|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| Individual Reflective Document |
| |  |  |  | | --- | --- | --- | | Milad Chowdhury | 3/3/25 | Secure Software Development | |

Contents

[Introduction 2](#_Toc191985516)

[Description: What Happened? 2](#_Toc191985517)

[Feelings: How Did I Feel? 2](#_Toc191985518)

[Evaluation: What Went Well and What Didn’t? 2](#_Toc191985519)

[What Could Be Improved? 2](#_Toc191985520)

[Analysis: What Was the Root Cause of My Challenges? 2](#_Toc191985521)

[Conclusion: What Have I Learned? 3](#_Toc191985522)

[Action Plan: How Will I Apply This Learning in the Future? 3](#_Toc191985523)

[References 4](#_Toc191985524)

## Introduction

This reflective document follows Gibbs' Reflective Cycle (1988) to critically analyse my learning experience while developing the Secure Copyright Management Application (SCMA). The cycle includes six key stages: Description, Feelings, Evaluation, Analysis, Conclusion, and Action Plan. I have included my GitHub Reflective Piece: [Reflective-Piece-for-e-Portfolio](https://mc24887.github.io/Reflective-Piece-for-E-Portfolio/).

## Description: What Happened?

This project focused on creating a secure document storage system with encryption, user authentication, and role-based access control. I faced challenges like authentication failures and encryption key management issues. I overcame these hurdles by using problem-solving techniques, seeking feedback, and enhancing my technical and security skills.

## Feelings: How Did I Feel?

At the project's start, I felt excited and overwhelmed by the complexities of secure application development. My initial attempts at password hashing led to authentication failures, which frustrated me. However, after troubleshooting, I finally achieved a functioning authentication system, which was very rewarding.

Another challenge was ensuring that encrypted files could be consistently decrypted. My first approach resulted in file corruption due to improper key storage, which created significant stress. Ultimately, successfully implementing persistent encryption key storage improved my confidence in my problem-solving abilities.

By the project's end, I felt satisfied and more competent, having gained a solid understanding of secure coding principles and cryptographic implementations.

## Evaluation: What Went Well and What Didn’t?

* **Strong Security Measures**: Implementing AES-256 encryption and SHA-256 password hashing ensured high levels of security.
* **Security Testing**: Using Bandit, Pylint, and Flake8 helped identify and resolve security vulnerabilities early.
* **Role-Based Access Control (RBAC)**: The system effectively enforced access restrictions between Admins and Users.

## What Could Be Improved?

* **Time Management**: Debugging issues took longer than expected, requiring better task prioritisation.
* **Database Optimisation**: Some queries could be further optimised for performance.
* **Code Documentation**: Initially, some functions lacked proper docstrings, which were later added to improve maintainability.

## Analysis: What Was the Root Cause of My Challenges?

**Authentication Issues**

* The first login system failed due to password hashing mismatches.
* Cause: The stored hashes were not compared adequately with input hashes.
* Solution: Debugging the login process and applying .strip() to remove unintended spaces.

**Encryption Key Persistence Issues**

* Encrypted files could not be decrypted after restarting the system.
* **Cause:** A new encryption key was generated each time the program ran.
* **Solution:** Implemented **persistent key storage** to ensure encryption consistency.

**SQL Query Errors**

* Some database queries failed due to **improper parameter binding**.
* **Cause:** SQL queries were formatted incorrectly, leading to errors.
* **Solution:** Used **parameterised queries** to prevent SQL injection and improve security.

## Conclusion: What Have I Learned?

This project significantly enhanced my technical, problem-solving, and security skills. I learned:

* How to implement secure authentication and password hashing.
* How to store and retrieve encrypted files securely.
* The importance of debugging and structured testing.
* How to apply RBAC principles in application security.
* The value of security-focused testing tools like Bandit and Pylint.

## Action Plan: How Will I Apply This Learning in the Future?

I have documented my experience in my GitHub e-portfolio to create a comprehensive record of my reflections and learning journey.

This repository contains additional insights, lessons learned, and key reflections on this project.

|  |  |
| --- | --- |
| **Skill** | **Future Application** |

|  |  |
| --- | --- |
| **Skill** | **Future Application** |
| Cryptography & Security | Implementing Hybrid Encryption Models (AES + RSA). |
| DevSecOps Practices | Learning to integrate CI/CD security pipelines. |
| Cloud Security | Applying encryption techniques in AWS & Azure. |
| Database Security | Strengthening SQL security practices. |
| Time Management | Using Agile methodologies to improve efficiency. |

## References

* Gibbs, G. (1988) *Learning by Doing: A Guide to Teaching and Learning Methods*. Oxford: Oxford Polytechnic.
* Python Software Foundation (2024) *Python Best Practices*. Available at: https://docs.python.org/ (Accessed: 2 March 2025).
* Khan, R. A., Khan, S. U., Han, H. U. and Ilyas, M. (2022) ‘Systematic literature review on security risks and its practices in secure software development’, *IEEE Access*, 10, pp. 5456–5481.
* Buelta, J. (2022) *Python Architecture Patterns: Master API Design, Event-Driven Structures, and Package Management in Python*. Birmingham, UK: Packt Publishing Ltd.
* Cryptography Documentation (2024) *Secure Encryption Using AES-256*. Available at: https://cryptography.io/ (Accessed: 2 March 2025).
* SQLite Documentation (2024) *Best Practices for Secure SQL Databases*. Available at: https://sqlite.org/ (Accessed: 2 March 2025).