Prepared By Group 1

Fiqki Azizah/ MD Chowdhury/Pok Chi Li/Sultan Alaryani/Mohsen Almazroei

SECURE COPYRIGHT MANAGEMENT APPLICATION DESIGN PROPOSAL

Contents

[1. Introduction 2](#_Toc190557540)

[2. System Design Overview 2](#_Toc190557541)

[2.1 System Features & Security 2](#_Toc190557542)

[3. UML Diagrams 2](#_Toc190557543)

[3.1 Use Case Diagram 2](#_Toc190557544)

[3.2 Sequence Diagram 3](#_Toc190557545)

[4. Database Design & Data Structures 5](#_Toc190557546)

[4.1 Database Schema and Application Design 5](#_Toc190557547)

[4.2 Data Flow Design 6](#_Toc190557548)

[5. Design Patterns & Paradigms 6](#_Toc190557549)

[5.1 Object-Oriented Principles 6](#_Toc190557550)

[5.2 Design Patterns 6](#_Toc190557551)

[6. Conclusion 7](#_Toc190557552)

# 1. Introduction

The rise in copyright disputes necessitates a secure solution for intellectual property protection. Secure Copyright Management Application (SCMA) provides an encrypted repository for lyrics, music scores, and audio recordings, ensuring integrity, security, and controlled access (Khan et al., 2022).

# 2. System Design Overview

## 2.1 System Features & Security

SCMA integrates multiple security measures for intellectual property protection:

* Secure Storage: AES-256 encrypted storage for digital content (Tutorialspoint, 2022).
* Integrity Verification: SHA-256 hashing ensures artefact integrity (Buelta, 2022).
* Role-Based Access Control (RBAC):

1. Admins: Manage artefacts, users, and permissions.
2. Users: Upload, view, and modify their artefacts.

* Timestamping:Logs creation and modification times (Object Management Group, 2020).
* Encryption: AES-256 encryption protects stored data (Khan et al., 2022).
* Audit Logs: Tracks user interactions for compliance.

These features ensure secure and controlled access to copyrighted content.

# 3. UML Diagrams

## 3.1 Use Case Diagram

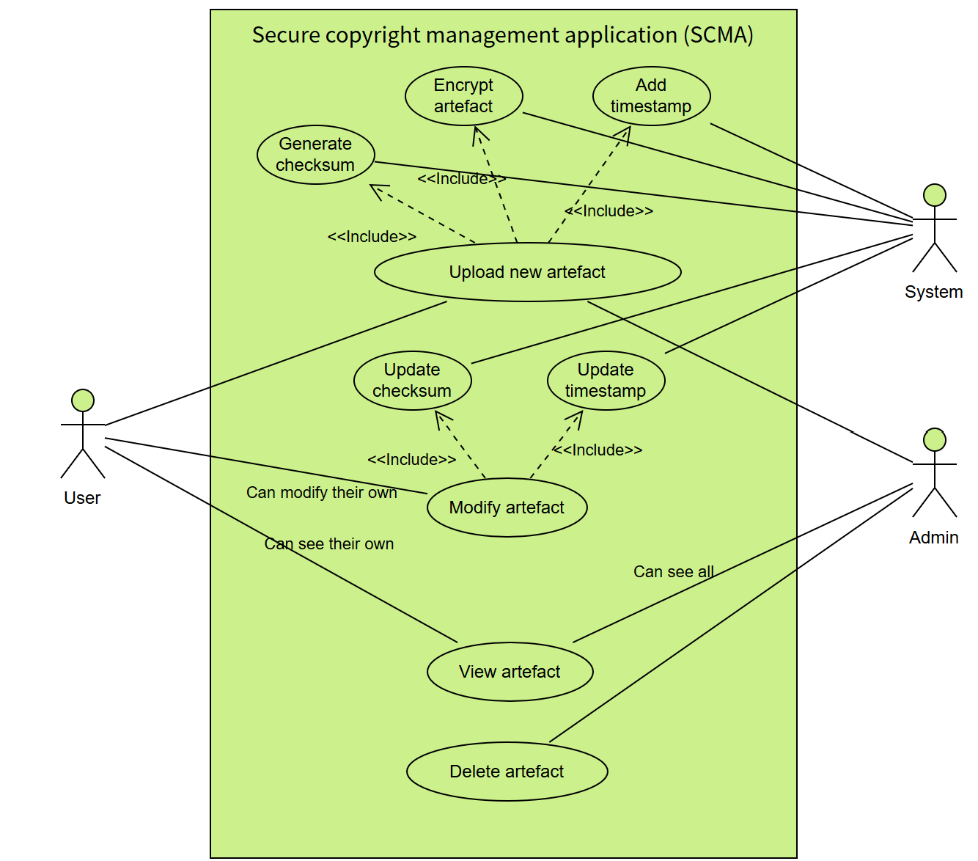


Figure 1: Use case diagram for SCMA

The Use Case Diagram illustrates key functionalities such as artefact upload, viewing, modification, and deletion, while enforcing security measures like authentication, encryption, and access control.

## 3.2 Sequence Diagram

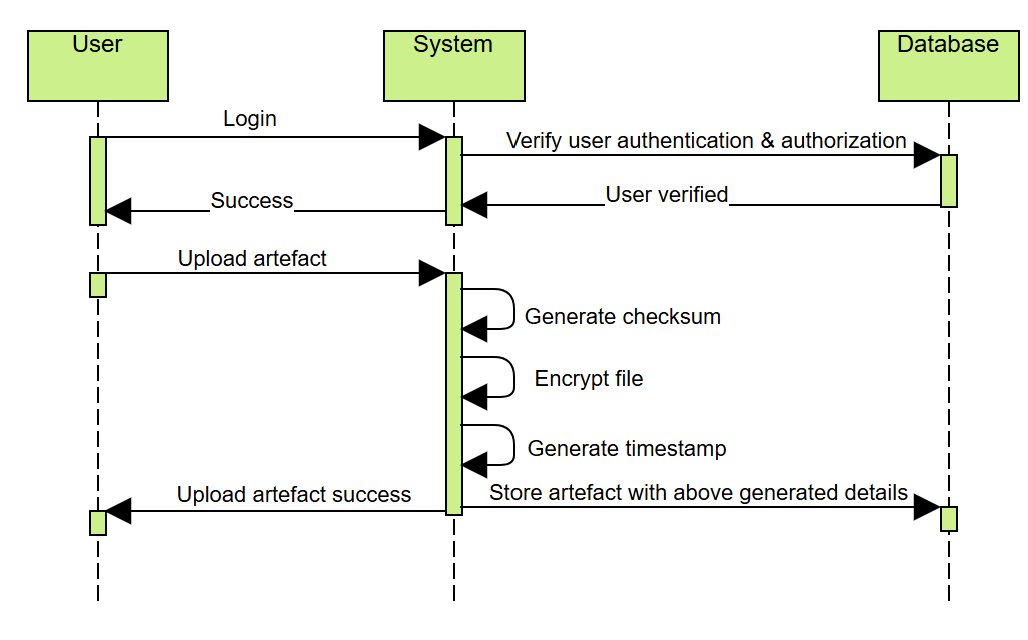


Figure 2: Sequence diagram for upload artefact

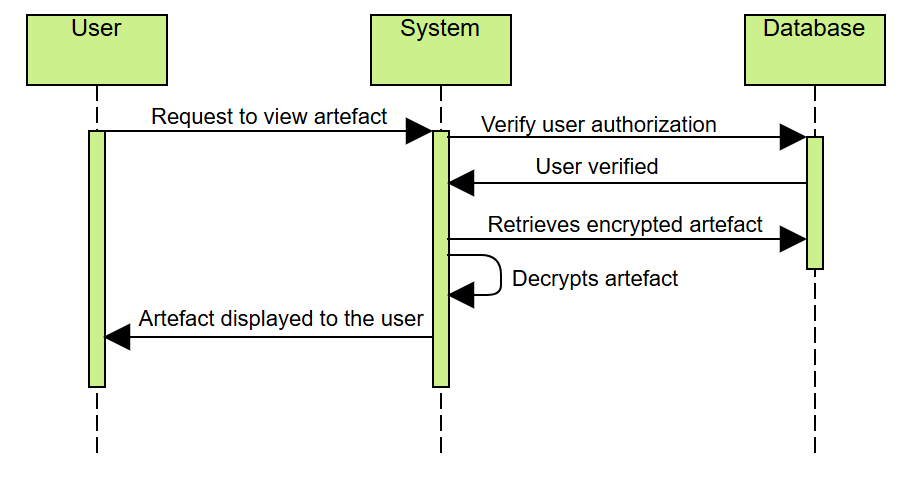


Figure 3: Sequence diagram for view artefact

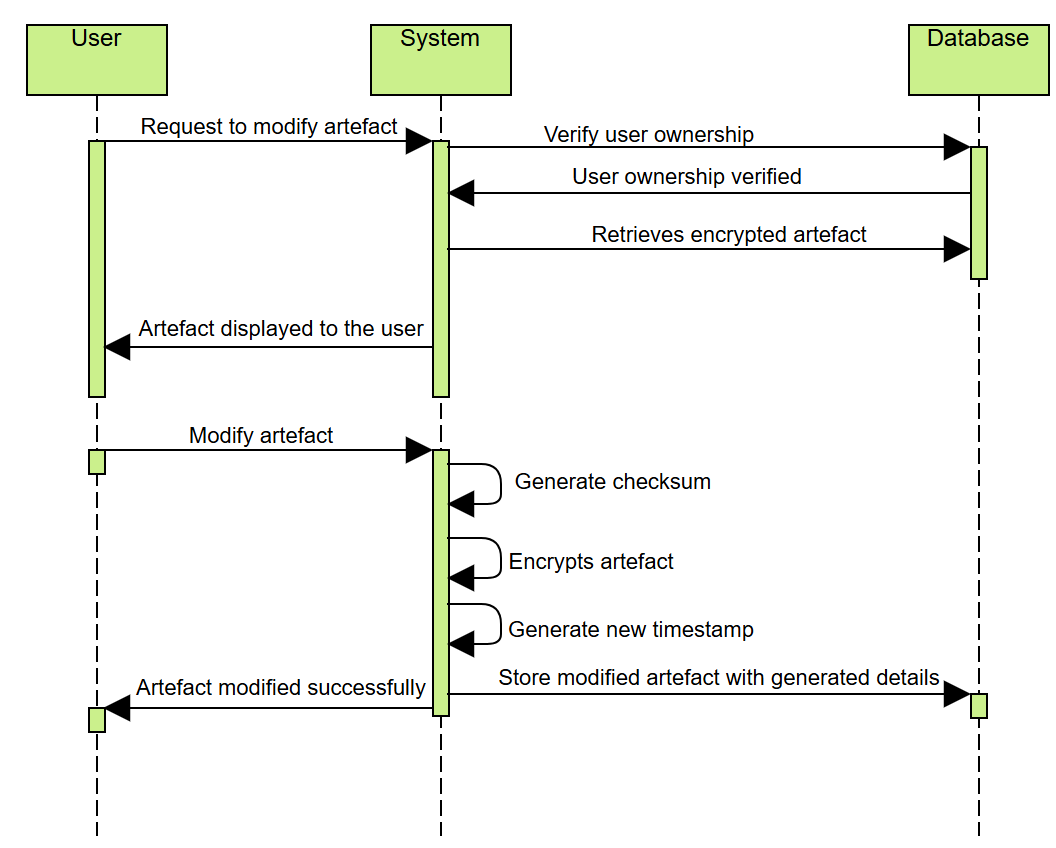


Figure 4: Sequence diagram for modify artefact

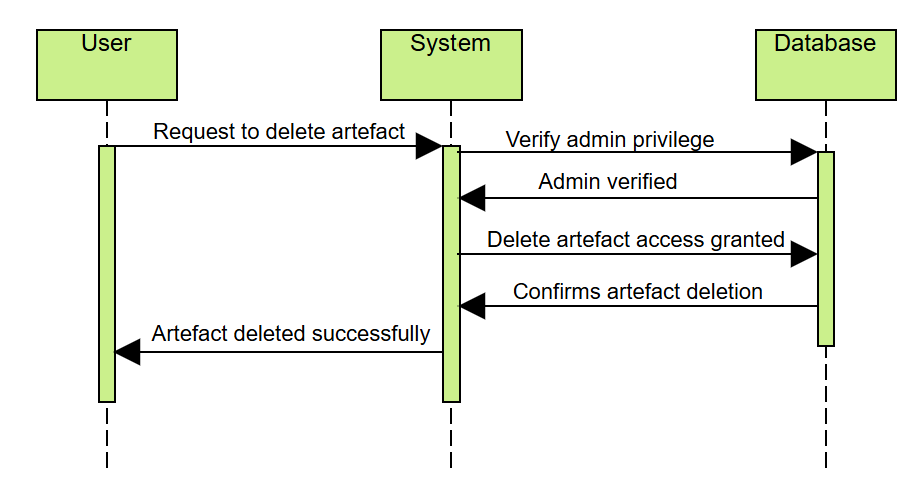


Figure 5: Sequence diagram for delete artefact

As shown in the sequence diagram, the system ensures secure artefact management by verifying access, encrypting data, maintaining integrity with checksums and timestamps, and confirming each action.

# 4. Database Design & Data Structures

## 4.1 **Database Schema and Application Design**

Figure 6: Class Diagram for application design

In Figure 6 above, the table named User, AccesLog, Artifact, Lyrics, MusicScore and AudioRecording are data table in the system. The system allows users to conduct different actions based on the user’s privilege (admin rights, viewer rights etc).

## 4.2 Data Flow Design

A diagram of a company

Description automatically generated

Figure 7: Data Flow Diagram

# 5. Design Patterns & Paradigms

## 5.1 Object-Oriented Principles

SCMA applies object-oriented principles to ensure modularity, scalability, and maintainability. Encapsulation restricts direct access to sensitive data, such as encrypted files, by defining access control methods. Inheritance enhances reusability, as entities like Lyrics, MusicScore, and AudioRecording inherit from a common Artifact superclass. Abstraction focuses on the essential aspects of the system (Breuer, 2023), allowing developers to work with high-level entities like Artifacts. While polymorphism allows a single interface to represent different forms (Vats, 2025), in this case multiple file types, enabling the system to handle Lyrics, MusicScores, and AudioRecordings uniformly.

## 5.2 Design Patterns

SCMA incorporates key design patterns to enhance system efficiency and security. The Factory Pattern is used to instantiate different Artifact objects based on user inputs. The Singleton Pattern ensures a single instance of core components, such as the encryption manager, to avoid redundancy. The Observer Pattern is applied in access logging, where modifications trigger automatic logging. Additionally, the Strategy Pattern allows flexible encryption method. These patterns enhance maintainability and adaptability while adhering to secure software engineering best practices.

# 6. Conclusion

The SCMA system is designed with robust security measures, ensuring safe storage, controlled access, and data integrity for copyrighted content. By leveraging object-oriented principles and well-established design patterns, SCMA provides a scalable, maintainable, and secure framework for intellectual property protection. The implementation of encryption techniques, role-based access control, and automated logging ensures compliance with security best practices.

# References

Breuer, M. (2023) ‘The Principle of Abstraction in Software Development: Simplifying Complexity’, Software Architecture, 25 July. Available at: https://www.bytefusion.de/2023/07/25/the-principle-of-abstraction-in-software-development-simplifying-complexity/ (Accessed: 16 February 2025).

Buelta, J. (2022) Python Architecture Patterns: Master API Design, Event-Driven Structures, and Package Management in Python. Birmingham: Packt Publishing Ltd. Available at: https://learning.oreilly.com/library/view/python-architecture-patterns/9781801819992/Text/Part\_2.xhtml (Accessed: 2 February 2025).

Khan, R. A., Khan, S. U., Khan, H. U., & Ilyas, M. (2022) 'Systematic Literature Review on Security Risks and its Practices in Secure Software Development', IEEE Access.

Object Management Group (2020) Unified Architecture Framework (UAF) Domain Metamodel. Version 1.1. Available at: https://www.omg.org/spec/UAF/1.1 (Accessed: 2 February 2025).

Tutorialspoint (2022) Cryptography with Python - Quick Guide. Available at: https://www.tutorialspoint.com/cryptography\_with\_python/cryptography\_with\_python\_quick\_guide.htm (Accessed: 2 February 2025).

Vats, R. (2025) ‘Polymorphism in OOP: What is it, its types, examples, benefits, & more’, upGrad Blog, 13 February. Available at: https://www.upgrad.com/blog/polymorphism-in-oops/ (Accessed: 16 February 2025).