Secure Software Development: Design Document

Team CURE

Document Scope

This document details the design of a system to facilitate the operation of an online store. The scope of the document includes:

- System and functional requirements
- High-level system design
- Security risks and countermeasures
- Design patterns and data modelling
- Tools and libraries

System Requirements

- Python 3.8 +
- PyCryptodome 3.19.0 +
- AES Cipher 6.6.0 +

Functional Requirements

The functional requirements for the store have been identified following the review of multiple academic works on designing similar software (William et al., 2020; Huang,

2022; Tenzin et al, 2022). This system will be designed for use by two actors, with individual requirements, as detailed in Figure 1 and Figure 3.

		Actors	
		Customer	Store Administrator
Actions	Create account	•	•
	Add product to catalogue		•
	Browse products	•	•
	Modify/delete products		•
	Create order	•	
	Add payment method	•	
	Modify/delete payment method	•	
	Modify/delete own account	•	•
	Modify/delete other account		•

Figure 1 – Functional requirements by user type

In addition to these application specific functional requirements, relevant legislative and best practice requirements will apply, such as the General Data Protection Regulation (GDPR) and the Open World Application Security Project (OWASP) Secure Coding Practices. Where user agreement is required, such as for GDPR (European Union, 2016), the user will be presented with the relevant information and asked to continue through the user view, as shown by the activity diagram in Figure 2.

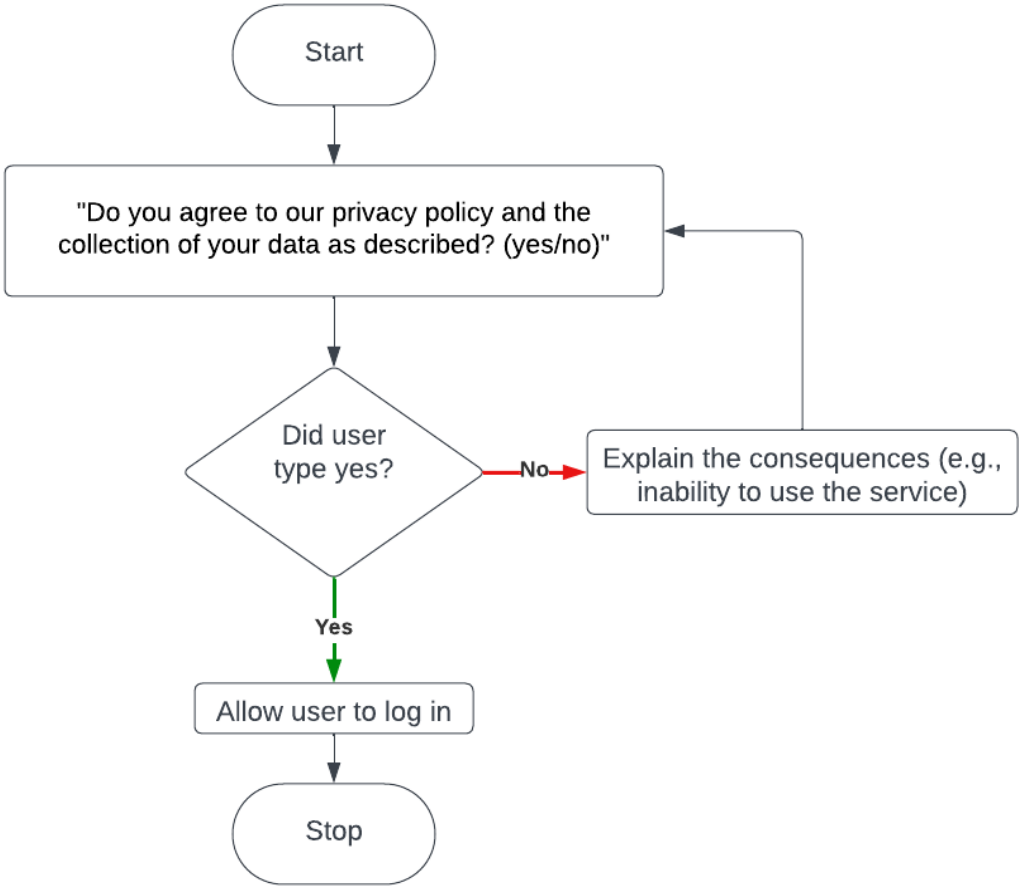

Figure 2 - Requesting customer acceptance of data usage for GDPR

High-Level System Design

The use case diagram shown in Figure 3 shows how each user type will interact with the system.

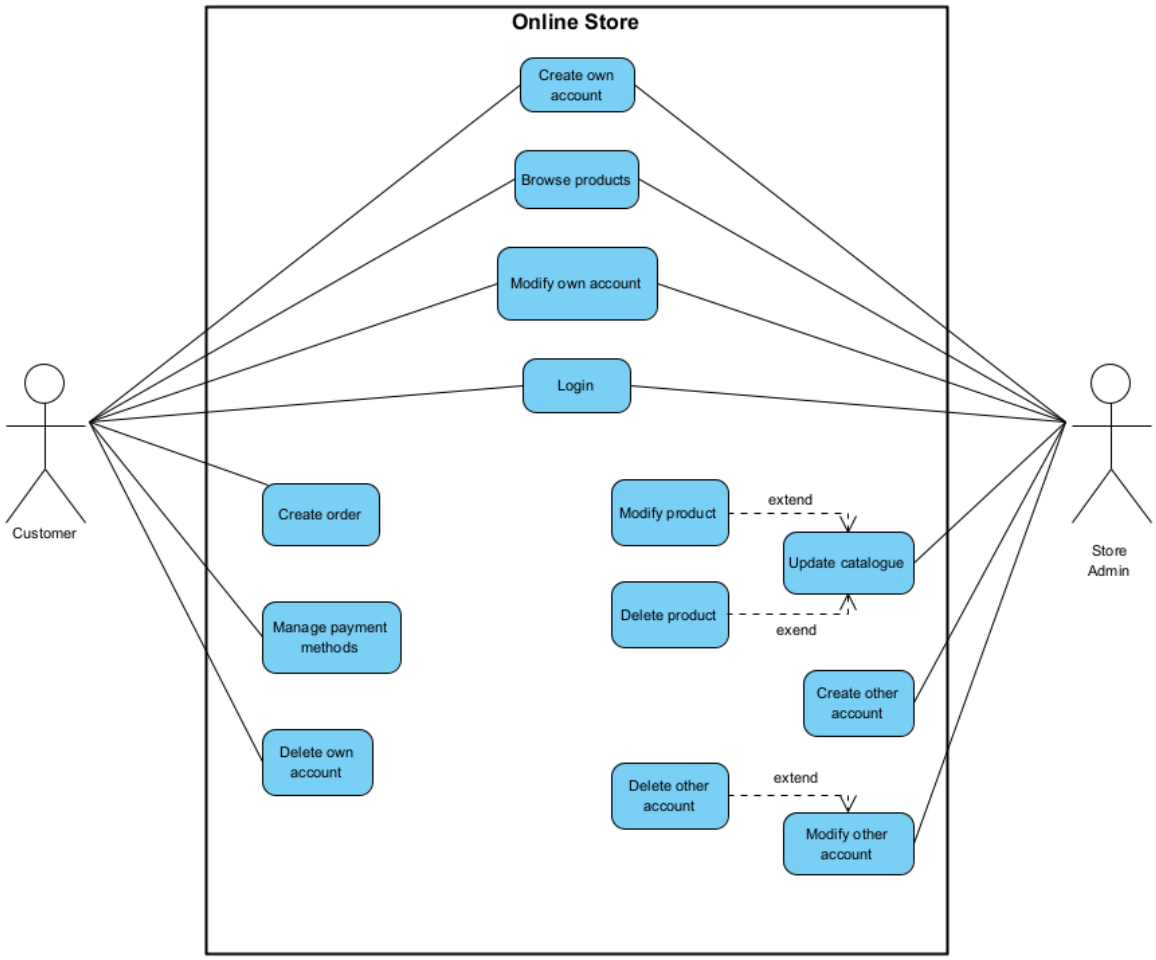

Figure 3 - System use case diagram

Both user levels will use the same authentication process for accessing their account. Depending on the level of the user, the available actions once logged in will differ.

Customers will be able to select available products to construct an order. They will also have the option to modify their account details or delete their own account entirely.

The *store administrator* will be able to add, modify or delete products from the catalogue. They will also be able to modify or create new accounts of either the customer or store administrator level. The store administrator is not able to delete their own account, but is able to delete the accounts of other store administrators. This ensures that there always remains at least one active store administrator account at any time.

Security Risks and Countermeasures

As Pillai (2017) states, security is an aspect to be considered right from the inception of the design and development of a program; it is not an afterthought. Threat modelling should be performed to anticipate security threats to the system from the foundation. The online store has been designed with consideration to the potential security risks and the implementation of appropriate countermeasures to protect sensitive user information, prevent unauthorised access, and ensure the integrity of the system (Rehim, 2016). Within the system design, it is demonstrated how these risks can be mitigated through sanitising and validating user inputs, with reference made to the OWASP Top Ten list of threats which developers should have awareness of (OWASP, 2021). Details of the perceived threats and countermeasures are listed below:

- Broken Access Control
    - Implement proper access controls and authorisation mechanisms.
    - Enforce RBAC or ABAC to restrict user permissions.
    - Regularly review and validate access control rules.

- Cryptographic Failures
    - Use established encryption libraries and algorithms.
    - Follow key management best practices.
    - Apply encryption to sensitive data at rest and in transit.

- Injection
    - Use parameterised queries and prepared statements to prevent SQL injection.
    - Implement input validation and sanitisation techniques.
    - Avoid string concatenation for constructing queries or executing commands.

- Identification and Authentication Failures
    - Implement strong authentication mechanisms.
    - Encourage MFA usage.
    - Implement account lockouts and regular account reviews (OWASP, 2021).

General Security measures that are considered for the online store are:

- Up-to-date software dependencies.
- Validated and sanitised user input.
- Secure session management.
- Logging and monitoring system activities.
- Conducting regular security assessments.

To model these threats and mitigations, the misuse case diagram (Pauli & Xu, 2005) shown in Figure 4 was produced. The identified risks (black use cases) are linked to the expected activities for users of the system (blue use cases). Mitigations are also included (green use cases) to show which expected activities will be protected and which risks will be mitigated against by the relevant countermeasure.

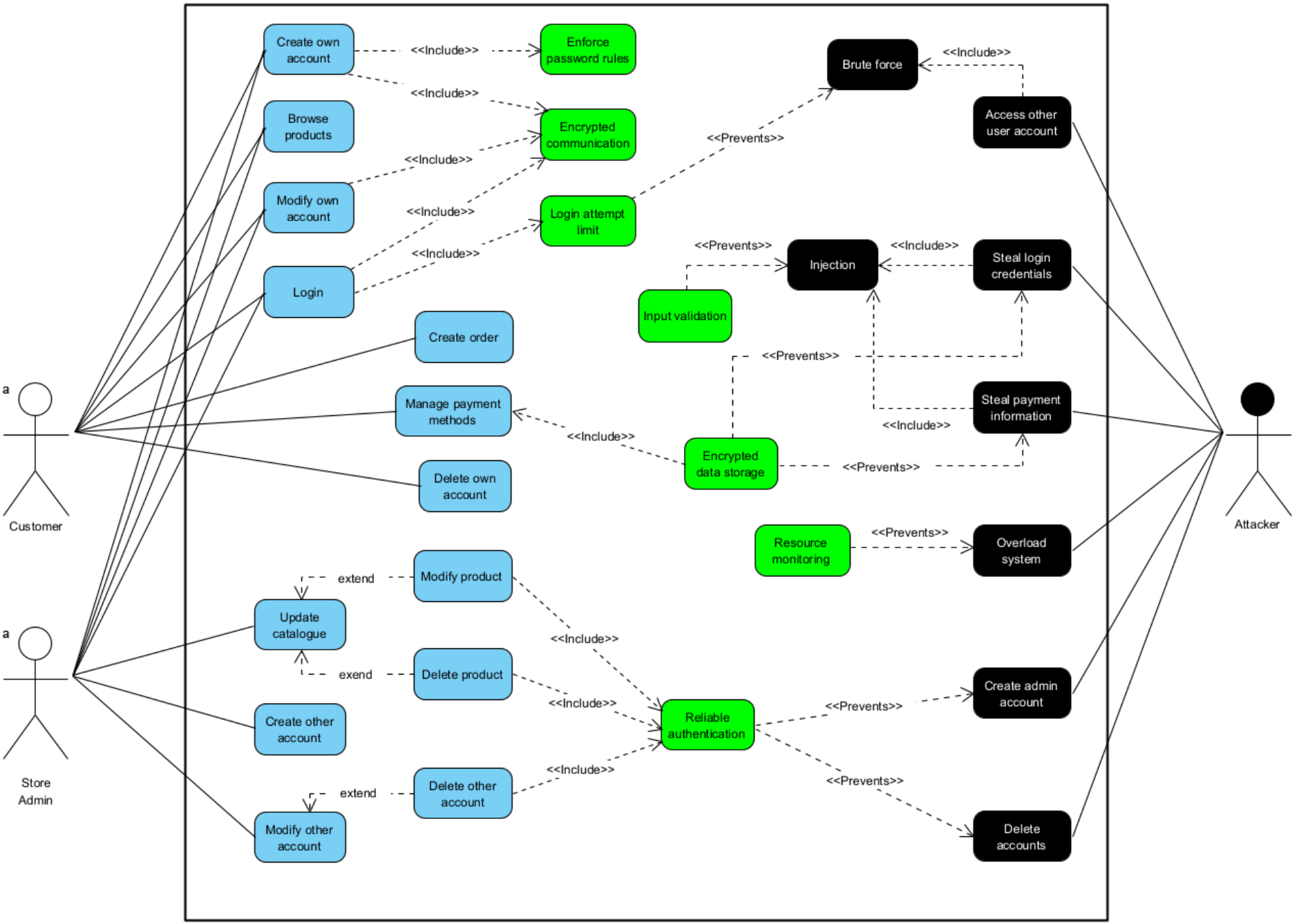

Figure 4 - System misuse case diagram

Design Patterns

The following design patterns are used for implementation of the system:

- Model-View-Controller (MVC)
    - The model-view-controller pattern is applied to manage the interaction between users and software components. Development of a robust controller ensures user input is sanitised and validated prior to being passed to functional services.

- Factory
    - The factory pattern is used to create users from within the controller, to prevent exposing the user model to the users.

- Singleton
    - The singleton pattern ensures only one Storage object exists, maintaining data control and reliability.

Data Modelling

Figure 5 shows the class diagram for the process of a customer placing an order.

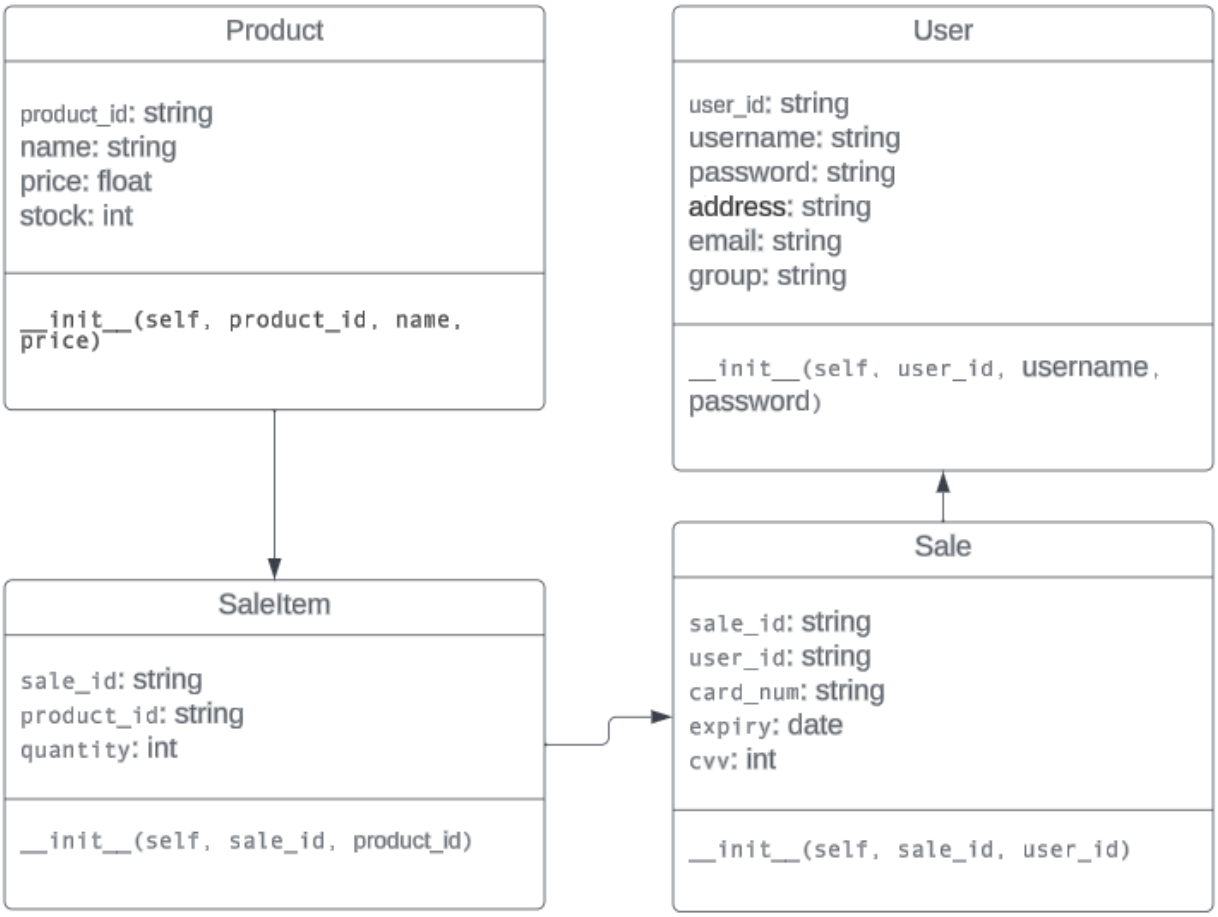

Figure 5 - Class diagram for online store

Data storage is achieved using the Storage class. The Storage class contains methods to add, get, update, and delete items for each entity type. Each entity type is stored in a separate dictionary within the Storage class (products, users, sales, sale_items), using unique identifiers (like product_id or user_id) as keys.

IStorage is used as an abstract interface to Storage. Using this interface means that in future iterations of the system, the method or design of Storage is able to change

without affecting the existing application. IStorage can also be easily mocked for testing, allowing unit tests to be performed without requiring the storage system to physically exist.

Storage implements the Singleton design pattern using a metaclass, SingletonMeta. In this pattern, the ConcreteStorage class can only have one instance throughout the application.

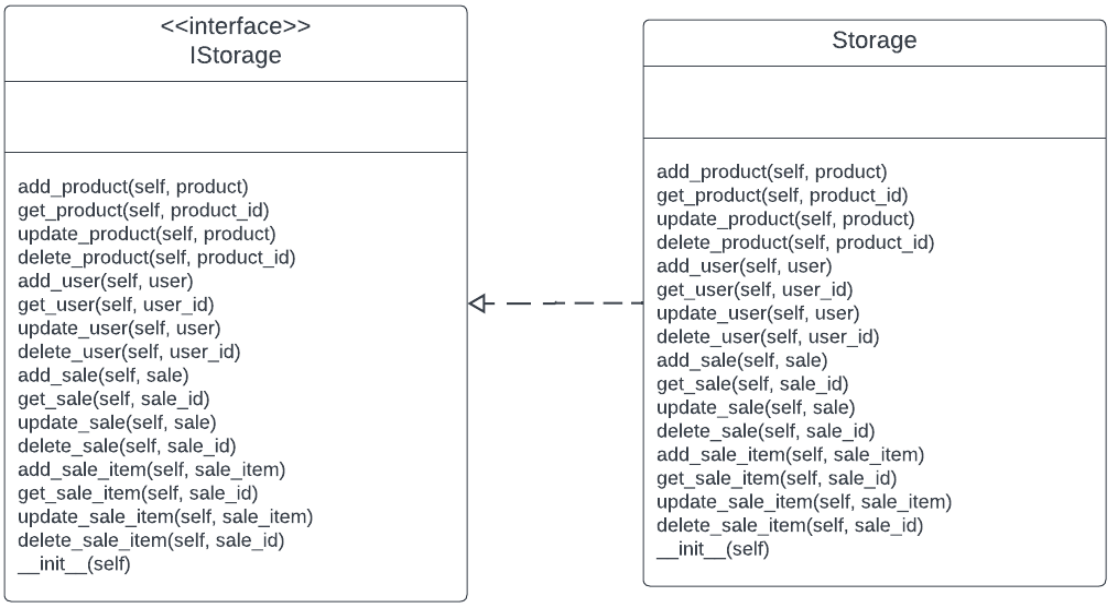

Figure 6 - Storage class and interface

Encryption will be performed using the AES algorithm, through use of the AES Cipher library. The encryption and decryption processes are outlined in Figure 7.

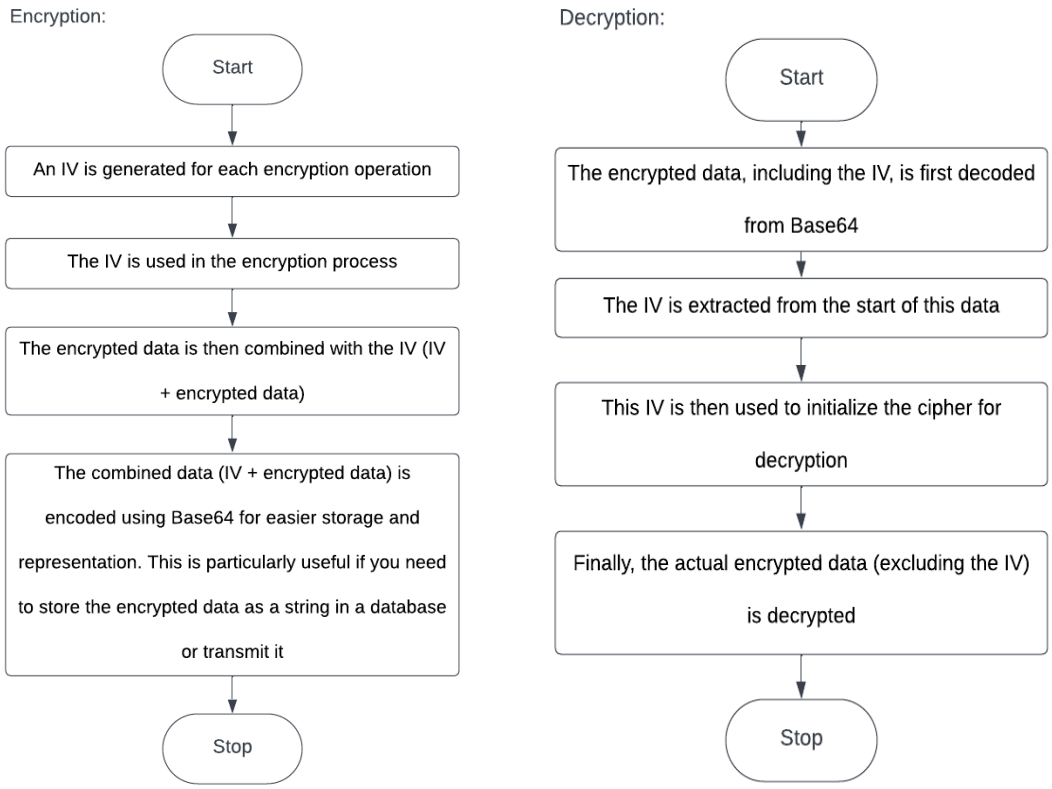

Figure 7 - AES encryption and decryption processes (note: IV = Initialisation vector)

Tools and Libraries

- Crypto (from pycryptodome)
    - pycryptodome is a self-contained Python package of low-level cryptographic primitives. It provides various cryptographic modules, including AES for symmetric key encryption.

- Base64
    - Base64 is used to encode binary data into ASCII characters and decode it back, which is used for easier storage and transmission.

- Hashlib
    - Hashlib supports various hash algorithms like SHA-1, SHA-256, SHA-512, and MD5.

- AES Cipher
    - Used for encrypting and decrypting data using the Advanced Encryption Standard (AES) algorithm.

Word count: 1049

References

European Union. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). Official Journal of the European Union: L119, 1-88.

Huang, L. (2022) *Full-stack E-commerce online store: A web application for purchasing and uploading products for every user.* Vaasa University of Applied Sciences.

OWASP (2021). OWASP Top Ten. Available at: https://owasp.org/top10 (Accessed: 12 December 2023).

Pauli, J. J.& Xu, D. (2005) Misuse case-based design and analysis of secure software architecture. *International Conference on Information Technology: Coding and Computing (ITCC'05)* 2: 398-403.

Pillai, A. B. (2017) Software architecture with Python: design and architect highly scalable, robust, clean, and high performance applications in Python. 1st ed. PACKT Publishing.

Rehim, R. (2016) Effective Python penetration testing : pen test your system like a pro and overcome vulnerabilities by leveraging Python scripts, libraries, and tools. 1st edition. Birmingham: Packt Publishing.

Tenzin, S., Lhamo, T. & Dorji, T. (2022) Design and Development of E-commerce: Web-application for Cooperative Store. *International Research Journal of Engineering and Technology (IRJET)* 9(2): 843-847.

William, G., Anthony, R. & Purnama, J. (2020) ‘Development of NodeJS based Backend System with Multiple Storefronts for Batik Online Store’, *International Conference on Engineering and Information Technology for Sustainable Industry (ICONETSI 2020).* Tangerang, Indonesia, 28-29 September. New York: ACM.