Assignment 1

Development Team Project

Question: Develop an application that provides a secure repository for an organisation with domain-specific requirements. In all domains, a user will need to be able to upload, download and share data.

Tutor: Dr Cathryn Peoples

Submission Date: 20 December 2021

Table of Contents

Table of Figures

No of words: 1107

1. Introduction

1.1. Brief Overview

Digitpol, an international Internet Forensics and cybersecurity company in the Netherlands, provides businesses with extensive experience in cybersecurity and cybercrime investigations (DIGITPOL, 2021). Since Digitpol is facing security challenges (digital asset thefts and lack of protection of forensics data), our aim is to revamp their system so that it follows the security best practices (OWASP, 2021; ISO/IEC 27000:2018).

1.2. Problem Statement

Due to various malevolent attacks and tampering of databases (Chopade & Pachghare, 2019), personal and sensitive data require extra protection, especially since they also fall under international privacy laws and regulations such as the GDPR (EU Regulations, 2018). Our main objective is to develop an application that provides a secure repository for Digipol that will allow authorised stakeholders to securely manage their documents in a protected repository which respects data privacy. The three aspects of software architecture – confidentiality, integrity, availability (CIA) – aided by authentication, authorisation, and non-reputability will be applied (Pillai, 2017).

2. Security Considerations

We will abide by the OWASP's list of software vulnerabilities (2021) and STRIDE threat modeling framework (Shostack, 2007). Used "to analyse the architecture and design of a solution even without having any implementation", STRIDE is "perfect for analyzing solutions in a design or prototype state" (Magin et al, 2015).

2.1. User Authentication and Authorisation

The system must:

- implement strict user authentication and authorisation policies (see activity and state diagrams),
- operate over secure web-connections and enable monitoring of user activities via systems audit logs. (see Use Case and ERD diagrams)
- fulfill a user role based and hierarchical authentication model (see UML diagrams).

Authentication reduces attacks from the inside and determines whether the right person is getting access to the authorised information, thus ensuring 'information security' (preservation of confidentiality, integrity, non-repudiation, reliability) (ISO). Moreover, to prevent 'Cryptographic Failures', previously called Sensitive Data Exposure (OWASP, 2021), user's data must be encrypted and decrypted using proper strategies and tools.

The proposed design will follow the different recommendations and best practices in use (OWASP, STRIDE, ISO) to certify that the system meets the most updated security standards. According to OWASP (2021), Broken Access Control has been the most common security risk, which possibly leads to sensitive data disclosure, unauthorised data modification or destruction. Thus, the following recommendations will be observed during the development phases:

- Deny by default
- Enforce Least Privileges

- Validate the permissions on every request
- Rate limit API and controller access

2.2. Security Vulnerabilities

Research shows that most security issues arise from software vulnerabilities, especially in its “configuration, design, and implementation”, and the “lack of knowledge about security concerns” (Medeiros et al, 2017). A table on the different vulnerabilities that can be encountered has been created in Appendix 1. This can help reduce software flaws and bugs and mitigate any damage. These migitations will also come with testings to ensure a proper implementation and good performance. Stress tests will test the system ability to deal with heavy network traffic.

3. Development Methodology

The repository system will be developed and delivered in Agile methodology. This means that quality assurance and user acceptance testing will occur continuously throughout the software development life cycle (SDLC) to ensure acceptance criteria have been met and shippable code is being produced (Srivastava et al, 2017).

3.1. Project Scope

Develop RESTful APIs (Patni, 2017) with user interfaces to support the following functionalities:

- User creation with separate roles and CRUD capabilities accordingly.
- User authentication and authorisation to perform the suite of relevant CRUD capabilities.
- GDPR Compliance Utilities (Requesting / Processing / Encrypting / Updating / Deleting)
- User activity monitoring
- Encryption – decryption
- Regex

3.2. Assumptions

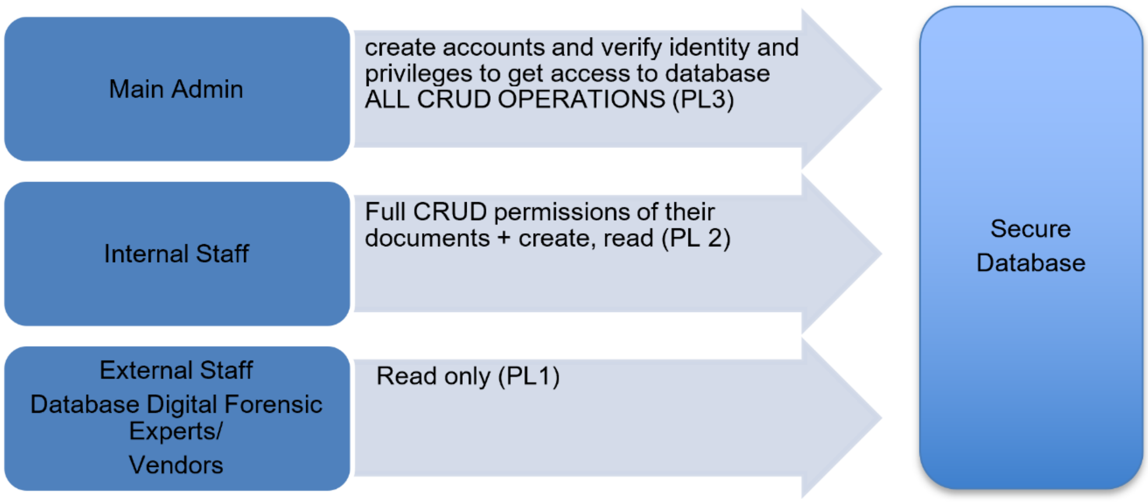
- Only authorised users access and securely manage documents in the repository.
- Three types of users identified: the main admin, the internal staff and external staff. They are cyber/cryptographic professionals who look into digital and cybercrimes (hacking, data breaches etc.)
- External staff:
  - recruited if no expertise exists internally
  - has a limited access to the database
- The “rights to be forgotten” or the right to erasure (EU-GDPR): delete accounts and information about staff upon request.
- Sensitive data on cybercrimes is stored at Digitpol. Users have access to “information beyond publicly available facts” and work “behind the scenes” (Digitpol, 2021)

- API rate limit to prevent DDos exploitation and to have minimal impact on network; server-side request parameters' validation (check requests coming from our system instead of other sources)

3.3. System Design

3.3.1. Users

Different users were identified: some have full control of the system (Admin); others need access to identify, collect and analyse evidence from the database (Sindhu & Meshram, 2012).


Main Admin
create accounts and verify identity and privileges to get access to database ALL CRUD OPERATIONS (PL3)
Internal Staff
Full CRUD permissions of their documents + create, read (PL 2)
External Staff Database Digital Forensic Experts/ Vendors
Read only (PL1)
Secure Database

Note: PL = Permission Level

Figure 1: Users' role description

Specific attention will be given to security, authorisation and authentication and will take into account different access and the least privilege principle (Sanders & Yue, 2019).

3.3.2. Architecture Design

REST APIs play a crucial role in ensuring smooth communication between the client and the server (Patni, 2017; Kolade, 2021). The architecture design will apply as many of the best practices and principles (OWASP, 2021) as possible in terms of API design.

- Use JSON as the format for sending and receiving data
- Use nouns instead of verbs in endpoints
- Use status codes in error handling
- Use nesting on endpoints to show relationships
- Use filtering/sorting/pagination to retrieve the data requested
- Use semantic versioning
- Apply techniques of object-oriented system design including inheritance, composition and polymorphism.

3.4. Software and Operating Environment

The deliverable will be accessible via a web browser. Documentation is included along with the code.

Development Tools and Libraries:

- Programming Language: Python 3.x
- Source Code Version Control: GitHub
- IDE: Visual Studio Code

- Frameworks: Flask and built-in utilities such as Jinja template engine to build user interfaces.
- Database: MySQL
- Password Hashing: Bcrypt
- Encrypt and decrypt data: Fetnet
- Well-presented codes that follow the appropriate styling and naming conventions
- Hashlib.py (for hashing)
- AWS[1]
- Requests.py
- EGRET/Lint: test Regex

3.5. Quality Control

- Pylint: to check for syntax errors.
- PyTest: functional testing
- Optional deliverable due to time constraint: cyclomatic complexity metric

Introduced by Thomos J. McCabe (1976), “Cyclomatic complexity” (CC) metrics, used extensively by industry professionals (Ebert & Cain, 2016), measures the source code complexity of a program (Madhavi, 2016; Sarwar et al, 2013) by producing a number of linearly independent paths through a program module (Ebert & Cain, 2016).

[1] Check: https://mosquitto.org/

3.6. UML diagrams

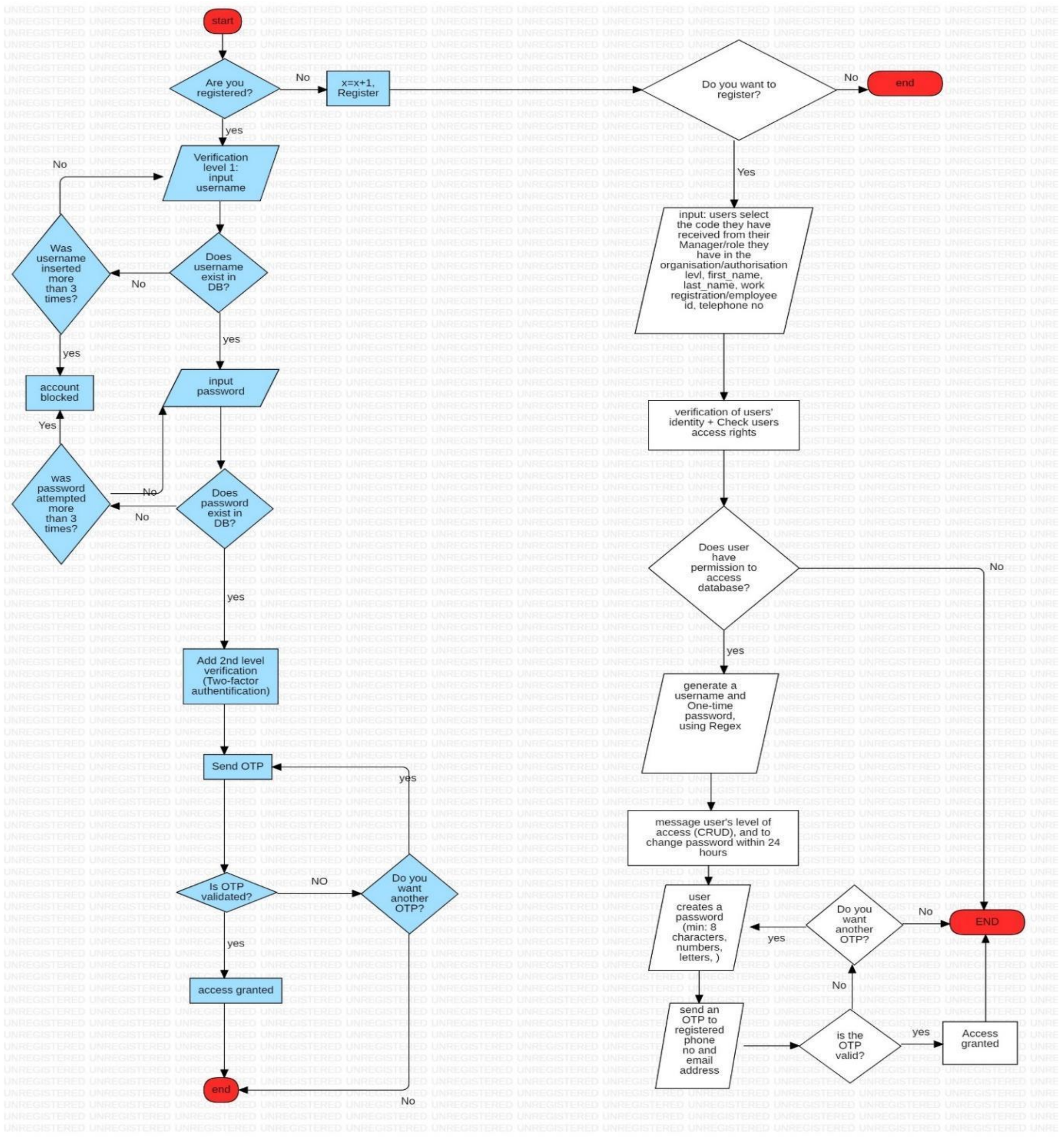

start
Are you registered?
No
x=x+1, Register
Do you want to register?
No
end
Yes
yes
Verification level 1: input username
No
input: users select the code they have received from their Manager/role they have in the organisation/authorisation levl, first_name, last_name, work registration/employee id, telephone no
Was username inserted more than 3 times?
Does username exist in DB?
No
yes
yes
account blocked
input password
verification of users' identity + Check users access rights
Yes
was password attempted more than 3 times?
No
Does password exist in DB?
No
Does user have permission to access database?
No
yes
yes
Add 2nd level verification (Two-factor authentification)
generate a username and One-time password, using Regex
Send OTP
yes
message user's level of access (CRUD), and to change password within 24 hours
Is OTP validated?
NO
Do you want another OTP?
user creates a password (min: 8 characters, numbers, letters,)
Do you want another OTP?
No
END
yes
yes
access granted
No
send an OTP to registered phone no and email address
is the OTP valid?
yes
Access granted
end
No

Figure 2: Security considerations for registration/login

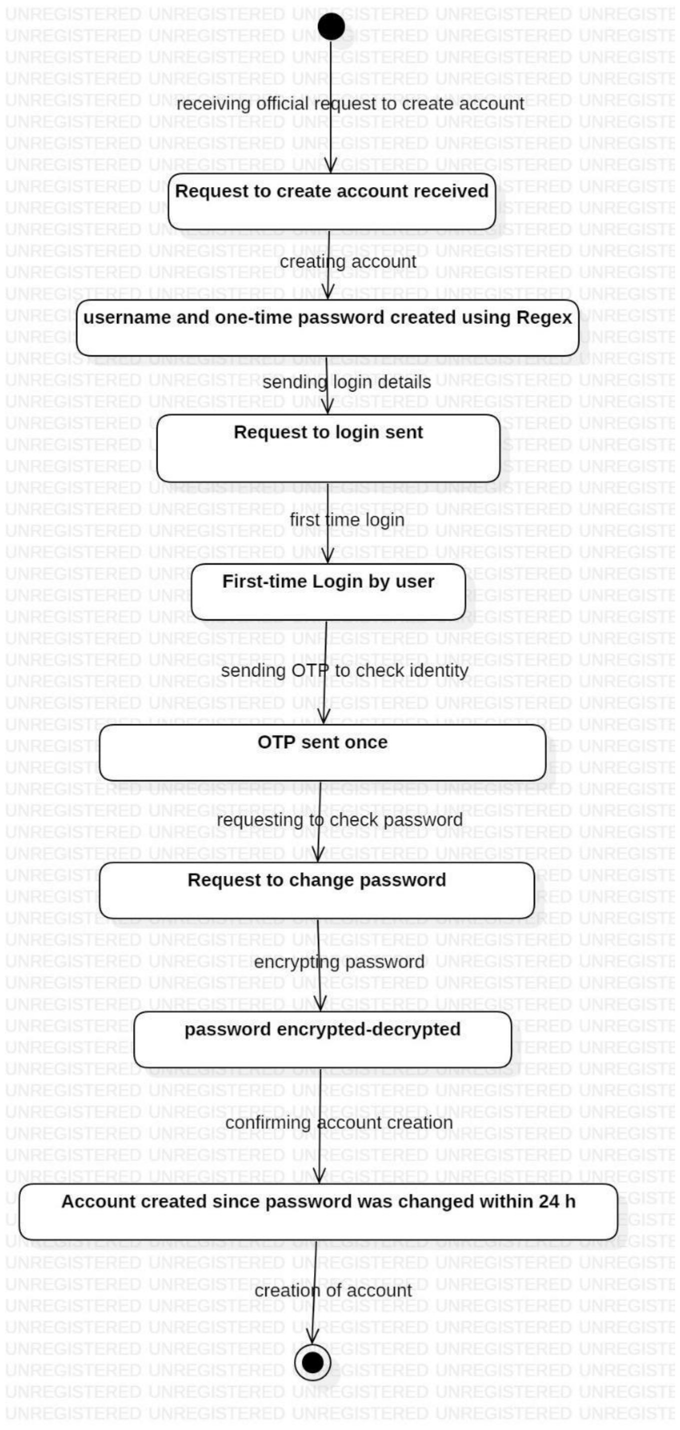

receiving official request to create account
Request to create account received
creating account
username and one-time password created using Regex
sending login details
Request to login sent
first time login
First-time Login by user
sending OTP to check identity
OTP sent once
requesting to check password
Request to change password
encrypting password
password encrypted-decrypted
confirming account creation
Account created since password was changed within 24 h
creation of account

Figure 3: Account creation

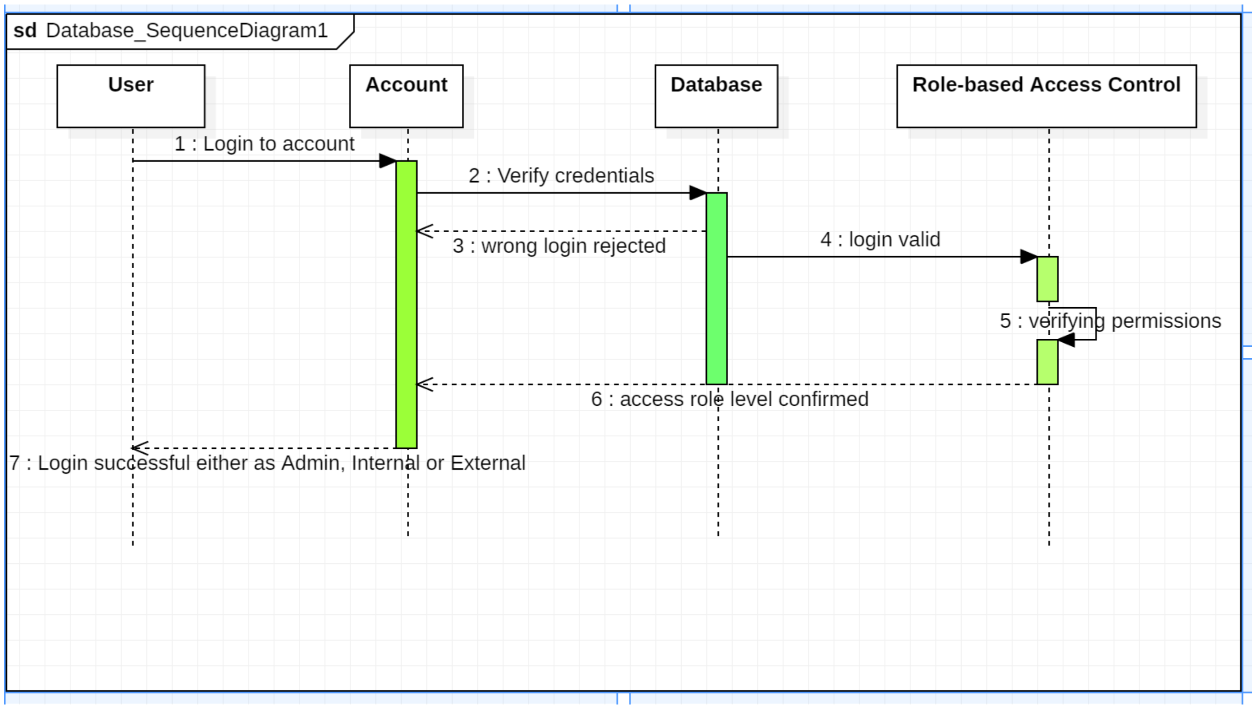

sd Database_SequenceDiagram1
User
Account
Database
Role-based Access Control
1 : Login to account
2 : Verify credentials
3 : wrong login rejected
4 : login valid
5 : verifying permissions
6 : access role level confirmed
7 : Login successful either as Admin, Internal or External

Figure 4: User Access to database

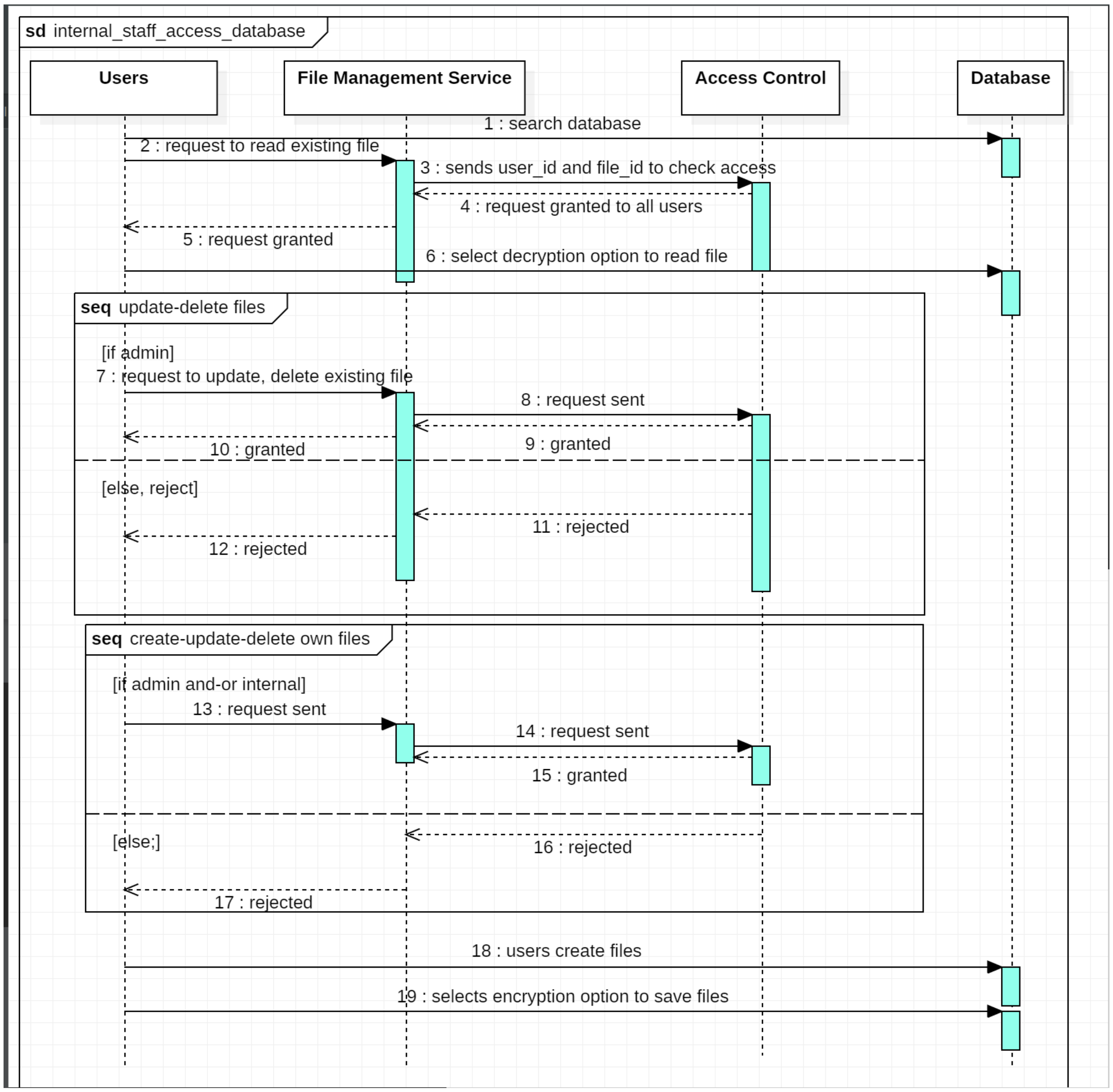

sd internal_staff_access_database
Users
File Management Service
Access Control
Database
1 : search database
2 : request to read existing file
3 : sends user_id and file_id to check access
4 : request granted to all users
5 : request granted
6 : select decryption option to read file
seq update-delete files
[if admin]
7 : request to update, delete existing file
8 : request sent
9 : granted
10 : granted
[else, reject]
11 : rejected
12 : rejected
seq create-update-delete own files
[if admin and-or internal]
13 : request sent
14 : request sent
15 : granted
[else;]
16 : rejected
17 : rejected
18 : users create files
19 : selects encryption option to save files

Figure 5: Access Control to Database (CRUD operations)

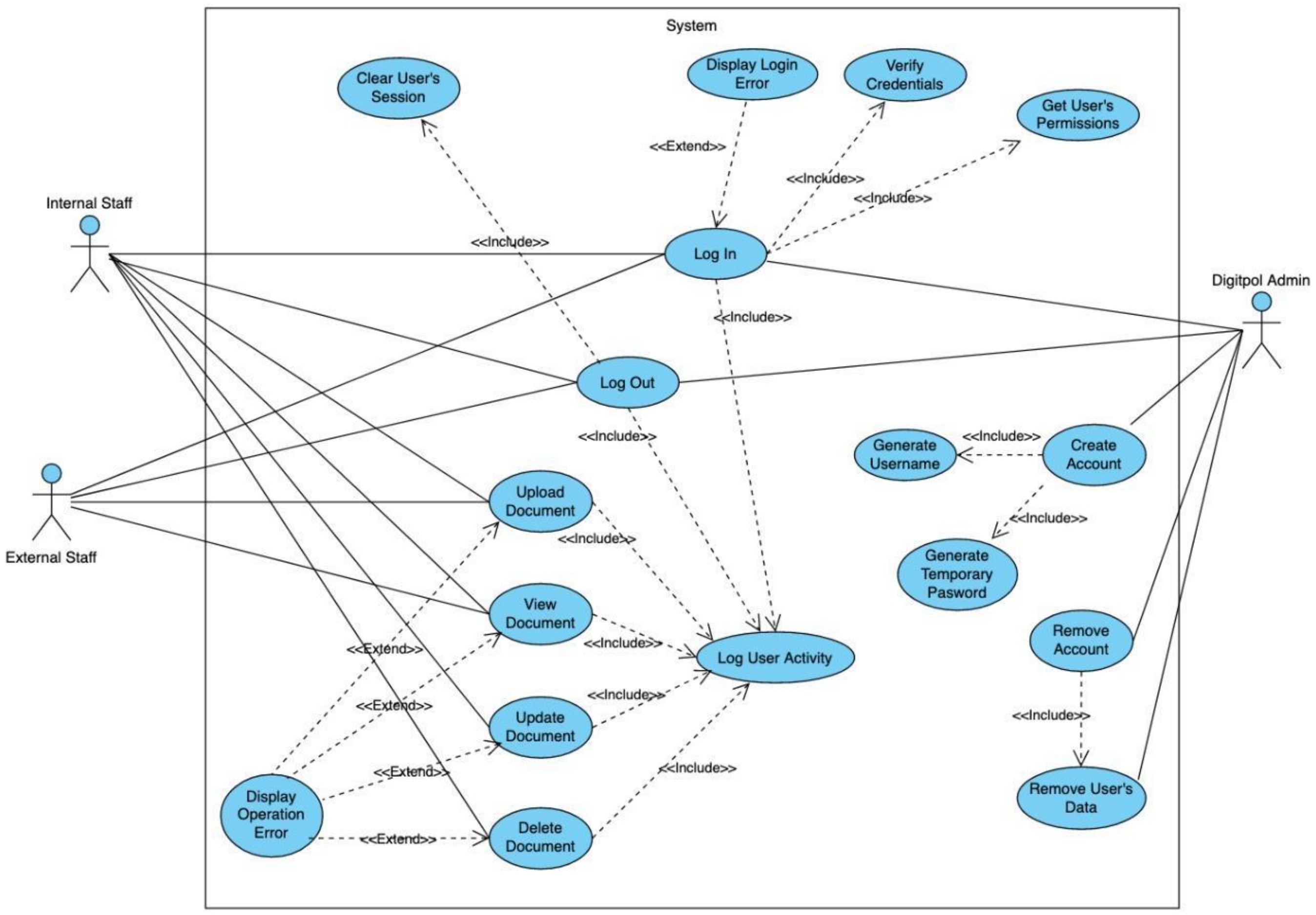


Figure 6: Use Case Diagram

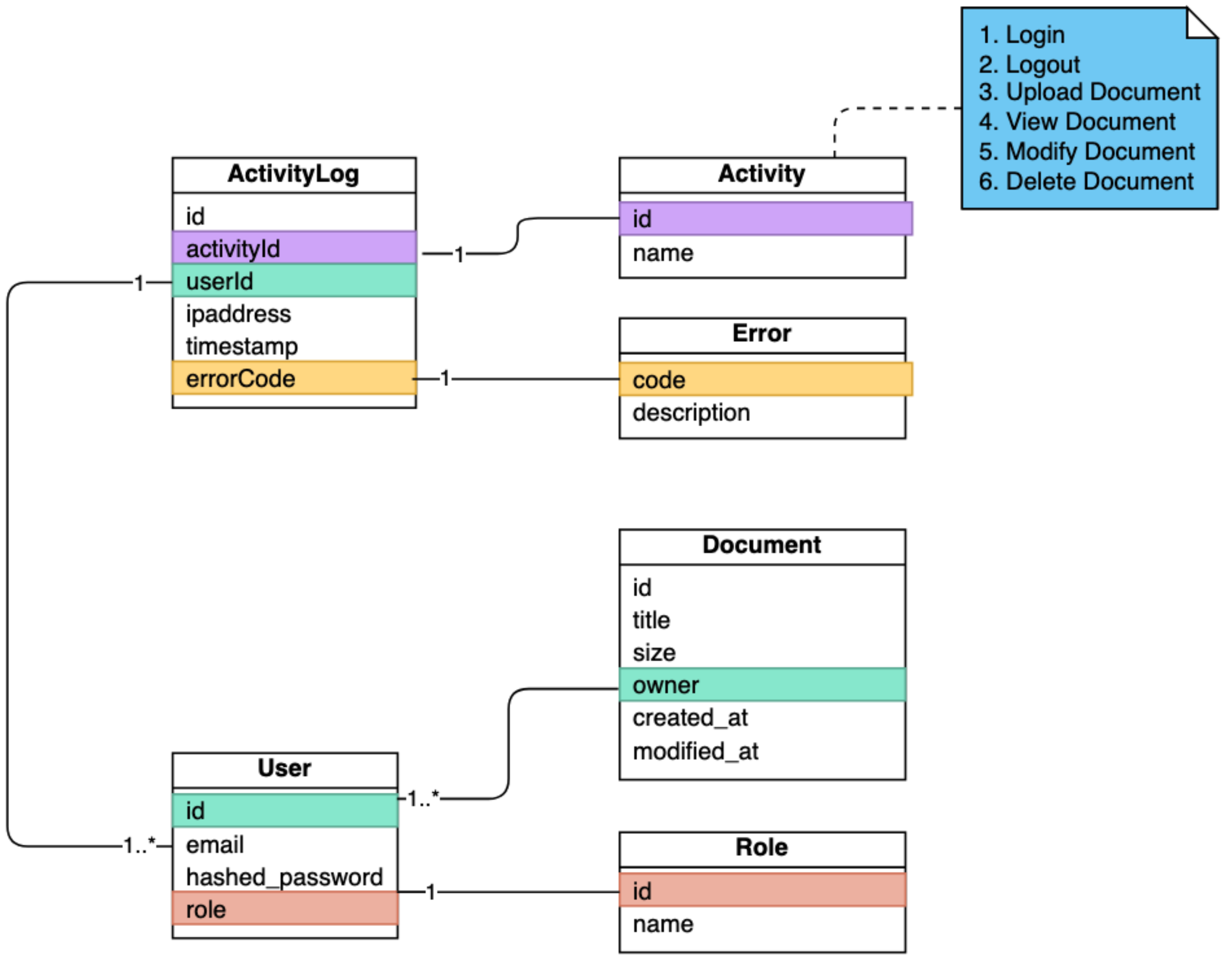


Figure 7: ERD Diagram

4. Appendix 1 – Security Vulnerabilities Table

Threat	Impact	Cause	Mitigation
Broken access control	Extreme (Highest)	Failure in properly authorising users	● Strong authentication policy ● Deny access to functionality by default. ● Clean code with binary access controls ● Use access control lists and role-based authentication mechanisms ● Handle access controls at server-side ● Constant testing and auditing of access controls ● Enforce record ownership
Sensitive data exposure	Varied	Occurs as a result of not adequately protecting the information where it is stored	● Evaluate the risk of third-parties ● Monitor all network access ● Identify all sensitive data ● Encrypt all data ● Evaluate all permissions ● Monitor the security posture of all vendors
Denial of Service (Erickson)	Moderate	Multiple systems flooding the bandwidth or resources of a targeted system	● Prepare a DDoS response plan ● Improve network security ● Ensure server redundancy ● Look for the warning signs ● Limit network broadcasting ● Use cloud-based protection ● Set up continuous monitoring
Injection	Extreme	Exploiting a vulnerable web application that doesn't properly validate user input	● Validate user input ● Thoroughly inspect API functions ● Protect sensitive data ● Implement output encoding
Buffer overflow	Moderate	Occurs when data written to a buffer also corrupts data values in memory	● Performing routine code auditing (automated or manual). ● Using compiler tools such as Libsafe and StackShield.

		addresses adjacent to the destination buffer due to insufficient bounds checking.	● Using safe functions such as strncat instead of strcat, strncpy instead of strcpy, etc ● Periodically scanning the application.

5. References

Cifuentes C. & Bierman, G. (2019) “What is a Secure Programming Language?”, 3rd Summit on Advances in Programming. Available from: https://drops.dagstuhl.de/opus/volltexte/2019/10546/pdf/LIPIcsSNAPL-2019-3.pd [Accessed 25 November 2021]

Chopade, R. & Pachghare V.K. (2019) “Ten years of critical review on database forensics research”, *Digital Investigation*. Elsevier. Available from: https://doi.org/10.1016/j.diin.2019.04.001 [Accessed 25 November 2021]

DIGITPOL (2021) *Netherlands Digital Forensic Investigation - DIGITPOL*. Available from: https://digitpol.com/netherlands-digital-forensic-investigation/. [Accessed 20 November 2021].

Ebert, C. & Cain, J. (2016). “Cyclomatic Complexity—40 Years Later”, *Cyclomatic Complexity. IEEE Software, 33(6), 27–29.* Available from: doi:10.1109/MS.2016.147 [Accessed 16 December 2021]

Erickson, Jon (2003) *Hacking: The Art of Exploitation*. San Francisco, No Starch Press.

European Data Protection Regulation (2018) *(EU) 2016/679 General Data Protection Regulation (GDPR)*. Available from: https://gdpr-info.eu/ [Accessed 30 November 2021]

ISO. (N.D) ISO/IEC 27000:2018: “3 Terms and Definitions”. Available from: https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en [Accessed 16 November 2021].

Kolade, C. (September 16, 2021), REST API Best Practices – REST Endpoint Design Examples, freeCodeCamp. Available from: https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/ [Accessed 4 December 2021]

Madhavi, D. (2016) “A White Box Testing Technique in Software Testing: Basis Path Testing”. *Journal for Research,* Volume 02, Issue 04.

Magin D et al. (2015) “Security Analysis of OpenRadio and SoftRAN with STRIDE Framework”. *The 24th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, Las Vegas, Nevada, USA. Available from: http://publica.fraunhofer.de/documents/N-360563.html [Accessed 25 November 2021]

Medeiros, N. et al (2017) "Software Metrics as Indicators of Security Vulnerabilities," *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*. Available from: doi: 10.1109/ISSRE.2017.11. [Accessed 16 December 2021]

Olivier M.S. (2009) “On metadata context in Database Forensics”, *Digital investigation*. Available from: https://doi.org/10.1016/j.diin.2008.10.001 [Accessed 22 November 2021]

Olsen, N. (January 22, 2021) “GDPR Compliance Statement”, PrivacyPolicies.com. Available from: https://www.privacypolicies.com/blog/gdpr-compliance-statement/ [Accessed 20 November 2021].

OWASP (Open Web Application Security Project). (2021) *C7: Enforce Access Controls*. Available from: https://owasp.org/www-project-proactive-controls/v3/en/c7-enforce-access-controls. [Accessed 20 November 2021].

OWASP. (2021) *OWASP Top Ten Web Application Security Risks*. Available from: https://owasp.org/www-project-top-ten/ [Accessed 20 November 2021].

Patni, S. (2017) *Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS*. California, Apress. Available from: DOI 10.1007/978-1-4842-2665-0 [Accessed 25 November 2021]

Peoples, C. (2021) Lecture Notes. University of Essex Online November 2021.

Pillai, A.B. (2017) Software Architecture with Python, Birmingham, Packt.

Sanders, M. & Yue, C. (2019) "Mining Least Privilege Attribute Based Access Control Policies". In *2019 Annual Computer Security Applications Conference* (ACSAC '19), December 9–13, San Juan, PR, USA. ACM, New York, NY, USA, Available from: https://doi.org/10.1145/3359789.3359805 [Accessed 25 November 2021]

Sarwar et al (2013) "Cyclomatic Complexity: The Nesting Problem". Available from: DOI: 10.1109/ICDIM.2013.669398 [Accessed 16 November 2021]

Shostack, A. (2007) "Stride Chart", *Microsoft Security*. Available from: https://www.microsoft.com/security/blog/2007/09/11/stride-chart/ [Accessed 4 December 2021]

Sindhu, K. K. & Meshram, B. B. (2012) "Digital Forensics and Cyber Crime Datamining", *Journal of Information Security* 3(3). Available from: DOI:10.4236/jis.2012.33024 [Accessed 25 November 2021]

Srivastava et al. (2017) "SCRUM Model for Agile Methodology", *International Conference on Computing, Communication and Automation* (ICCCA2017), IEEE. Available from: DOI: 10.1109/CCAA.2017.8229928 [Accessed 4 December 2021]