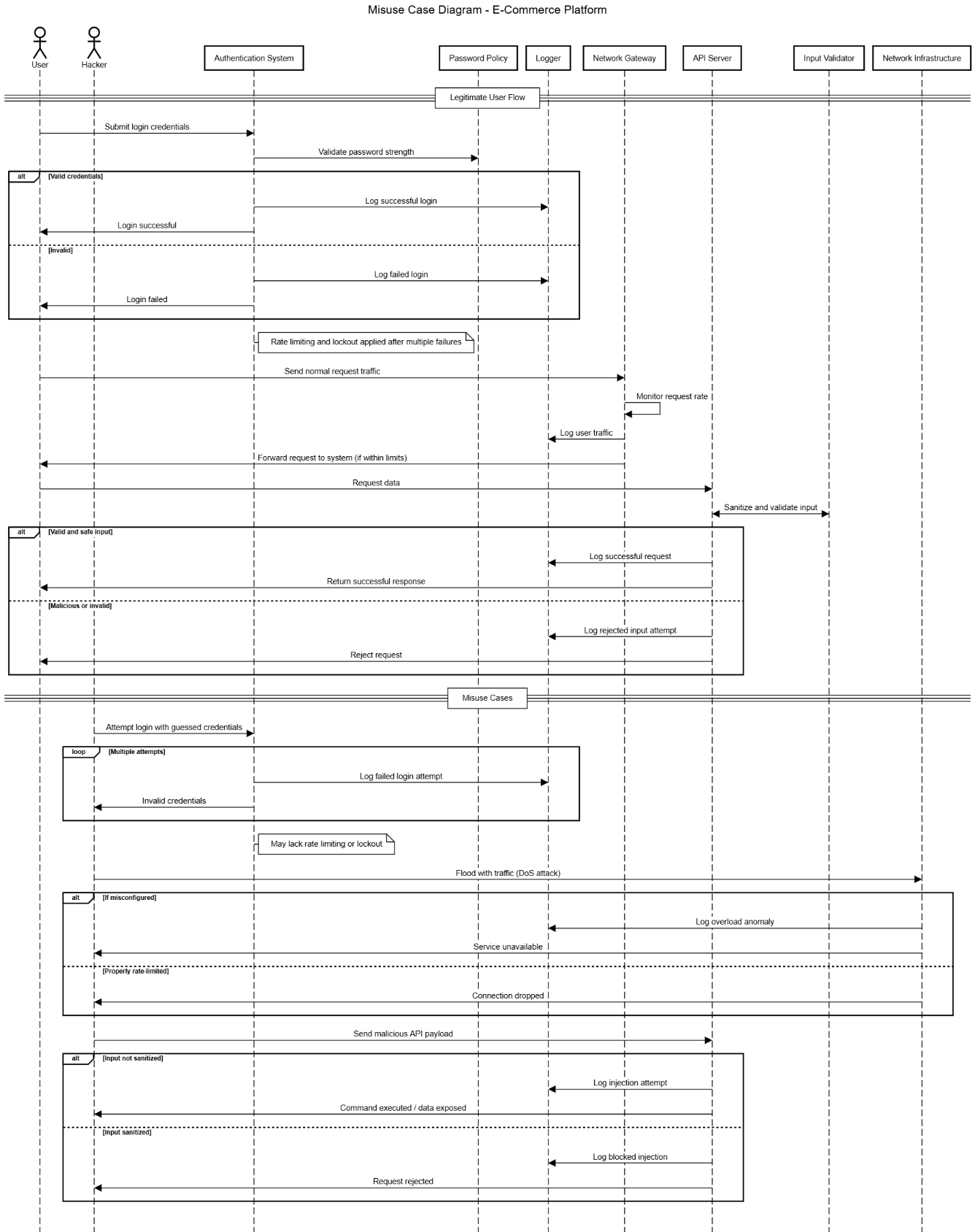
Misuse Activity Diagram

Figure 2 - Models both legitimate and malicious system interactions across components like authentication, API, and network. Highlights OWASP threats A03, A05, A07, and A10 with embedded mitigations.

Tools and Libraries

The following Python libraries will be utilised:

* Click: This library will be used to create the Command Line Interface (CLI). Click simplifies the creation of CLI applications by allowing developers to define commands and options with decorators, making it easy to manage user inputs and provide intuitive command structures. (bcrypt (n.d.))
* PyOTP: For enhanced security, PyOTP will be integrated to implement two-factor authentication (2FA). It generates and verifies time-based one-time passwords that add an extra layer of user authentication during login. (PyOTP (n.d.))
* Pytest: To ensure code quality and reliability, Pytest will serve as the testing framework. It offers features for writing unit, integration, and system tests with support for assertions and fixtures, enabling comprehensive test coverage throughout development. (pytest (n.d.))
* Keyring: provides a straightforward user interface for securely storing and retrieving credentials via the system's keyring service. It enhances application security by safeguarding sensitive data by supporting several backends and enabling developers to handle private data, like as API keys and passwords, without hardcoding. (Zhang, K. (n.d.))

Project Management and Solo Development Approach

Following submission of the team design document, each team member will individually implement the secure CLI-based application using a structured solo development process. This aligns with the solo developer models described by Pagotto et al. (2016) and Moyo and Mnkandla (2020), which emphasise modular design, disciplined task management, and secure software engineering practices.

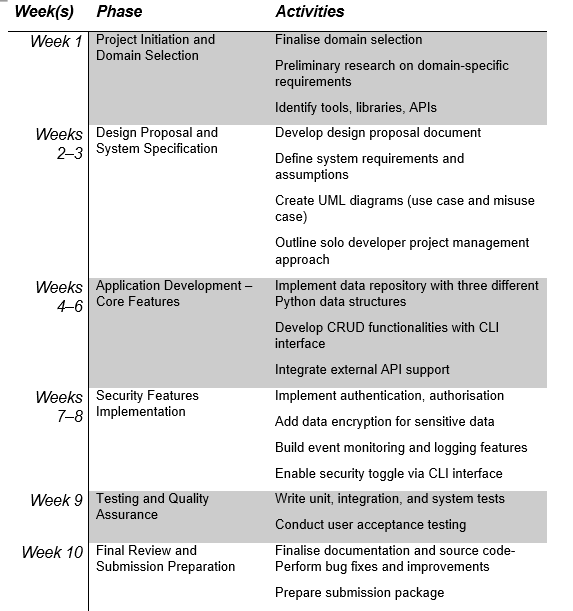
To ensure effective and traceable implementation, each developer will undertake the following:

* Initialize a version-controlled repository using Git, with regular commits documenting feature progress.
* Define a personal Kanban board (e.g., using GitHub Projects) to track and prioritise development tasks.
* Follow test-driven development principles by writing unit tests before implementing features.
* Prioritize implementation of security-related modules early in the lifecycle, including authentication, encryption, and logging.
* Integrate a third-party API to meet functional requirements and simulate real-world interactions.
* Maintain modular, object-oriented code to facilitate testing, debugging, and future extensibility.
* Conduct integration and user acceptance testing upon completion of core features.
* Maintain brief development logs to record decisions, issues encountered, and resolutions.

This approach promotes accountability, security, and maintainability while ensuring the final system aligns with both academic expectations and the project’s technical objectives.

Project Timeline

This timeline lays out a structured approach to deliver the application within 10 weeks, ensuring thorough design, development, and testing phases.



References

D’Adamo, I., González-Sánchez, R., Medina-Salgado, M.S. and Settembre-Blundo, D. (2021) ‘E-Commerce Calls for Cyber-Security and Sustainability: How European Citizens Look for a Trusted Online Environment’, Sustainability, 13(12), p.6752. <https://doi.org/10.3390/su13126752>

Grassi, P.A., Garcia, M.E. and Fenton, J.L., 2017. Digital Identity Guidelines. NIST special publication, 800, pp.63-3. Available at: <https://doi.org/10.6028/NIST.SP.800-63b> (Accessed 29 May 2025).

Moyo, M. and Mnkandla, E. (2020) ‘A framework for solo software development’, 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), IEEE, pp. 1–7. <https://doi.org/10.1109/icABCD49160.2020.9183763>

Pagotto, E., Spínola, R.O. and Seaman, C. (2016) ‘Characterizing solo software development’, Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–10. <https://doi.org/10.1145/2961111.2962635>

OWASP (2021) OWASP Top Ten: The Ten Most Critical Web Application Security Risks. Available at: <https://owasp.org/Top10> (Accessed: 4 June 2025).

bcrypt (n.d.) bcrypt. Python Package Index. Available at: <https://pypi.org/project/bcrypt/> (Accessed: 4 June 2025).

PyOTP (n.d.) Python One-Time Password Library. GitHub. Available at: <https://github.com/pyauth/pyotp> (Accessed: 4 June 2025).

Click (n.d.) Click – Python Package for Creating Command Line Interfaces. Python Package Index. Available at: <https://pypi.org/project/click/> (Accessed: 4 June 2025).

pytest (n.d.) pytest – Python testing tool. Python Package Index. Available at: <https://pypi.org/project/pytest/> (Accessed: 4 June 2025).

GDPR (n.d.a) General Data Protection Regulation (EU). Available at: <https://gdpr-info.eu/> (Accessed: 4 June 2025).

Mermaid (n.d.) Mermaid – Markdown-inspired diagramming tool. Available at: <https://mermaid.js.org/> (Accessed: 5 June 2025).

SequenceDiagram.org (n.d.) Create UML sequence diagrams in the browser. Available at: <https://sequencediagram.org/> (Accessed: 5 June 2025).

Zhang, K. (n.d.). keyring: Store and access your passwords safely. [online] PyPI. Available at: <https://pypi.org/project/keyring/> (Accessed: 7 June 2025).