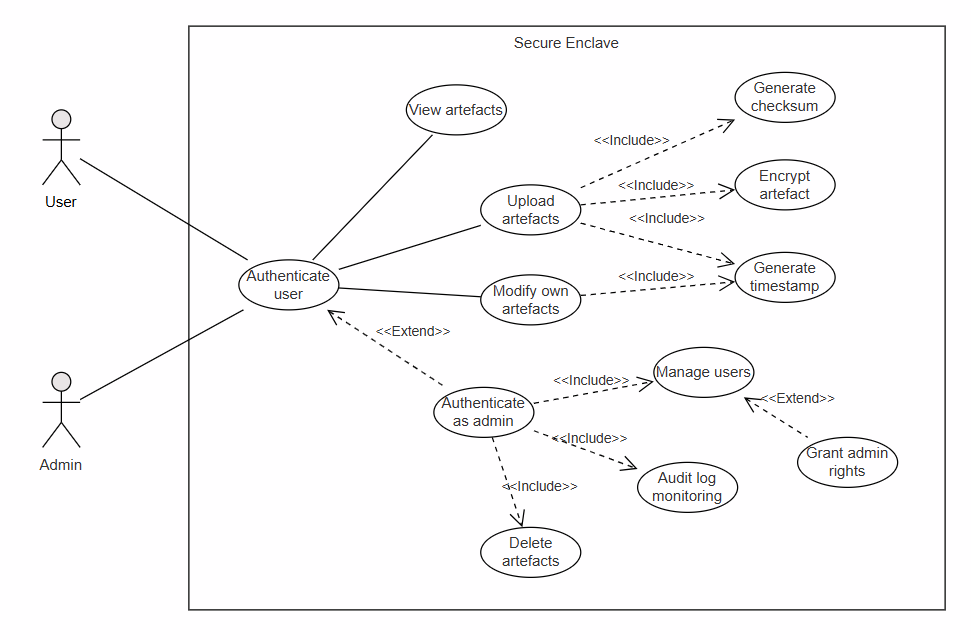
# DEVELOPMENT TEAM PROJECT: DESIGN DOCUMENT

## Introduction

The digital transformation of the music industry raises concerns about copyright infringement and unauthorised replication (Kumar & Mahajan, 2024). To address this, we propose a secure digital application that ensures Confidentiality, Integrity, and Availability (CIA) of copyrighted materials (Zhang, 2024) through encryption, access control, and audit logging. This document presents the system’s design using Unified Modelling Language (UML) diagrams, illustrating architecture, data flow, and user interactions.

## Use Case Diagram

Song et al. (2024) state that a UML use case diagram can be used to “express application requirements in an intuitive and standardised way”. Figure One demonstrates the use cases for the proposed application. The key actors are the User and Administrator, with distinct privileges being granted to the Administrator. By giving minimal rights, this lowers the attack surface for a potential malicious user or insider and complies with the “principle of least privilege” (Billoir et al., 2024).

Figure One: *UML use case diagram*

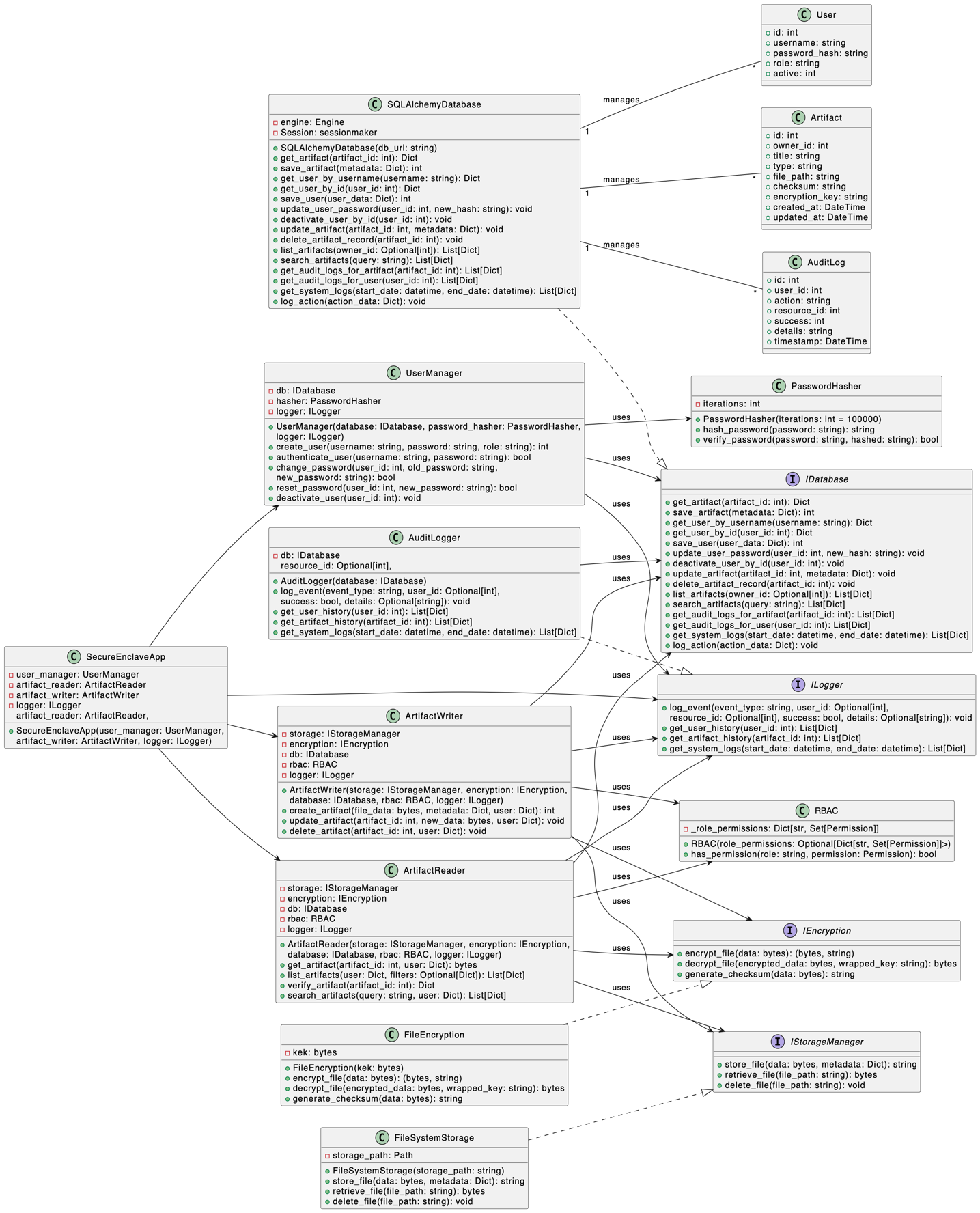
## Class Diagram

Class diagrams capture a system’s classes, interfaces, and their relationships (Booch et al., 1999). Our design emphasises modularity, extensibility, and security, following object-oriented best practices (Larman, 2004). By defining interfaces for core services, we uphold the Interface Segregation Principle (Martin, 2008) and the Open-Closed Principle (Gamma et al., 1995). This approach decouples components, allowing new implementations (e.g., cloud storage) without modifying existing classes.

Concrete classes such as *FileSystemStorage* and *FileEncryption* demonstrate the Strategy Pattern (Freeman & Robson, 2004), supporting interchangeable implementations for these services. Additionally, classes like *ArtefactReader* and *ArtefactWriter* employ constructor injection (Fowler, 2005; Seemann, 2012), enabling tests to substitute or mock dependencies. We preserve encapsulation through private attributes (e.g., \_role\_permissions in *RBAC*) and controlled data access (Booch et al., 2007), safeguarding internal state and data integrity.

By implementing the *Database* interface, *SQLDatabase* provides an abstraction layer that supports polymorphism and code reuse across various database backends. Meanwhile, the domain model remains lean by isolating business logic in dedicated classes such as *UserManager*. This separation adheres to the Single Responsibility Principle (Martin, 2008), allowing each class to focus on a specific concern and enhancing overall clarity and maintainability.

By isolating components behind interfaces, the system remains adaptable to changing requirements while retaining robust security and role-based access control (Sandhu et al., 1996; Stallings & Brown, 2024). These design choices collectively meet the copyright protection requirements outlined in the project brief.

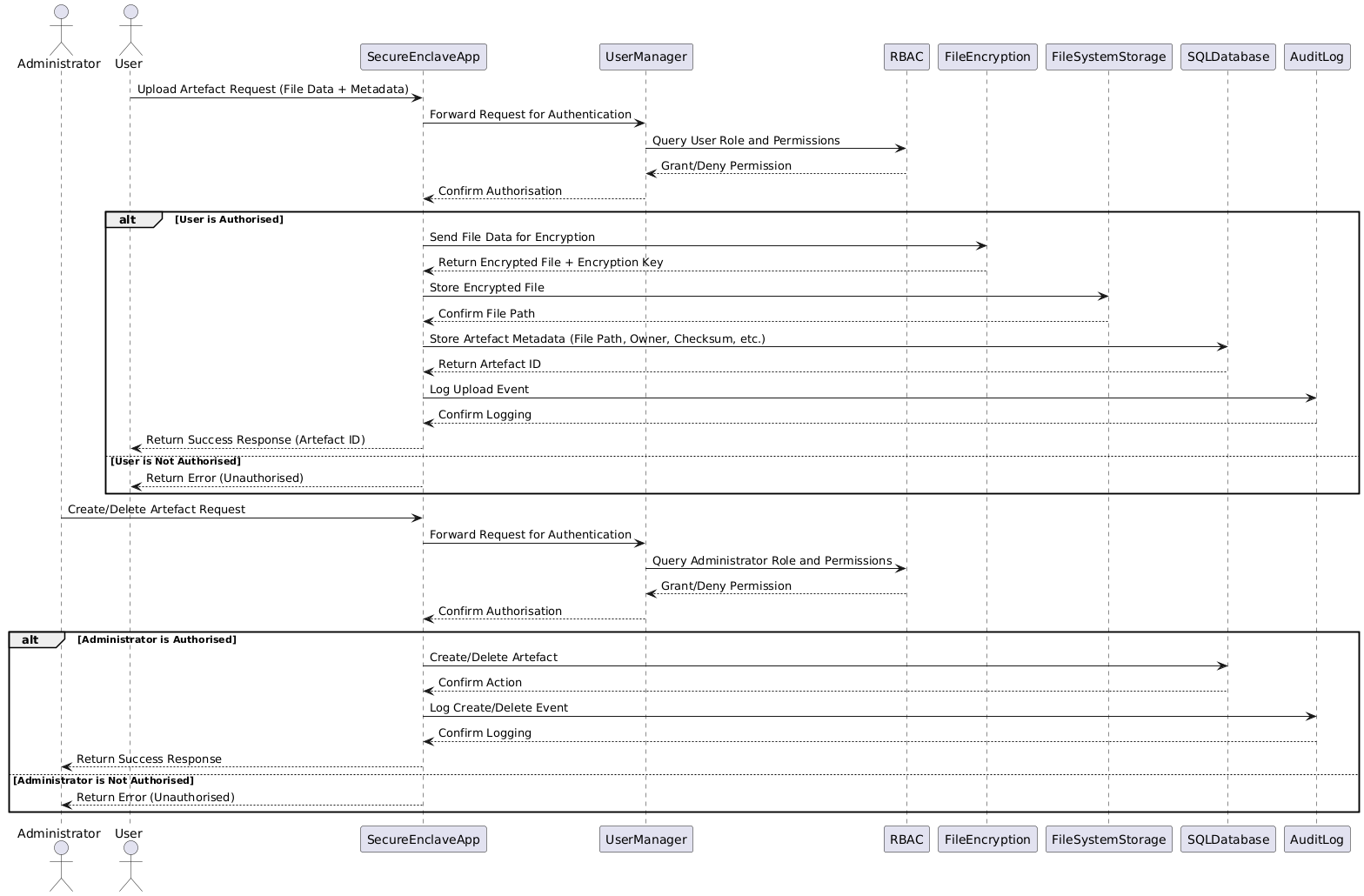
Figure Two: *UML class diagram*

## Sequence Diagram

Sequence diagrams model system interactions, offering clear visualisations for behaviour analysis and early issue detection (Baniassad & Murphy, 2006). It serves as a critical component of the system design, offering a structured representation of the interactions between actors and system components" (IEEE Systems Journal, 2025).

Although their static nature and complexity in large systems can reduce their effectiveness (Booch et al., 2007), it is recommended to accompany sequence diagrams with activity diagrams and iterative refinement. Despite these limitations, they remain necessary for secure designs (Fernandez et al., 2021; Khan et al., 2022).

The sequence diagram below shows the interaction between objects during key processes, like adding an artefact and verifying its integrity.

Figure Three: *Secure Artefact Upload Sequence Diagram*

**Process**

1. **Actors**

* User
* Administrator

1. **User Upload:**

* User sends an upload request to SecureEnclaveApp.
* UserManager authenticates via RBAC.
* If authorised, FileEncryption encrypts the file, stored in FileSystemStorage.
* Metadata is saved in SQLDatabase, and the event is logged in AuditLog.
* User receives a successful response with an Artefact ID.

1. **Administrator Actions**

* Similar authentication and authorisation steps apply.
* Artefacts are created/deleted in SQLDatabase, logged in AuditLog, and a success response is returned.

1. **Security Considerations**

* **Authentication & Authorisation:** Managed by UserManager and RBAC.
* **Confidentiality:** Files are encrypted using AES-256.
* **Integrity:** Checksums ensure file consistency.
* **Audit & Compliance:** Logs maintained by AuditLog.

## Conclusion

This project proposes a secure application for copyright management in the music industry using encryption, checksums, and timestamping. It follows object-oriented design and includes strong security measures to protect intellectual property. Implementing this solution can reduce copyright infringement risks and protect creators' rights.

## References

Baniassad, E. and Murphy, G.C. (2006) 'Interactive Exploration of UML Sequence Diagrams', IEEE Conference Publication, pp. 272–276. Available at: https://ieeexplore.ieee.org/document/1684295 (Accessed: 11 February 2025).

Billoir, E. et al. (2024) 'Implementing the principle of least administrative privilege on operating systems: challenges and perspectives', Annales des télécommunications [Online], 79(11–12), pp. 857–880. DOI: https://doi.org/10.1007/s12243-024-01033-5.

Booch, G., Rumbaugh, J. and Jacobson, I. (1999) The Unified Modelling Language User Guide. 2nd edn. Addison-Wesley.

Booch, G., et al. (2007) Object-Oriented Analysis and Design with Applications. Addison-Wesley.

Brown, A. (2021) 'Role-Based Access Control in Modern Systems', Journal of Cybersecurity, 12(3), pp. 45-60.

Fernandez, E.B., et al. (2021) 'Security Patterns in Practice', IEEE Software, 38(4), pp. 23-30.

Fowler, M. (2005) Inversion of Control Containers and the Dependency Injection pattern. Available at: https://www.martinfowler.com/articles/injection.html (Accessed: 14 February 2025).

Freeman, E. and Robson, E. (2004) Head First Design Patterns. O'Reilly Media.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

IEEE Systems Journal (2025) 'Research on the architecture design and key technologies'. IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10295824/ (Accessed: 15 February 2025).

Khan, S., et al. (2022) 'Advanced Encryption Standards for Secure Data Storage', Journal of Information Security, 14(1), pp. 78-92.

Kumar, T. and Mahajan, R. (2024) 'The Digital Sonata: Digitalisation's Impact on India's Music Industry and Artist Revenues', Digital Society. Springer. Available at: https://link.springer.com/article/10.1007/s44206-024-00147-6 (Accessed: 15 February 2025).

Larman, C. (2004) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process. 3rd edn. Pearson Education.

Martin, R.C. (2008) Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

Sandhu, R., Coyne, E., Feinstein, H. and Youman, C. (1996) 'Role-Based Access Control Models', IEEE Computer, 29(2), pp. 38–47.

Seemann, M. (2012) Dependency Injection in .NET. Manning Publications.

Song, S. et al. (2024) 'A deep learning-based approach to similarity calculation for UML use case models', Expert Systems with Applications [Online], 251, 123927. DOI: https://doi.org/10.1016/j.eswa.2024.123927.

Stallings, W. and Brown, L. (2024) Computer Security: Principles and Practice. 5th edn. Pearson.

Zhang, W.F. (2024) 'Digital music resource copyright management mechanism based on blockchain', 2024 IEEE/ACIS 27th International Conference on Computer and Information Science. IEEE. Available at: https://ieeexplore.ieee.org/abstract/document/10673923 (Accessed: 15 February 2025).