Design Document: Cyber Crime Awareness Platform

Introduction

The increased demand in online activities have resulted in higher chances of cyber threats such as hacking, identity theft and data breaches (Veale & Ian, 2020). Therefore, IT systems will continue to be vulnerable to frequent attacks.

The National Centre for Cyber Security (NSCS) in the Netherlands is part of the Ministry of Justice and Security that is responsible for the security of the Dutch government and provides a unified approach to address cybercrimes and ICT security (Kaska, 2015). The NSCS provides a portal that supports and guides organizations with security concerns by providing a wide range of technical solutions that includes vulnerability management, risk management and incident response. Similarly, A web application will be designed to prompt cyber security awareness through the following functionalities:

1. Submission of attempted cybercrime or system vulnerability concerns by the public.
2. Providing immediate response to the reporting individual according to the case topic.
3. Assigning an officer to address/advise on reported system vulnerabilities.

Application Description & System Features

The proposed application will be a web portal for NSCS staff to create, read, update and delete submitted security concerns. Hence it will allow enrolled members of the public to log security cases, track the progress of their reports, and upload supporting files. There will be levels of authentication which users must go through in order to log into the system.

- Accessed through a web application.
- A portal to upload security concerns and supporting documents.
- Registering security concerns as a case ID.
- Load balancer to control traffic and avoid system overload.
- Log Monitoring of all user activities.
- Input monitoring.
- APIs for user confirmation.

User Roles

We will employ role-based access control and rights will be granted to users according to their roles. This approach will facilitate better administration over authorisation.

User	Role Responsibilities	Role Based Access Type
Administrator	The back-end staff. Responsible for system features, control user accounts and review system logs. Limited access to the database following the principle of least privileges. Requires strong authorizations and authentication.	**Create:** -officers' accounts -new databases and system upgrading **Read:** -system logs (system access instances) -officers' accounts **Update:** -officers' accounts in particular officers' access authorities **Delete:** -user accounts
Officer	Responsible for security reports submitted by the public and investigation. Reads and updates report status and sends feedback and security measures to the reporting entity and input investigation notes. Requires strong authorizations and authentication.	**Create:** -security measures feedback to the public -case entries **Read:** -public reports, feedback, status and investigation notes **Update:** -Case status, security reports and investigation notes -public user information (require data protection officer (DPO) approval under GDPR) **Delete:** -Case records, status, security reports and investigation notes -public users' information (with DPO approval)

User	**Role Responsibilities**	**Role Based Access Type**
Public	Report digital security concerns and view officer's feedback. Their personal data is required for case follow-up purposes. Requires authentication.	**Create:** -Case entries and security reports **Read:** Views OWN data, report, feedback and case status

Application Architecture

- **Design Patterns:** Abstract factory to create several instances of classes and support main ability and reusability (Gamma & Helm, 2000).
- **Architectural Patterns:** Django's MTV architectural pattern will be used to handle HTTP requests and the front-end components of the application. The user interface shall also be provided.
- **APIs:** REST and SOAP web services to establish communications between the client and server. Implementation of CRUD.
- **Encryption:** Employing AES symmetric algorithm in our data encryption due to its key length options. Our system is open to the public and expects a high volume of entries, AES is preferred as it is less resource consuming as compared to RAS.

<u>Secure Software Design</u>

We will employ the waterfall model of software development. In the planning and design stages, STRIDE model is used, as well as recommendations of OWASP 2021 top 10s, to perform risk analysis with possible solutions (see below). Testing will be performed in the implementation stage.

Security threat	**Consequences**	**Security Measures**
Spoofing (False identity)	Confidentiality: Release of sensitive information	Authentication
Web parameter tampering	Integrity: Modify application data	Encryption
	Confidentiality/ integrity: Insert of malicious codes	Input validation, security code review
Repudiation	Integrity: Deny of actions	Encryption
Information Disclosure	Confidentiality: Unintentionally reveals sensitive information	Use generic error messages
Denial of Service (DoS)	Availability: Delay service	Log monitoring - identify early sign of DoS
Elevation of Privilege	Confidentiality: Unintended users gain capabilities	Authorization

<u>OWASP Guidelines</u>

1. Authentication - strong password required (OWASP, 2021) to prevent brute force and two-factor authentication (one-time password suggested) to prevent risk of compromised passwords (Imperva, n.d.)

2. Authorisation – see “User Role” for role-based access controls

3. Input validation – an easy-to-use but not perfect measure against malicious inputs (Hurbism, 2010).

4. Security code review - selected as complement of input validation for its ease of implementation along the development stage.

5. Cryptography – another lawyer of protection – see “Application Architecture” above.

6. Monitoring - a detective control per 2021 OWASP top-10.

<u>GDPR compliance (GDPR, nd)</u>

1. <u>Data collection</u>
   Personal user data for identifying users or cases being investigated are collected under data minimisation principle (GDPR 5(1c)) and are handled along GDPR guidelines. Prior consent from users is required unless for public interests (GDPR 5 and 6) (further parental consent is required for minors (GDPR 8)).

2. <u>Data Recording</u>
   Only required data will be stored (GDPR 5(1c)) and shall be kept for as long as necessary (GDPR 5(1e) and 25).

3. <u>Data Retrieval</u>
   Rights of data subjects including data retrieval (GDPR 15), update (GDPR 16), delete/ withdraw consent/ restriction of processing (GDPR 17-18) and portability (GDPR 20) are observed.

4. <u>Data Maintenance</u>
   Data are maintained under security measures pursuant to GDPR 5(1f) (authorisation (GDPR 32(2)), authentication (GDPR 32(2)), encryption (GDPR 32(1a)), pseudonymisation (GDPR 32(1a)), backup (GDPR 32(2)) - limited to strategy).

All entries shall be validated and file uploads shall be scanned before being attached to records. The infrastructure will be built with the focus on protecting the application from intentional and unintentional threats.

Implementation Tools

Python 3.10, Django	Will be used as a framework for rapid and secure web application development.
SQLite	Database Management
Microsoft Azure, Unittest	External python library for function testing and deployment platform.
Other External Libraries	**OTP:** PyOTP, qrcode, Pillow, google authenticator **Encryption:** Cryptography **Input Validation:** PyInputPlus **Code review**: pylint, Bandit

Server / System Requirements:

1. Storage: SQLite 1 TB disk space
2. Access type: Local or remote access
3. Server processor: Intel Xeon 3.6 GHz, DDR4 8GB Ram

UML Diagrams

1. Use Case Diagram
   Appendix 1
2. Activity Diagram
   Appendix 2
3. Class Diagram
   Appendix 3

References

General Data Protection Regulation (GDPR). (n.d.) European Union. Available from: https://gdpr.eu/tag/gdr/ [Accessed 14 February 2023]

Hurbism. (2010) Better: Input Validation or Output Encoding? 31 May 2010. Available from: https://blogs.msmvps.com/alunj/2010/05/31/better-input-validation-or-output-encoding/#:~:text=So%2C%20that's%20where%20Output%20Encoding,told%20apart%20from%20each%20other%E2%80%9D. [Access on 6 Feb 2023]

OWASP. (2021) A07: 2021 –Identification and Authentication Failures/, available from https://owasp.org/Top10/A07_2021- Identification_and_Authentication_Failures// [Access on 6 Feb 2023]

Imperva. (n.d.) Two Factor Authentication (2FA). Available from: https://www.imperva.com/learn/application-security/2fa-two-factor-authentication/ [Accessed on 15 Feb 2023]

Godinho et al. “Torrent Poisoning Protection with a Reverse Proxy Server.” Electronics (Basel) 12.1 (2022): 165–. Web.

Stoianov et al. “Integrated Security Infrastructures for Law Enforcement Agencies.” Multimedia tools and applications 74.12 (2015): 4453–4468. Web.

Kancherla et al. “Epiviz File Server: Query, Transform and Interactively Explore Data from Indexed Genomic Files.” Bioinformatics 36.18 (2020): 4682–4690. Web.

KAMEI, H. ,TAKAKI, N. “A Method for High-Throughput Deduplication for Primary File Server by Using Prefetch Cache.” Electronics and communications in Japan 99.12 (2016): 54–64. Web.

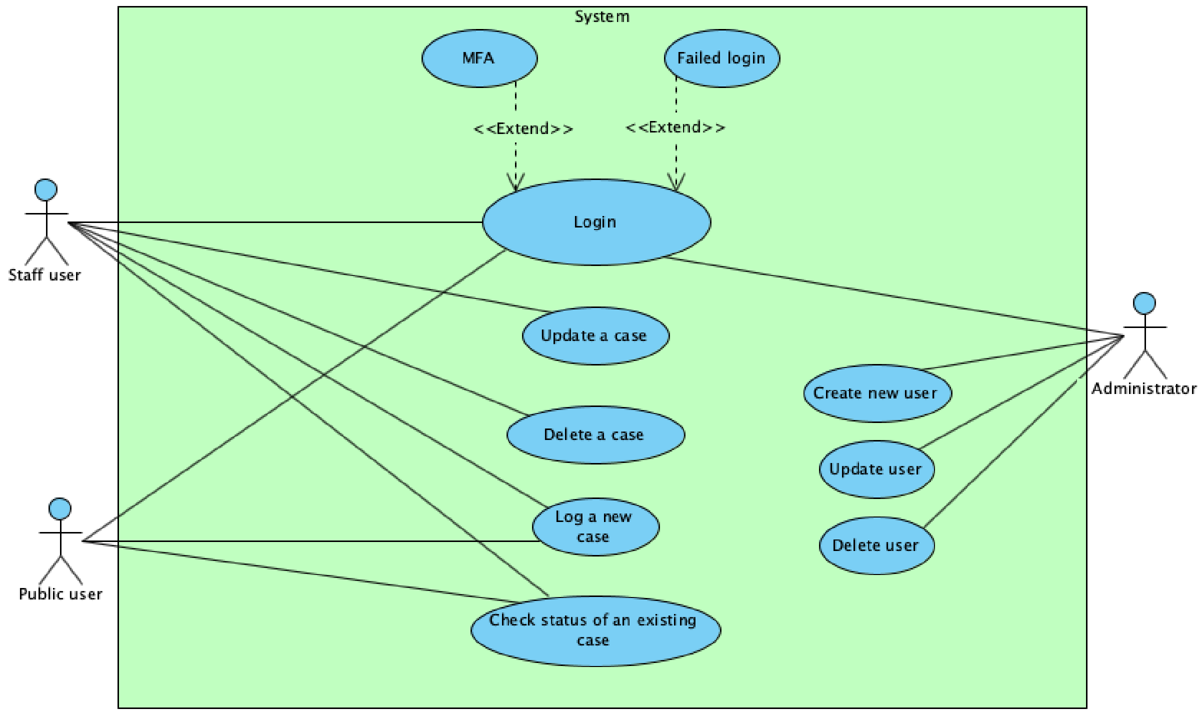
Davidson, L., Moss, J. Pro SQL Server Relational Database Design and Implementation. 5th ed. 2016. Berkeley, CA: Apress, 2016. Web. https://learning.oreilly.com/library/view/pro-sql-server/9781484219720/ [Accessed on 17 February 2023]

Gamma, E. & Helm, R., 2000. *Design Patterns: Abstraction and Reuse of Object-Oriented Design.* Verlag , ECOOP.

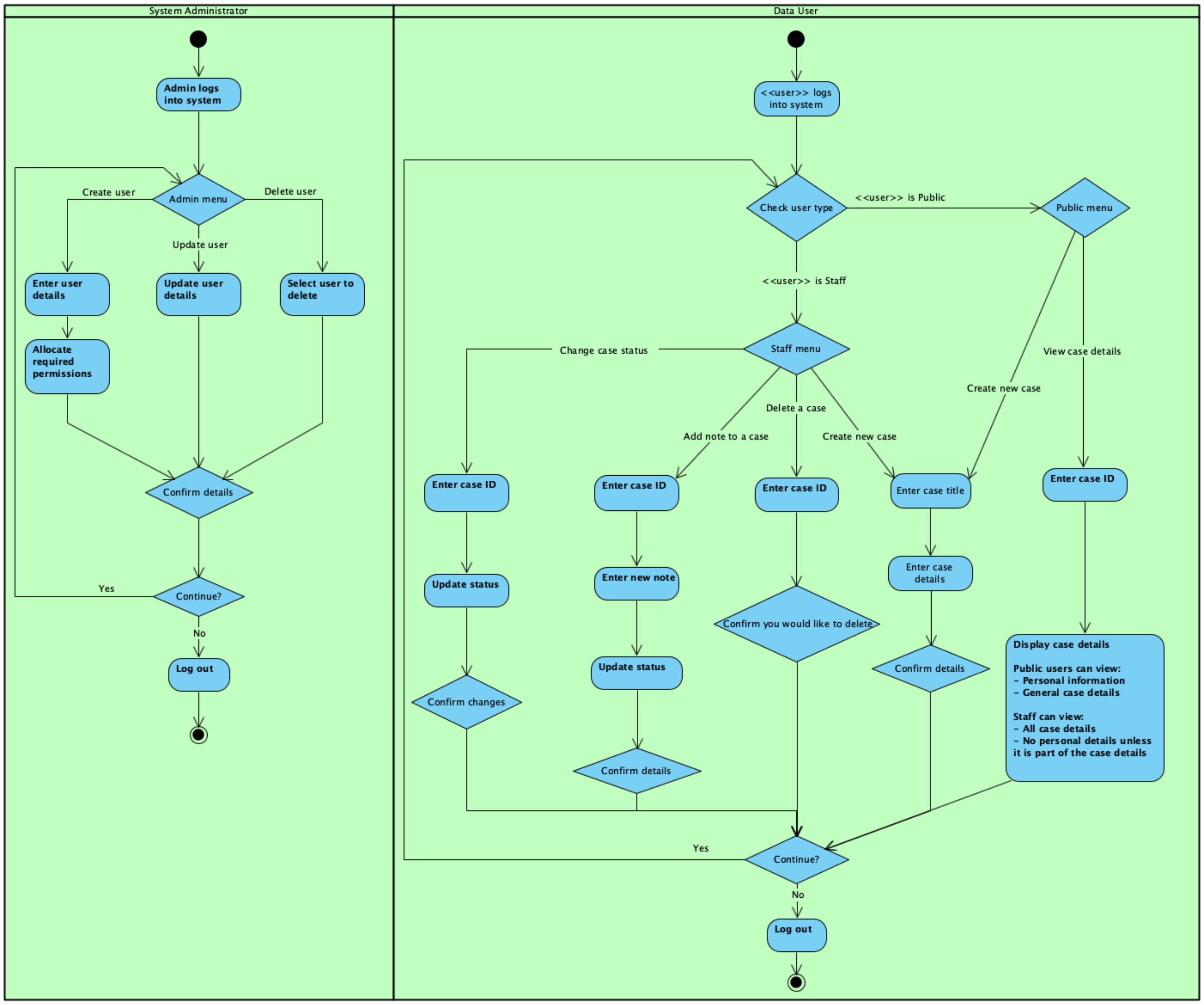
Kaska, K., 2015. *National Cyber Security Organization: the Netherlands,* Taillinn: CCDCOE.

Veale, M. & Ian, B., 2020. Cybersecurity. *Internet Policy Review,* 9(4), pp. 1-22.

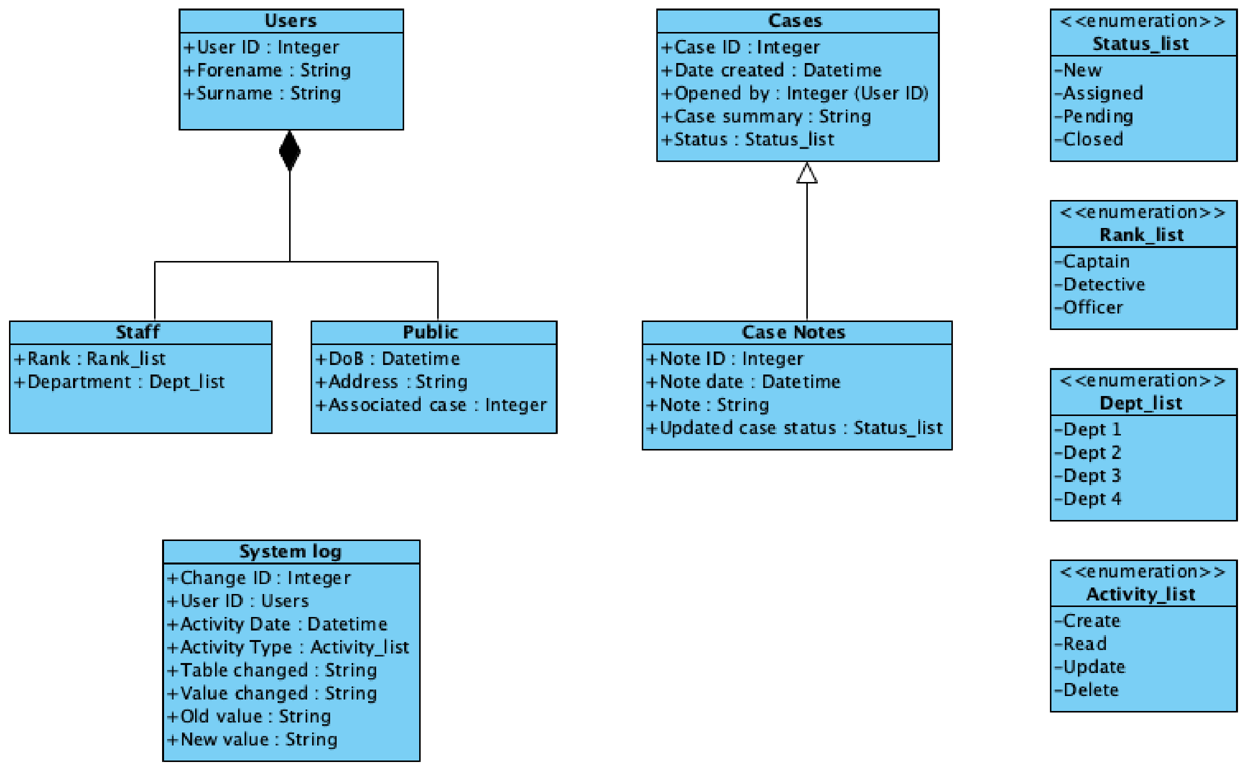
Appendix 1: Use case Diagram: System Features & Interactions



Appendix 2: Activity Diagram : Staff & Public Login Process



Appendix 3: Class diagram


Users
+User ID : Integer
+Forename : String
+Surname : String
Cases
+Case ID : Integer
+Date created : Datetime
+Opened by : Integer (User ID)
+Case summary : String
+Status : Status_list
<<enumeration>>
Status_list
-New
-Assigned
-Pending
-Closed
<<enumeration>>
Rank_list
-Captain
-Detective
-Officer
Staff
+Rank : Rank_list
+Department : Dept_list
Public
+DoB : Datetime
+Address : String
+Associated case : Integer
Case Notes
+Note ID : Integer
+Note date : Datetime
+Note : String
+Updated case status : Status_list
<<enumeration>>
Dept_list
-Dept 1
-Dept 2
-Dept 3
-Dept 4
System log
+Change ID : Integer
+User ID : Users
+Activity Date : Datetime
+Activity Type : Activity_list
+Table changed : String
+Value changed : String
+Old value : String
+New value : String
<<enumeration>>
Activity_list
-Create
-Read
-Update
-Delete