Secure Software Development

Assignment 1: Design Document

Team: Complete-Meditation

Authors:
Grzegorz Pikus
Elham Mortazavi Sarmad
Maja Tagt
Richard Allen
Wang Wang

Word count: 1097

1. Introduction

Our system has been designed for CERN- one of the biggest research organizations working on basic research in physical science. The organization is characterized by having some features such as periodic pressure on resources, interactive response requirements between request and reply and substantial data download requirements (Essex University student Platform, 2022).

Our system will allow users to access data/files in a remote repository. CERN users can use this system to share data/files with other users. The system should be designed to reduce the chance of data leaking as little as possible to reduce unwanted exposure of sensitive data., and only authorized users can access specific data.

2. GDPR considerations

To assess legal requirements for the technical requirements, the Concretisation of legal requirements (KORA) have had been used (Ringmann et al 2018). To comply with GDPR the following technical requirements have been identified:

- Authentication and authorisation for purpose limitation and prevent unauthorised access, Article 5(1)(f) (GDPR, 2018).
- Event monitoring to comply with incident and incident management, Article 32 (GDPR, 2018).
- Data encryption to secure data subjects' data, Article 28(3)(b) (GDPR, 2018).
- Risk analysis identified SQL injection as high risk, therefore additional technical requirement Article 32(2) (GDPR, 2018).
- Back up of database to comply with Article 32(1)(c) (GDPR, 2018).
- Ability to download data to comply with subjects' data requests Article 32(1)(c) (GDPR, 2018).

3. Design Considerations

3.1. Goals and Guidelines

Design an application which:

- is compatible with CERN system to make sure it works with their system
- can meet all privacy and security requirements.
- allows local or remote accessibility (Van der kleut, 2022).
- supports access via web browser/command line.

To fulfill the goals, there are some considerations:

- CERN System requirements:
  - Multi-core CPUs to speed up the processes.
  - Reliable internet connection for remote computer access.
  - Storage spaces for application programs, system software, configuration files and database (IBM, 2021).
- Constraints (Pillai, 2017):
  - scalability and availability requirements of the application.
  - organizational preference for adopting architectures like their current ones.
  - organizational preference for adopting approaches to manage cost and time

3.2 Technologies

- Python 3.10
- Flask with HTTPS enabled as a framework to build microservices.
- SQLite
- Pylint to use for code quality/Error detection/Duplicate code detection.
- Bandit will be used as a static analysis tool to detect security defects.
- Mccabe to evaluate code complexity.
- Hashlib
- MySQL.connector Module
- AEAD Encryption algorithm

4. Proposed system

The system will be developed for the CERN, allowing users to create, upload, download, and delete files/data from a remote repository. When a user accesses the system, it will base on the user's permission to generate a view and allow actions on files/data. It will have a rich permission configuration to prevent hindering staffs' daily work and gain high security to mitigate data leaks.

For the purpose of building a high-quality, easy extension in the future and long-term maintainable software, all the components in the system will be written in single responsibility and interact with interface modern OOP style. The system also is designed to apply the microservice style, and each component can interact with the message or REST-API style. When processing user requests, any request should have its security policy for validation.

4.1. Graphical Diagrams

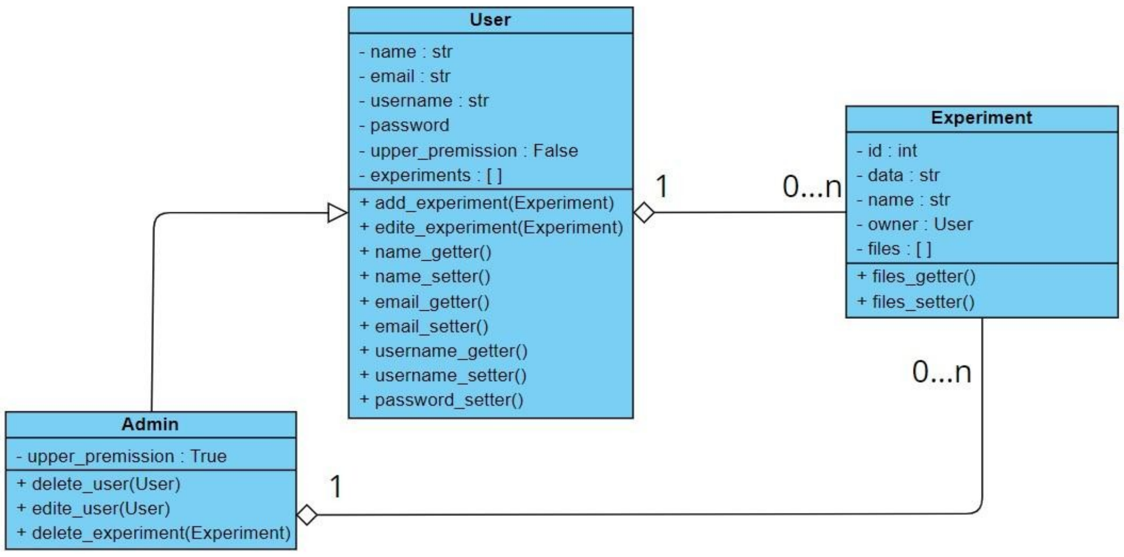


Chart 1. Class Diagram

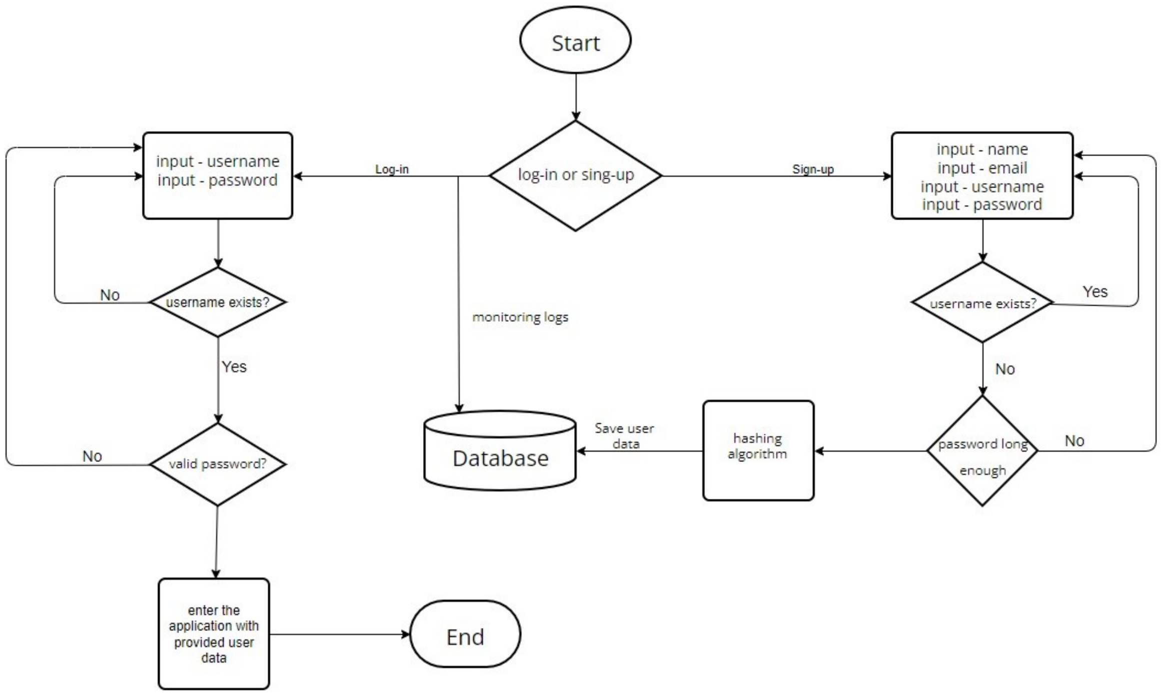


Chart 2. Flowchart of authentication (log-in and sign-up).

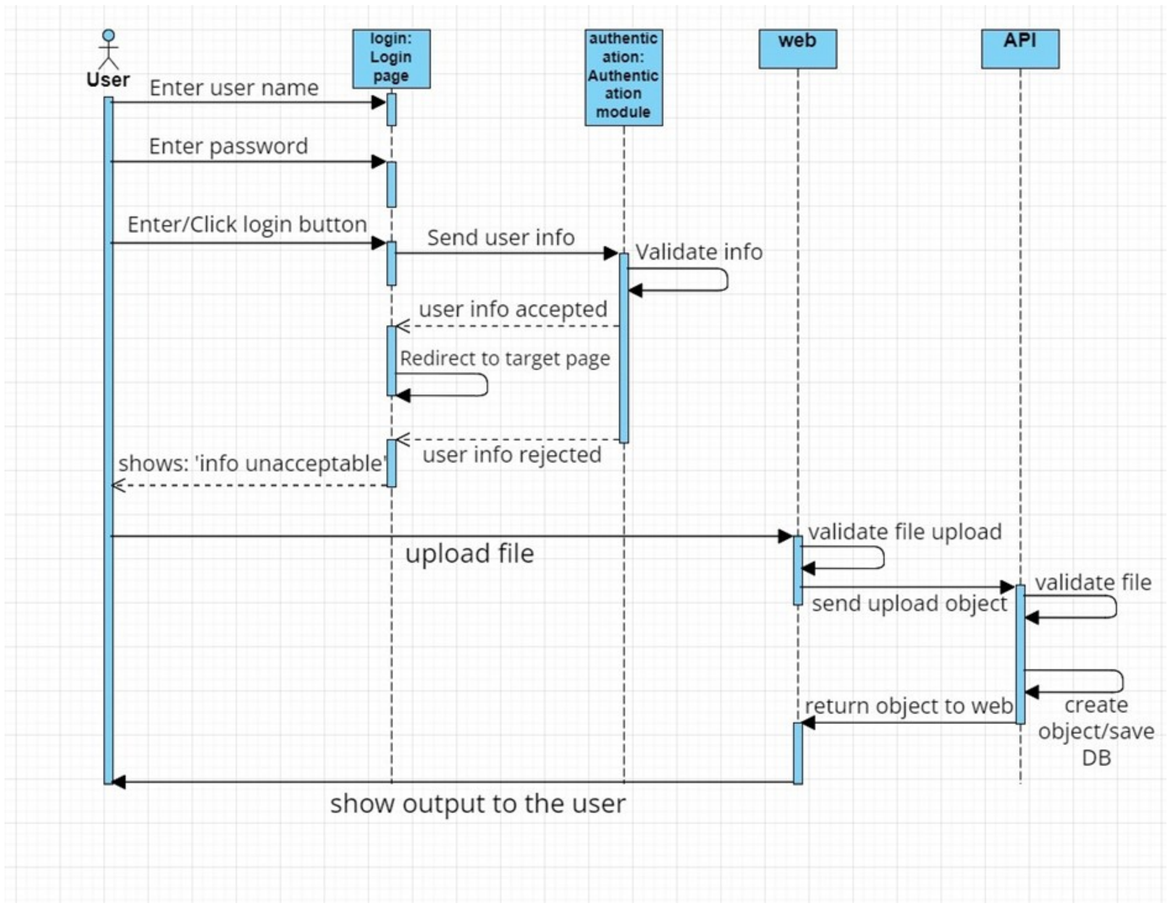


Chart 3. Sequence Diagram

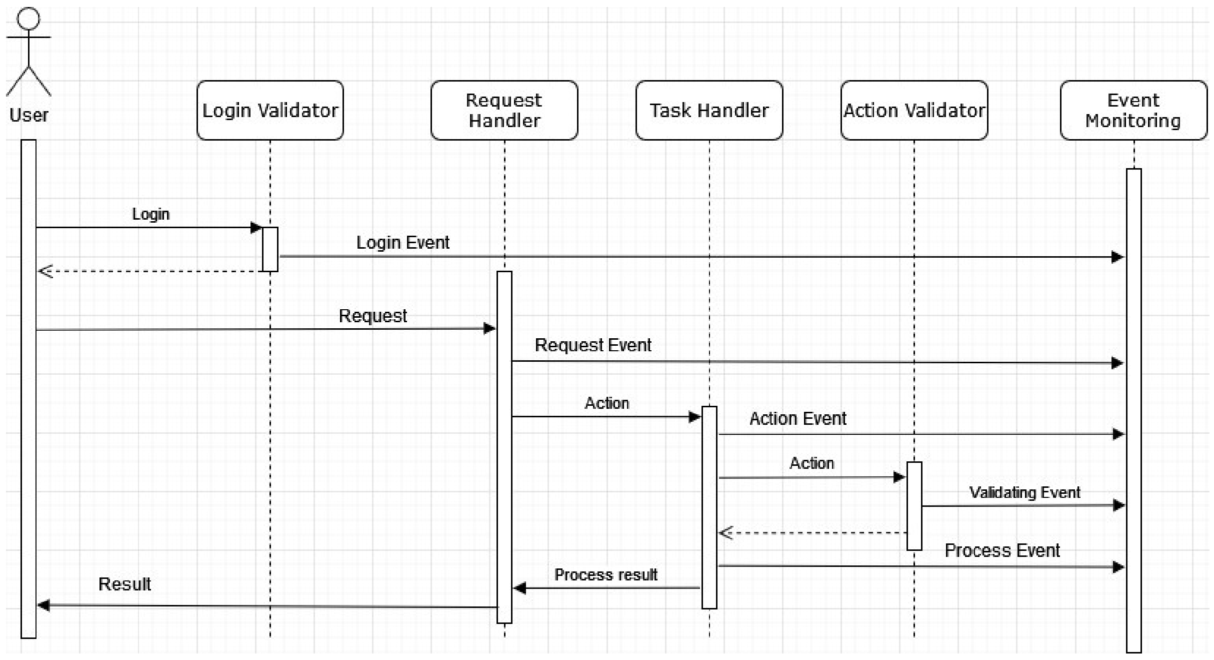


Chart 4. User Request Processing Sequence Diagram

5. System Attack Surfaces

To design and develop a secure and reliable system, some common vulnerabilities should be avoided or the system should be equipped with some mechanism as follows:

5.1 Injection

As a CERN database, the data stored may be available in the public domain. However, an attacker could gain admin account access details, which brings a threat to other databases connected in the system as scalability is a consideration for our project, and there is the threat of data deletion or tampering (Ballmann, 2021).

Considering user input is the main vector for SQL injection attacks, the database needs to distinguish between legitimate requests and to satanize potentially harmful ones. In this project we are using Python parameterised queries to perform MySQL database operations, it is a defensive option recommended by OWASP (OWASP, 2021)

5.2 Authentication

In our system design we are going to implement log-in and sign-up functions to get access to the application. A user can sign-up to the application by providing: name, e-mail address, username and finally password. The user can have access to the application by logging-in by providing her/his username and password.

We are going to implement hashing function to securely save users password by SHA-256 algorithm. It will transform each user's password with a random length into a sequence 256 bits and all secured hashed passwords will be stored in the database. This algorithm is deterministic and has avalanche effect, even a small change results in diametrical change in the output. Moreover, to make the application more secure, application logs are necessary. This would be simply a file that contains all the events regarding logging-in or attempts of logging-in (Kent 2006). We are going to implement a proper solution to save logging events in the database. Based on OWASP advice, sufficient logging and monitoring can prevent many attacks by proper log files and security sensitive data analyzing (OWASP, 2018).

5.3 Authorisation

The system will have two roles, user and administrator, each with different system access and authorisation (See Appendix A).

The whole system focuses on reducing data leaking and unauthorized access, but high security does not mean it will bring obstacles to regular users' daily use. The system will provide various configurations to each user request to satisfy different scenarios. Every file on the server will have a classification, and only users who meet the permission requirement can perform corresponding actions. For example, the system can provide config to a user with a specific IP address or range to operate a file; or only a particular user can download one specification file.

5.4 Event monitoring

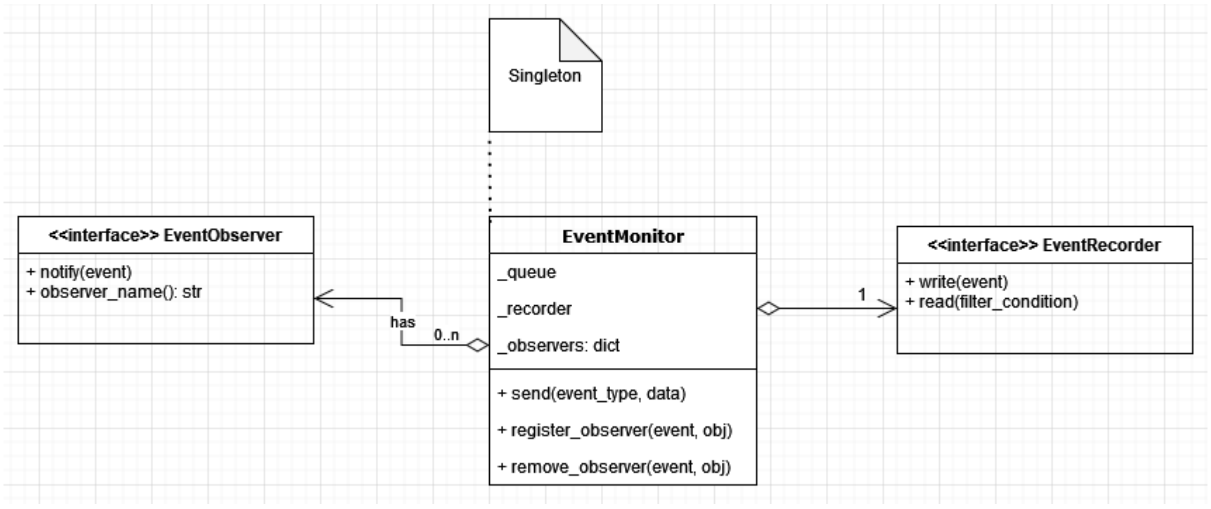


Chart 3. Event monitoring

There will be an appropriate component to record all the events in the system, and system maintainers can audit data to determine any suspect action.

5.5 Data encryption

To comply the GDPR principles of confidentiality and integrity, it is recommended to apply OWASP top ten threat level on a system application connected to the web (Vagenas, 2019). In relation to encryption of data, OWASP recommends data falling under the special protection of the GDPR to be encrypted in both transit and in rest (OWASP, 2022).

Always Encrypted (AE) is a solution by Microsoft that stores keys separate from the database, and only decrypts data when it reaches the client, which holds the decryption key (Microsoft, 2022).

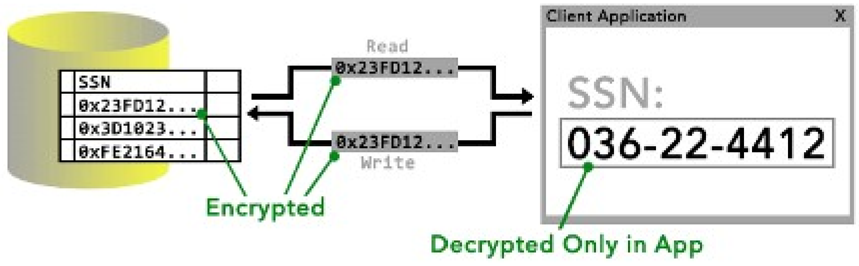


Figure 3. Data encryption in transit (Bertrand 2015).

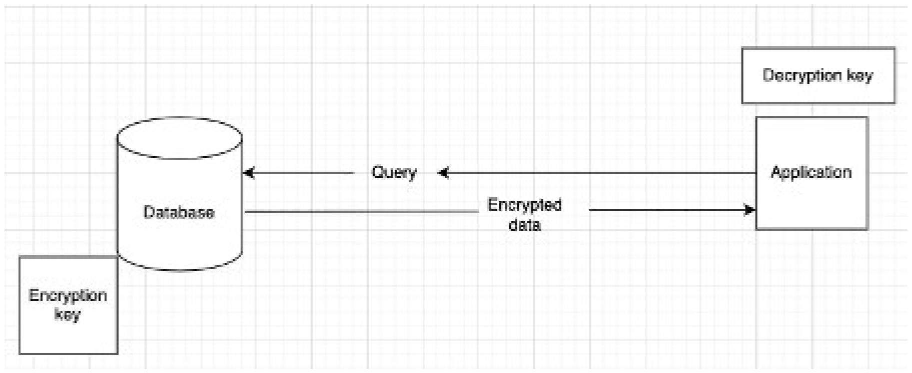
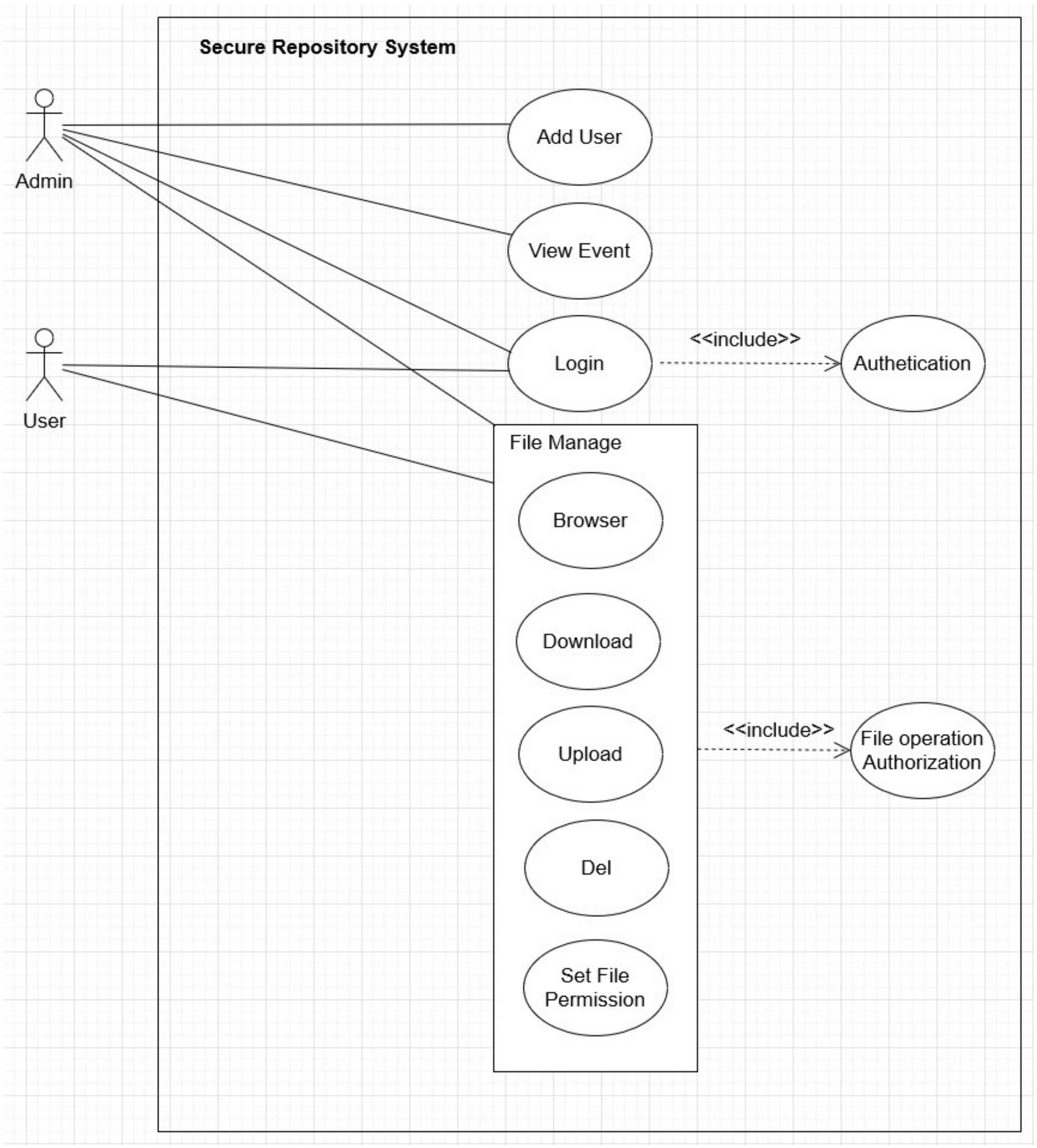


Chart 6. Flowchart detailing encryption

Appendix



Appendix A Use Case Diagram

References

Domenique Ringmann, S., Langweg, H. and Waldvogel, M. (2018). Requirements for Legally Compliant Software Based on the GDPR. [online] OTM 2018 Conferences. Switzerland: Springer Nature AG 2018, pp.258–276. Available at: https://link.springer.com/chapter/10.1007/978-3-030-02671-4_15#ref-CR10 [Accessed 23 Jul. 2022].

Hassan Kilavo, Salehe I. Mrutu & Robert G. Dudu (2021). Securing Relational Databases against Security Vulnerabilities: A Case of Microsoft SQL Server and PostgreSQL, Journal of Applied Security Research, DOI: 10.1080/19361610.2021.2006032

Sheldon, R. (2017). *SQL Server Encryption: Always Encrypted*. [online] Available at: https://www.red-gate.com/simple-talk/databases/sql-server/database-administration-sql-server/sql-server-encryption-always-encrypted/ [Accessed 19 Jul. 2022].

Cryptographic Storage Cheat Sheet (2022). *Cryptographic Storage - OWASP Cheat Sheet Series*. [online] Available at: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html#cipher-modes [Accessed 19 Jul. 2022].

McGrew, D., Foley, J., Paterson, K. (2014). Authenticated Encryption with AES-CBC and HMAC-SHA draft-mcgrew-aead-aes-cbc-hmac-sha2-05.txt. [online] Available at: https://datatracker.ietf.org/doc/html/draft-mcgrew-aead-aes-cbc-hmac-sha2-05 [Accessed 19 Jul. 2022].

Syed, A. (2018). *Is SQL Server Always Encrypted, for sensitive data encryption, right for your environment*. [online] SQL Shack - articles about database auditing, server performance, data recovery, and more. Available at: https://www.sqlshack.com/is-sql-server-always-encrypted-for-sensitive-data-encryption-right-for-your-environment/ [Accessed 19 Jul. 2022].

Bertrand, A. (2019). *SQL Server 2016 Always Encrypted*. [online] Available at: https://www.mssqltips.com/sqlservertip/4011/sql-server-2016-always-encrypted/ [Accessed 19 Jul. 2022].

Ballmann, B. (2021) *Understanding network hacks : attack and defense with Python 3*. Second edition. [Online]. Berlin, Germany: Springer.

PYnative, (2021). *Python MySQL Execute Parametrized Query using Prepared Statements.* [online] Available at: https://pynative.com/python-mysql-execute-parameterized-query-using-prepared-statement/#h-use-of-parameterized-query-and-prepared-statement [accessed on 21 July 2022]

OWASP, (2021). *SQL Injection Prevention Cheat Sheet.* [online] Available at: https://cheatsheetseries.owasp.org/cheatsheets/Query_Parameterization_Cheat_Sheet.html [accessed 21 July 2022]

Kent, K., Souppaya, M., (2006) [Online] "Guide to Computer Security Log Management, Recommendations of the National Institute of Standards and Technology" available at: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf [Accessed on 18. 07. 2022].

OWASP (2018) [Online] "C9: Implement Security Logging and Monitoring, available", at:https://owasp.org/www-project-proactive-controls/v3/en/c9-security-logging [Accessed on 18. 07. 2022].

Secure Software Development. (2022).University of Essex online Student platform. https://www.my-course.co.uk/mod/assign/view.php?id=660949 Available from: [Accessed 23 July]

Pillai, A. B. (2017). Software architecture with Python. 1st ed. Packt Publishing.

Van der kleut, J. (2022). Remote computer access: what is it and what are the risks? Available from: https://us.norton.com/internetsecurity-how-to-remote-computer-access.html# [Accessed 18 July 2022]

N.D. (2021). Storage requirements overview. Available from: https://www.ibm.com/docs/en/cmofz/10.1.0?topic=zos-storage-requirements-overview [Accessed 18 July 2022]