2022

DESIGN DOCUMENT

GROUP 5

DERVISHEV, OKAY; LAPIERRE, PATRICIA; RIGAS, IASON; ROONGTA, YASH; FABRICE WOUCHE

UNIVERSITY OF ESSEX ONLINE | SSD_PCOM7E NOVEMBER 2022

Content Table

Figure List

1. Introduction

This document presents the models used to create and direct the solution. These solution's models presented in this document answer the business needs translated into requirements.

Table 1: Architectural views

View-diagram	Objectif	Element	Diagram's location in this report
Target business process model	Illustration of the desired business process which shows the high level of the steps in the process.	Steps in the process	See *Section 2 - Figure 1 - Process model*
Context	Project scope overview representation, which shows links between different parts of the solution	Zones, Structure, Components	See *Section 3 - Figure 2 – Context view*
Use Case in Unified *Modelling Language (UML)*	Definition of interaction between actors/system by software application	Actor Use Case	See *Section 4 - Figure 3 – Use Case diagram UML*
Activity in *Unified Modelling Language (UML)*	Linkage between functions in use case	Actor Use Case Software applicative Function	See *Section 5 - Figure 4 - Activity diagram UML with boundaries*

a. Goal

The solution's goal is to allow different types of users to interact with the Dutch Police Internet Forensics' documentation management system.

b. Scope

i. General

This solution model is intended to replace the actual solution for the following reasons:

- Ensure the centralization of reports in one repository, and
- Avoid reports' duplication.

ii. Utilisation

The new solution is the principal input in the content management solution.

c. Business approaches

i. Team dynamic

Elements	Selections
Engagement	● 2 meetings/week: 1 weeknight + 1 weekend day ● Duration: max 2h/meeting
Planning schedule	Week 1: Group formation Week 2: Draft of the Design Document Week 3: Deadline – 28th November 2022-23h55 Final of the Design Document Week 4: Draft of the Coding Output Week 5: Draft of the Coding Output Week 6: Deadline – 19th December 2022-23h55 Final of the coding Output
Communication channels	● Documentation: OneDrive ● Quick answers: WhatsApp Chat
Responsibility Assignment Matrix (RAM)	● Iason: Submits the assignments to the portal ● Yash: Provides links to meetings and technology support ● Patricia: Organises the assignment's structure ● Okay: Does additional research ● Fabrice: Provides Quality Insurance on assignment

ii. Development paradigms

Elements	**Selections**
Development paradigm	Software Development Life Cycle (SDLC): ● Conceptual definition ● Functional requirements determination ● Design review ● Coding ● Code review walk-through ● System test review ● Maintenance and change management (Chapple, Steward, and Gibson, 2021)
Dynamic of development	Hybrid: Agile and Traditional
Architecture	On Premise, Internal
Code sharing platform	GitHub

d. Security

i. Confidentiality grade

Considering the sensitive information in the content management system, every part of the solution implementation is highly confidential.

e. Reference documents

i. Security approaches

Safety should be considered at every stage of development, especially for critical applications that handles sensitive information. It is possible to avoid costly and potentially dangerous security breaches by building security into the system. *Closed systems* are challenging to integrate, but they are more secure. The product should have a *default configuration* that is easy to install and secure. The *Keep it Simple* principle is the encouragement to avoid overcomplicating the environment, organisation, or product design. The more complex a system, the more difficult it is to secure (*Ibid*).

ii. Technical approaches

Confinement, *bounds*, and *isolation* make designing secure systems more difficult, but they also make it possible to create more secure systems. *Confinement* ensures that an active process can only access specific resources (such as memory). *Bounds* are the limitation of authorisation assigned to a process to limit the resources the process can interact with, and the types of interactions allowed. *Isolation* is the key to confinement, the physical or logical separation of resources to prevent unauthorised access. *Access control* is a security measure only authorised users can access a computer system or network.

iii. Assumptions

In the event there's a compatibility issue with the components, equivalent alternative components or libraries will be used which will be marked as addendums to appendix in the original design document with version iterations.

2. Target business process

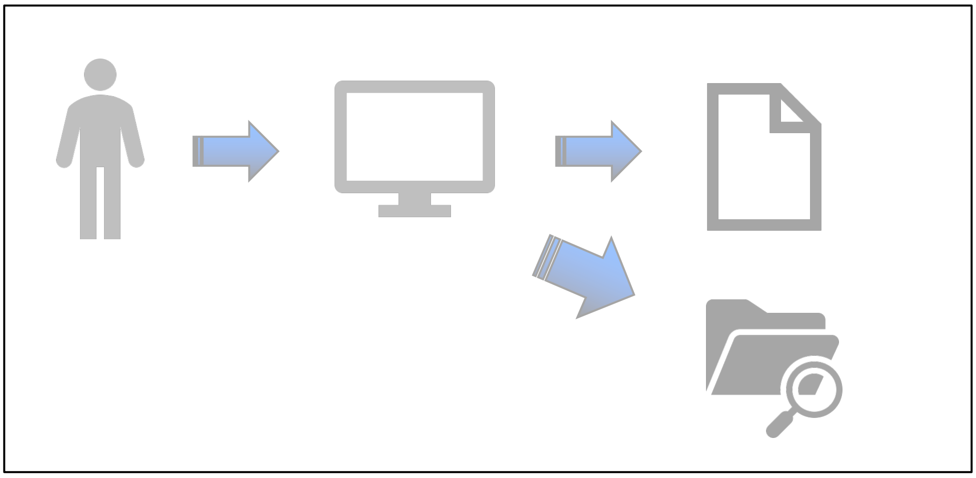
a. Definition

Table 2: Target Process

Process Name	Input	Goals	Scope
Content management	● New files ● Creation of new users ● Secret Token for the MFA for users	● Allowing the right people to have access to elements in the content management solution ● Allowing the right people to upload new police files ● Allowing a follow-up by admin people	This process applies to the content management of all the Dutch Police Internet Forensics - internal use only.

b. Target business process model

Figure 1 - Process model



3. Context view

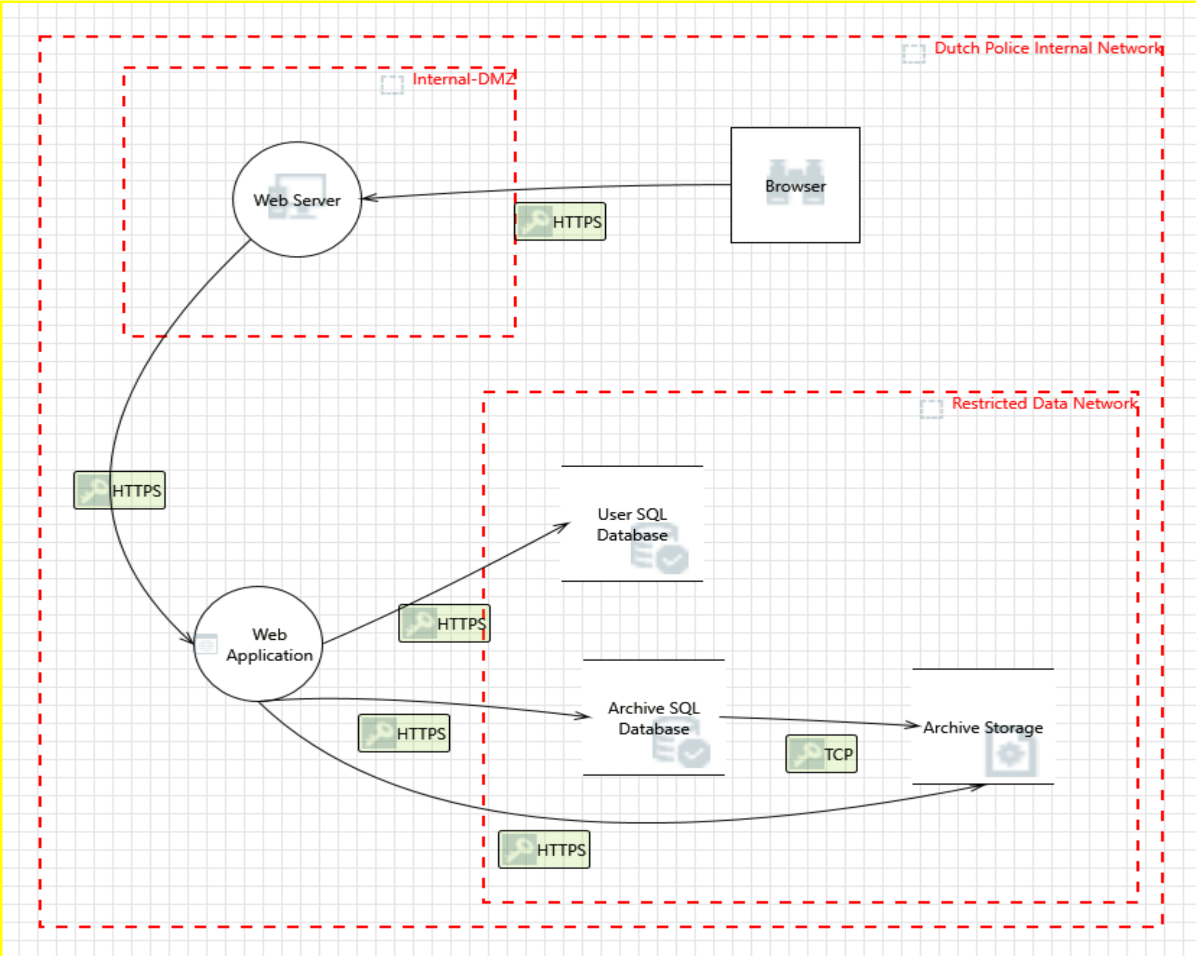
a. Description

The solution is divided in three sub-contexts:

Elements	Selections
Dutch Police Internal Networking	All the components: ● 2 Ubuntu 20.04/18.04 Webserver/Application (1) ● database server (1).
Internal-DMZ	Web server situated in the demilitarized zones
Restricted Data Network	Databases and archive storage located in the restricted data network.

b. Context view diagram

Figure 2 – Context view



c. System requirements

Component	Requirement	State	Scaling Method
Processor	Dual Core Processor	Scalable	Migration
RAM	4GB DDR4 RAM	Scalable	Migration
Local File Storage	500 Gigabytes	Scalable	Expansion
Network/Cloud File Storage	1 Terabytes	Scalable	Migration/Expansion

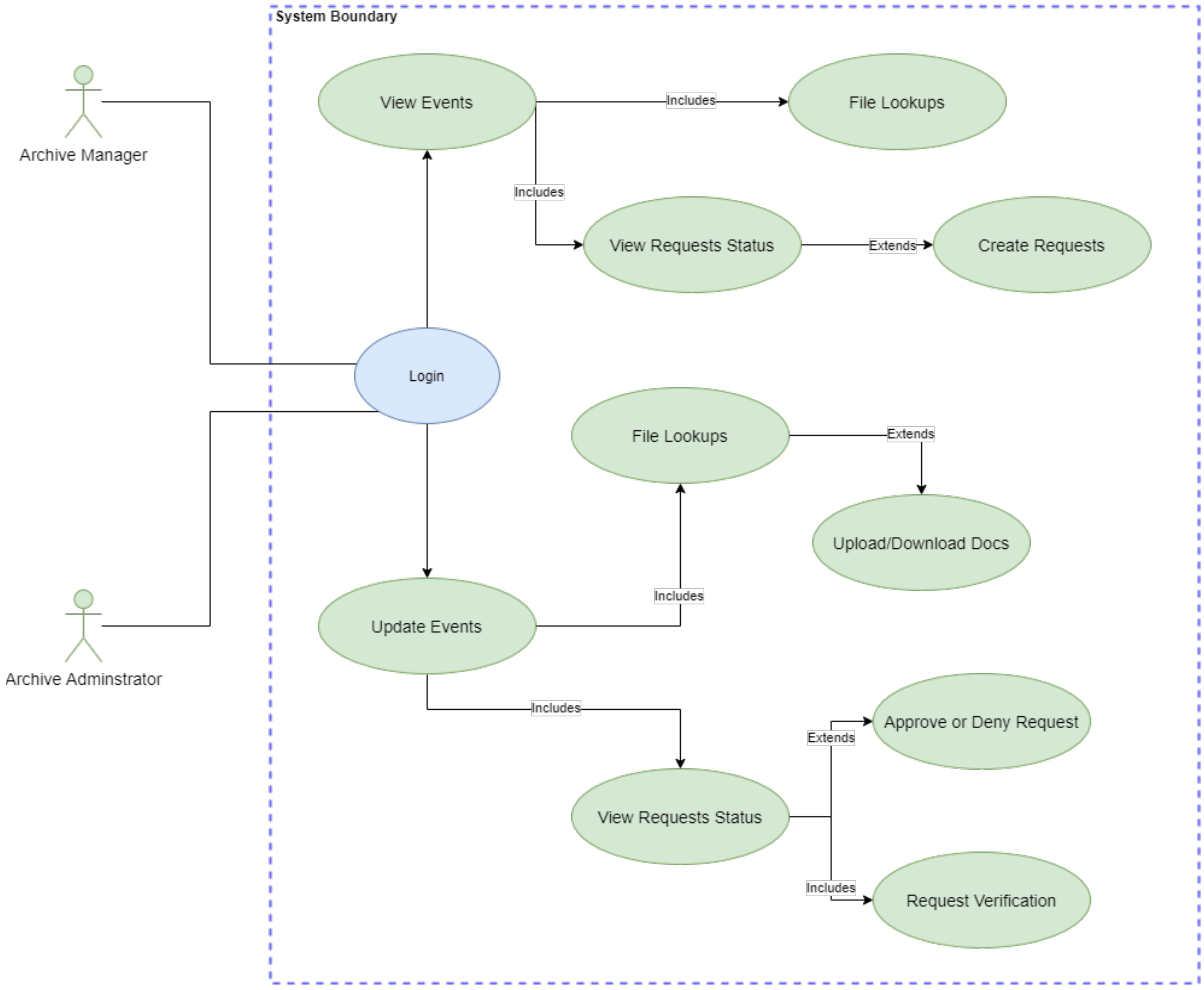
4. Use Case

a. Description

Use case identifier		
Interact with the content management solution		
Precondition		
The user is authorised to access the content management solution The user has it credentials to log in A Valid Request is available to map a particular file(s) request Relevant approvals to the requests are recorded into the system		
Postcondition		
File(s) desired are present in the system File(s) desired are retrieval		
Error situation		
User’s credentials are not valid File(s) desired do not exist in the system File(s) desired have lapsed the retention period and are no longer available in the system.		
Actors		
Actor type	Archive Manager	Archive Administrator
Trigger	Require access to the content management solution	Require access to the content management solution
Role	Non-Administrative	Administrative
Access right	Limited	Limited
Possible actions	(1) Log in in the content management solution (2) Create files (C) (3) Read files (R)	(1) Log in in the content management solution (2) Create files (C) (3) Read files (R) (4) Update files (U) (5) Delete files (D)
Constraints	(1) Update files (U) (2) Delete files (D)	

b. Use Case view diagram

Figure 3 – Use Case diagram UML



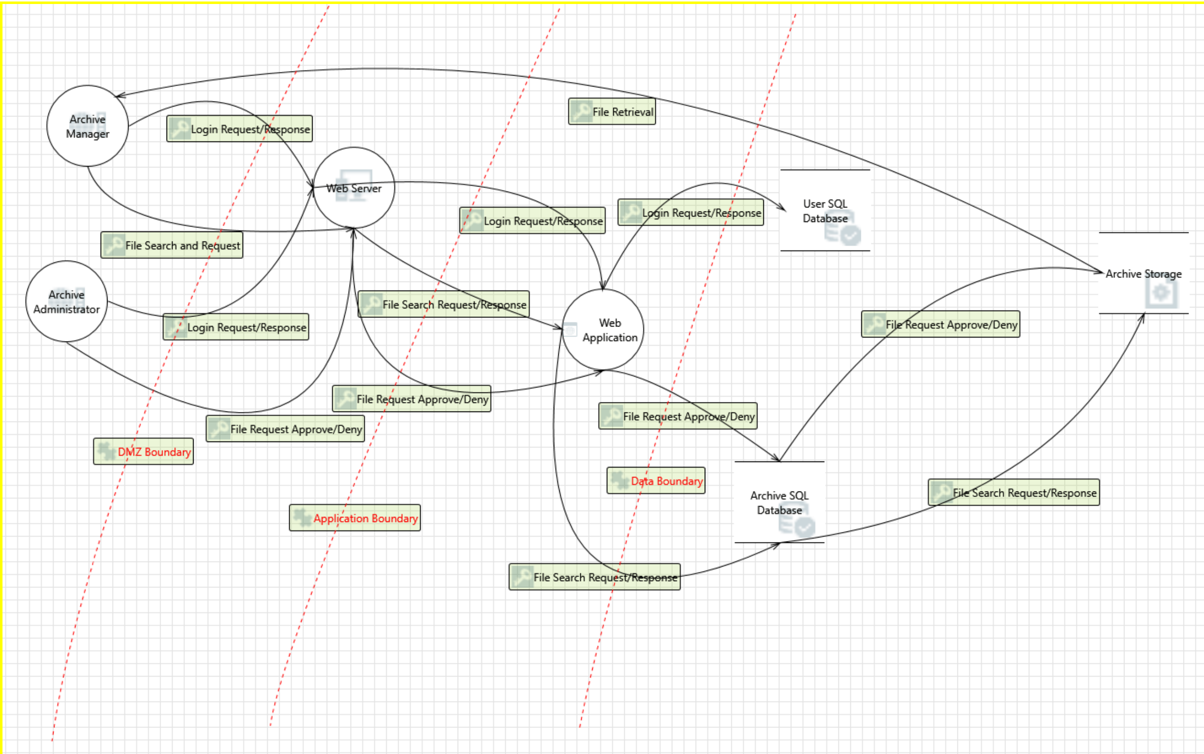
5. Activity Case

a. Description

Network boundaries separate the system on a network level. This aids in preventing unauthorised traffic and free hops to-and-from the systems for a regular user.

b. Activity view diagram

Figure 4 - Activity diagram UML with boundaries



c. Design decision

Component	Selections	Reasons justifying the selections
Data In Transit	HTTPS and relevant secure protocols such as SFTP, SSH, VPN over SSL wherever applicable.	HTTPS and equivalent secure protocols with strong cipher suites in place would quash any attempts of Man-In-The-Middle (MiTM) attacks or network sniffing. (Durumeric et al., 2017).
Data At Rest	BitLocker or equivalent secure encryption for the physical drive.	In the event the physical drive is readable to an unauthorised entity, any read/write memory allocation would first prompt for a bitlocker key. Without this key it's not possible to read and write data to disk (Kornblum, 2009).
Data in Database Store	Salting and hashing for sensitive data such as passwords.	Protects sensitive data in the event of a data breach or data leak, making offline cracking of leaked passwords much more complex (Sriramya and Karthika, 2015).
Application Framework	Django Framework	Django uses Model-View-Controller (MVC) pattern which separates internal representations for the user

		between the information viewed and accepted (Mallawaarachchi, 2017).
Application Code	Python 3.x	Python is an Object programming language

6. Security by Design Components

a. Description

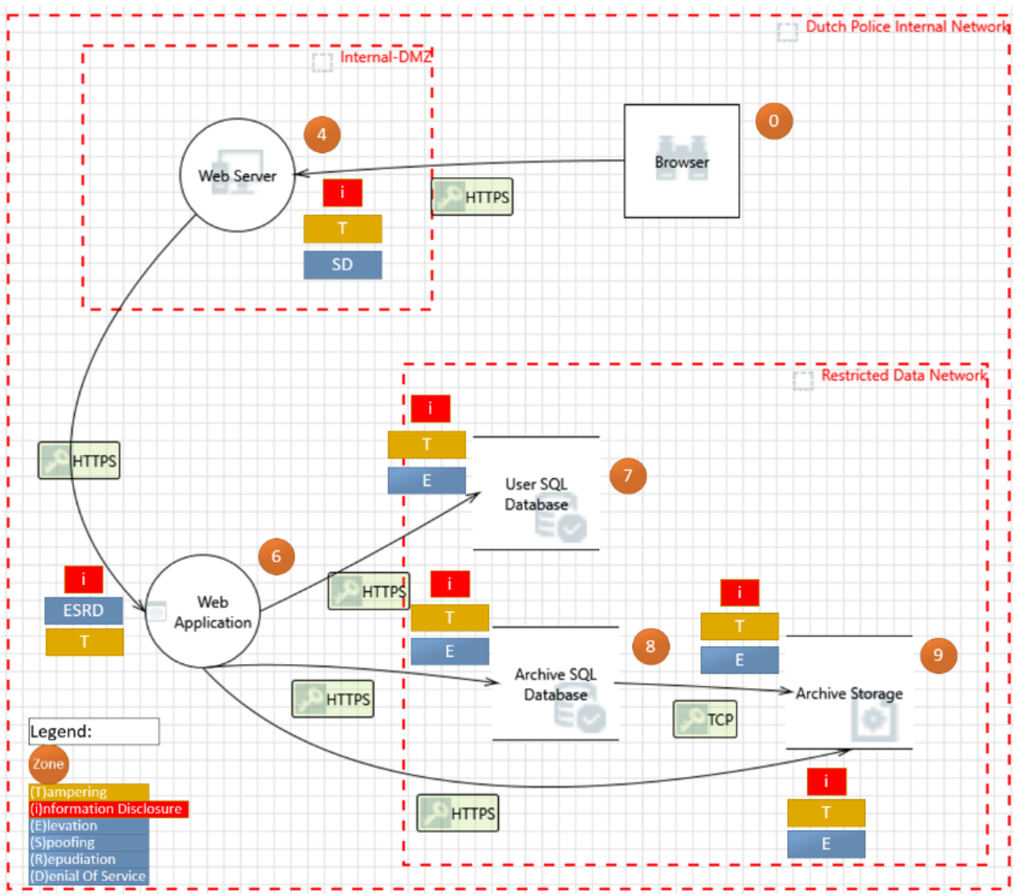
Component	Approach	Tools, Technology, Procedure	Reasons justifying the selections
Threat Model	STRIDE following Rapid Threat Model Prototyping [RTMP]	Microsoft Threat Modelling Tool	The most popular STRIDE based threat model tool, widespread adaptation and gives granular control over each object. Automated Risk Analysis also provided within the tool (Kamongi et al., 2014)
Web Security	OWASP Top 10	BurpSuite	Widely adapted semi-automated tool for web penetration tests. Allows to intercept and modify requests with the help of a proxy. “An open source Burpsuite plugin that identifies SSO protocols automatically in a browser’s HTTP traffic and helps penetration testers and security auditors to manipulate SSO flows easily.” (Mainka et al., 1970)
Code Review	OWASP ASVS	Pylint, Semgrep [Default YAML]	Open-Source tools to perform static analysis on the code, discovers any significant security bug or insecure implementation early in the SDLC, preventing expensive rework post deployment.

Vulnerability Scan	Automated vulnerability scanning	Nessus Essentials	Vulnerability Scanner for the installed server and its underlying libraries, binaries, packages and more. Gives insight of architectural risks that could lead to remote code execution or privilege escalations.
Risk Scoring	CVSS	CVSS 3.1	Widely adapted risk-scoring framework, used to prioritise security risks discovered. CVSS aims to help create consistent scores that accurately show the impact of vulnerabilities, considering each user's environment (Mell, Scarfone and Romanosky, 2006)

b. Security risks, vulnerabilities & mitigations

For threat modelling, the rapid threat model prototyping methodology (Hill, 2022) was used in combination with STRIDE, OWASP Top 10, and the ATT&CK framework. Initially, the zones of trust were identified in the system diagram with trust ranging from 0 (no trust) to 9 (high trust). Subsequently, using STRIDE for each model element STRIDE values were assigned as shown below:

Figure 5 – Security system diagram



The threats identified by STRIDE were mapped to the corresponding OWASP Top 10 risks for each of the threat model elements above:

Model Element	OWASP top 10 risks
Browser to Web Server	A2- Cryptographic Failures A3- Injection A7- Identification and Authentication Failures A9- Security Logging and Monitoring Failures A10- Server-Side Request Forgery
Web Server to Web Application	A1- Broken Access Control A2- Cryptographic Failures A3- Injection A7- Identification and Authentication Failures A9- Security Logging and Monitoring Failures A10- Server-Side Request Forgery

Web Application to User SQL Database	A2- Cryptographic Failures A3- Injection
Web Application to Archive SQL Database	A2- Cryptographic Failures A3- Injection
Web Application to Archive Storage	A2- Cryptographic Failures A3- Injection A8- Software and Data Integrity Failures
Archive SQL Database to Archive Storage	A2- Cryptographic Failures A3- Injection
All Elements	A4- Insecure Design A5- Security Misconfiguration A6- Vulnerable and Outdated Components A8- Software and Data Integrity Failures

Subsequently by considering list of possible mitigations in OWASP top 10, CWE and ATT&CK the following list of mitigations was composed per risk identified:

OWASP Top 10	Applicable mitigations
A1- Broken Access Control	● Deny by default ● Implement access control mechanism ● Disable web server directory listing and remove metadata from web roots ● Log access control failures
A2- Cryptographic Failures	● HTTPS (TLS) ● AES256 for stored data?
A3- Injection	● Input validation and sanitization, ● Escape special characters ● Use safe API ● Use LIMIT and SQL controls to prevent mass disclosure of records
A4- Insecure Design	● Establish secure development lifecycle ● Use threat modelling

	• Write unit and integration tests to validate all critical flows
A5- Security Misconfiguration	• Minimal platform, do not install unused features and frameworks • Review and update configurations, patch management and review cloud storage permissions • Segmented application architecture
A6- Vulnerable and Outdated Components	• Remove unused dependencies and unnecessary features • Inventory the versions of both client-side and server-side components • Obtain components from official sources over secure links • Monitor for libraries and components that are unmaintained
A7- Identification and Authentication Failures	• Implement Multi Factor Authentication • Password policy (NIST guidelines) • Limit failed login attempts • Testing password against top 10000 password list
A8- Software and Data Integrity Failures	• Verify software with digital signatures • Ensure libraries and dependencies from trusted repository • Use OWASP dependency checker • Review process for the code • Ensure code integrity
A9- Security Logging and Monitoring Failures	• All login, access control, and server-side input validation failures shall be logged • Logs are encoded in appropriate format
A10- Server-Side Request Forgery	• Sanitise and validate input data • Disable HTTP redirections

7. Tools and Libraries

Tools and Libraries	Selections	Reasons justifying the selections
Application OS	Ubuntu 20.04/18.04	Most popular open source Linux platform, wide use and adaptability increases overall compatibility for packages.
IDE	Visual Studio Code	User-friendly while providing all the functionalities
Backend Coding Platform	Python 3.10.6	See *Section 5 - Activity Case*
Frontend UI	HTML5/CSS	User friendly web graphics development
Web Framework	Django 4.1.3	See *Section 5 - Activity Case*
Database	MySQL 8.x	Compatible with Django and has high compatibility.
SSL Certificates	LetsEncrypt	Open-source certificate issuer, reduces costs and allows for easy deployment through automated scripts.
Cryptography	- Hashing: SHA1/SHA256/SHA512 - Salt: salt.modules.mysql	Protects sensitive data in the event of a data breach or data leak, makes offline cracking of leaked passwords much complex
Libraries	● OpenSSL ● Totp ● mysql-connector-python	OpenSSL permits the use and creation of certificates as relevant to secure and validate HTTPS communications python totp allows smooth creation of MFA tokens. mysql-connector-python library enables easy translation of python to mysql connection to run SQL queries via code.

*Note: This list is not exhaustive, only intends some libraries to showcase major libraries in use.

8. Environment

a. Non-Production

The non-production environment (development and staging/User Acceptance Testing) will be used to validate our installation and deployment procedures. The infrastructure installed will remain and reuse during upgrades.

b. Production

The production infrastructures will be used to interact with the documentation management system.

Reference

Chapple, M; Steward, J. M.; and Gibson, D (2021). CISSP Certified Information Systems Security Professional – Official Study Guide – 9th Edition, *John Wiley & Sons, Inc,* 1250 pages.

Durumeric, Z., Ma, Z., Springall, D., Barnes, R., Sullivan, N., Bursztein, E., Bailey, M., Halderman, J.A. and Paxson, V. (2017). The Security Impact of HTTPS Interception. Proceedings 2017 Network and Distributed System Security Symposium. [Online]

Gesellschaft für Informatik e.V. Available from: https://dspace.gi.de/handle/20.500.12116/1977 [Accessed 27 November 2022].

Hill, G. (2022). Rapid Threat Model Prototyping. Available from: https://github.com/geoffrey-hill-tutamantic/rapid-threat-model-prototyping-docs [Accessed 20 November 2022].

Kamongi, P., Gomathisankaran, M. and Kavi, K. (2014) *Nemesis: Automated Architecture for Threat Modeling and Risk Assessment for Cloud Computing.* Available from: https://csrl.cse.unt.edu/kavi/Research/PSSAT-2014.pdf [Accessed 20 November 2022].

Kornblum, J.D. (2009). Implementing BitLocker Drive Encryption for forensic analysis. *Digital Investigation*, 5(3-4), pp.75–84. [Online].

Mainka, C. *et al.* (1970). Automatic recognition, processing and attacking of single sign-on protocols with BURP suite, Startseite - Digitale Bibliothek - Gesellschaft für Informatik *e.V.* Gesellschaft für Informatik e.V. Available from: https://dspace.gi.de/handle/20.500.12116/1977 [Accessed 27 November 2022].

Mallawaarachchi (2017). 10 Common Software Architectural; Patterns in a nutshell. Available from: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013 [Accessed 18 November 2022].

Mell, P., Scarfone, K. and Romanosky, S., 2006. Common vulnerability scoring system. *IEEE Security & Privacy*, *4*(6), pp.85-89.

Seidl, M.; Scholz, M; Huemer, C.; and Kappel, G. (2012). UML @ Classroom – An Introduction to Object-Oriented Modeling, *Springer*, 215 pages.