1. Report Objective

This report's objective is to detail a comprehensive design for the Dutch Police Internet Forensics domain.

2. Privacy and Security Requirements

- The processing of data will fall under the Directive 2016/680 of the European Parliament, which is referred to as the Police Directive, as well as GDPR, i.e., EU 2016/679 (Seyyar et al, 2020)
- The police directive protects an individual's personal data, which is being used for the detection or persecution of criminal activities (Seyyar et al, 2020):
- The internet is global and has no boundaries, therefore surveillance and monitoring can span international borders, jurisdictions and legal systems (Dermott et al, 2019)

3. Assumptions

Infrastructure will be physically based in the Netherlands to avoid jurisdiction issues.

4. System Design Overview

The system's design will incorporate:

- Web architecture due to its accessibility through the internet
- Cloud hosting
- Key components of the architecture will be:
  - Hadoop
  - HDFS
  - MapReduce
  - Apache Cassandra
  - HBase
  - Elastic Search
  - Kafka
  - A central service which processes all requests such as authentication, authorization, data uploads, queries, and content retrieval and sends all messages to the logging service (Seyyar et al, 2020).
- The Model-View-Controller (MVC) architectural pattern is proposed because it:
  - Separates the business logic and presentation layer
  - Supports encapsulation by dividing the system into 3
  - Self-contains classes and objects

- No single point-of-failure, so the architecture is based on distributed storage and processing (Beek et al, 2015)

5. How the Design will meet the System Requirements and Technical Challenges

Security is a top priority due to the sensitivity of the data being processed, thus supporting confidentiality, integrity and availability (Beek et al, 2015).

If there is an issue with confidentiality and integrity, then this could result in privacy breaches and leaks of sensitive data (Beek etal, 2015), regulatory fines and reputational damage.

The system needs to be designed to support non-reputability to stop users from denying events (Pillai, 2017)

Confidentiality

System Requirements	Design Proposal to Meet the System Requirements & Technical Challenges (including hardware design)
• Privacy functionality needs to be built into the system (Seyyar et al, 2020) • Personal data needs to be destroyed according to retention and destroy policies (Seyyar et al, 2020) • Datasets for testing and training should be anonymized (Seyyar et al, 2020) • Article 6 of the police directive states that anonymization should be applied to data for different types of personal data, (Seyyar et al, 2020) • There is a regulatory requirement for data to be held for the shortest time possible. • There needs to be user roles, which restrict access to data and functions Controls around user registration, including the onboarding of new users and removing access for leavers • Auditing and Logging -Audit trails of who is searching for what and controls around searches. Availability of audit trails to authorised users Multi factor authentication **Technical Challenges :** Unauthorised sharing of information (Beek et al, 2015)	• HTTPS -Traffic between the SQL database and the web application will use HTTPS for communication • Use of SSL certificate to encrypt traffic between the web server and the client browser. • Encryption of all data (in transit and when persisted) • Encryption keys stored in a different location/domain from the encrypted data (Beek et al, 2015) • Availability of data to certain users based on roles and permissions (Beek et al, 2015) • Multi-factor Authentication -To logon, a valid password should be entered (account is locked after the 3rd attempt), then an OTP is required (should be entered in 30 seconds). • Firewall -use of web application firewall to control traffic and access to the web application and the database. • All requests to a central service are persisted as logs and the log files are backed-up offline (Beek et al, 2015) • Log files to be encrypted and contain authentication and authorization requests. • Logging system can be queried and reports can be generated only by an authorised user. • Log files should not contain private information (Beek et al, 2015) • Kerberos is used between hosts (Beek et al, 2015) • Monitoring Tool to monitor activities, traffic, CPU and memory usage. • Data level authentication • Authorization -Identity and user management, where users obtain access rights based on their role (Seyyar et al, 2020) • Anonymized Data - Privacy data in log files is anonymized and can be reversed using cryptographic keys. • Functionality to tag an object or data as being 'confidential' so as to restrict unauthorised access (Seyyar et al, 2020) • Segregated Network and role-based access control system to restrict network access

Integrity

System Requirements	Design Proposal to Meet the System Requirements and Technical Challenges (including hardware design to support the requirements)	Hardware Proposal to meet the CIA and System Requirements
• Raw Data cannot be Modified (Dermott et al, 2019) • Existence of only valid data values • Audit Trail and Logging -Legal audit trail which proves that the data has not been altered in anyway • **Technical Challenges** • Due to different data types and images (Dermott et al, 2019) Data should not be altered in transit or when persisted.	• Input Validation of all attributes • Data Replication -HDFS provides replication three times • Encrypt data when in transit and when persisted • A copy of the raw data will be persisted in WORM storage prior to processing • Different tools, in the form of libraries will be used to process unstructured data	• WORM storage, which supports a retention date, will be used so data cannot be updated or deleted until the retention period is reached.

Availability

System Requirements	Design Proposal to Meet the System Requirements and Technical Challenges (including hardware design to support the requirements)
• Performance -a fast response (within 3 seconds for a query) • Remote Access -Ability to use the system on IoT devices • Patching -A maintenance window of 8 hours for patching (preferably Sunday -1am to 8am) • Data Processing -Ability to process 3-terabytes of data per hour, which is the requirement for another big data forensic platform called Hansken, which is similar in terms of system requirements (Seyyar et al, 2020) • **Technical Challenges** • Has to process big data both structured and unstructured from different sources, looking for patterns and correlation • Searching across vast amounts of data, including multi-media and voice, and chat messages, emails, photo's etc. (Seyyar et al, 2020). • The system's ability to process petabytes of data within the processing SLA's	• Centralization of data, software, processing and storage -so as to provide faster forensic analysis (Seyyar et al, 2020) • Decentralised architecture and scalable. • Caching plug-ins will be used to increase the speed of the website • The proposed architecture (of Hadoop, HDFS, etc) provides the ability to process vast amounts of data in an acceptable timeframe. • The Web server will be clustered for high-availability and also act as a fail-over. • Data and File compression will be used to better manage storage capacity requirements • Regular and automated backups will be conducted in case of a disaster recovery and business continuity incident • A load-balancer will be used that will act as a reverse proxy, distribute traffic across servers in order to increase the number of concurrent users' capacity and to improve the reliability of the system • Fault Detection -Monitoring heartbeats and ping/echo messages sent to nodes (Pillai, 2017) • Fault Recovery -Using techniques like: rollback and retry. • The fully implemented system will need 1,000 CPU's with more than five terabytes of combined RAM processing in parallel • The database will be sized to support 10 million artefacts, which will need to be fully indexed and associated with meta data, so that data will not need re-processing • Storage needs to be in terms of petabytes (Beek et al, 2015) • Storage (SAN) will be dynamically added as the system grows due to cloud hosting

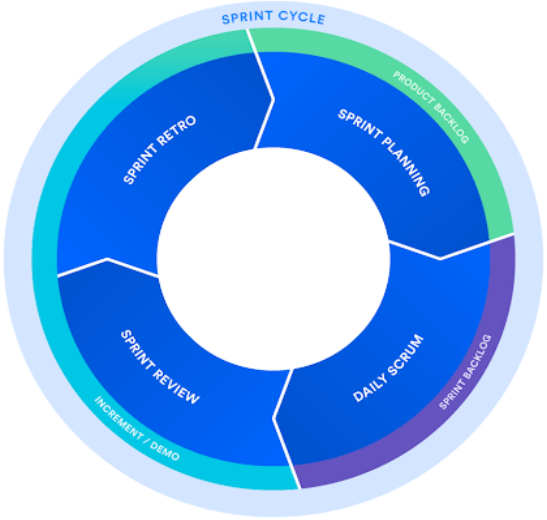
6. Threat Modelling

The current top 10 web application threats from OWASP (OWASP, 2022) were analysed, and the proposed mitigations were incorporated into the design (See Appendix B).

In addition, STRIDE threats were analysed and the design will also include the threat mitigations listed (See Appendix C) (Microsoft, 2003).

7. SDLC Methodology and Approach

- Scrum agile is the proposed methodology, because of the following benefits :
  - Transparency
  - Flexibility
  - The project is divided into smaller sprints and so there is continuous delivery
  - Regular user feedback as part of each sprint
  - Supports changing requirements throughout the development process as compared to the waterfall methodology



Scrum Agile Framework from https://www.atlassian.com/agile/scrum

8. Proof of Concept – Sprint One

- A proof of concept (POC) will be the first phase of the project, and will be divided into multiple sprints, the sprints will be prioritized according to the highest rated threats.
- Design decisions will also be reviewed during the POC, which makes agile a good approach for an evolving system
- Included will be code reviews, automated testing, in terms of unit, integration and regression testing (Beek et al, 2015)

- The UML artefacts scope is limited to the first sprint of the POC

9. Proof of Concept – Sprint One Scope

Sprint One will be a 4 week sprint, and will focus on the current top OWASP threats (See Appendix). The sprint will include a mitigation from each of the first one to seven OWASP categories. The below threat mitigation categories were also identified as part of STRIDE (See Appendix), apart from the implementation of the MVC pattern :

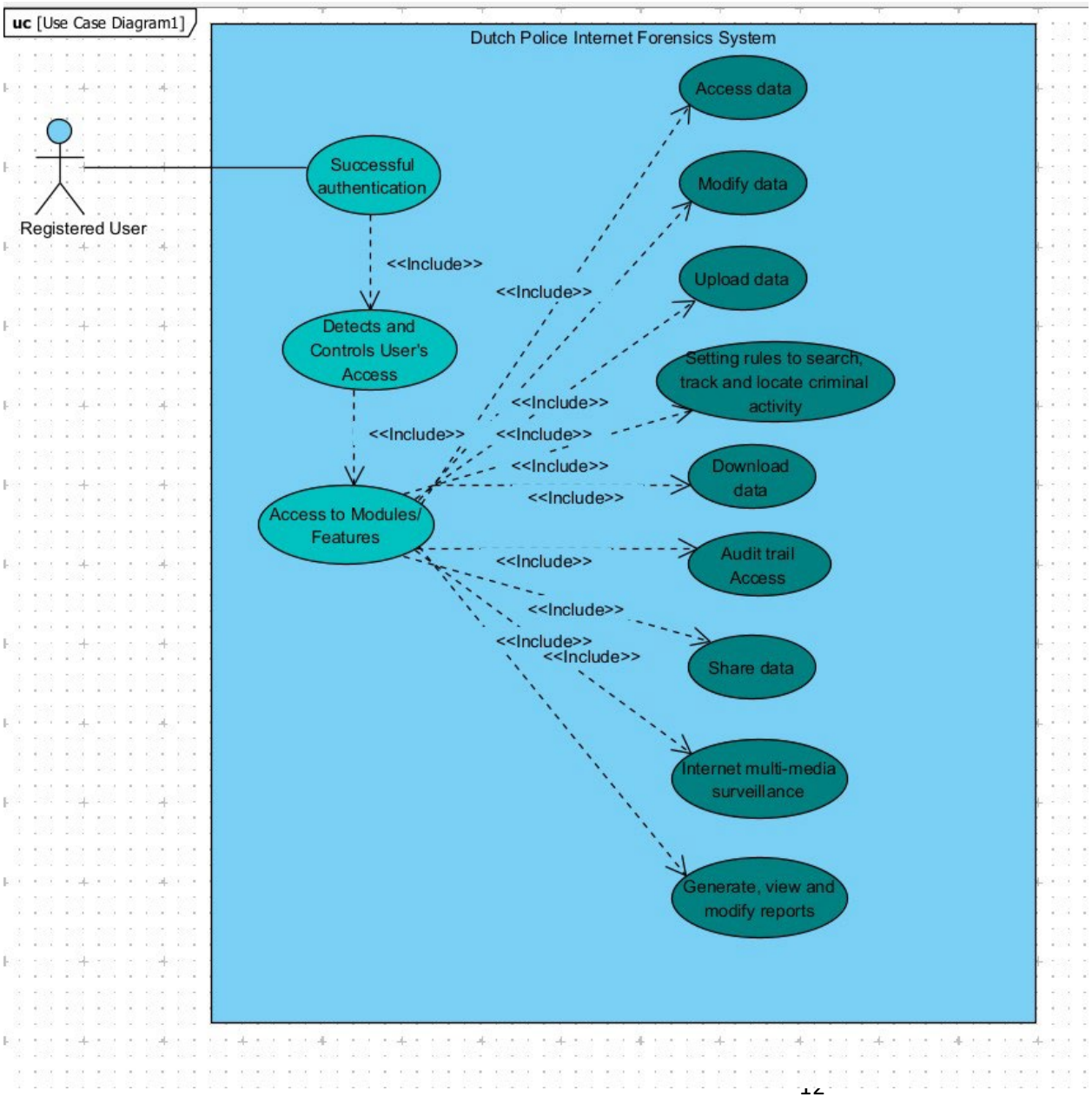
Threat Mitigation Category	Proposed Threat Mitigation Changes	OWASP	STRIDE
Authorization	• User permissions and roles • An administrator UI where a locked user can be unlocked, and where new users can be added and leavers de-activated	Yes	Yes
Multi-factor Authentication	• Password standards • OTP • Three attempts to logon	Yes	Yes
Input Validation	• Logon UI which includes input validation • Server-side input validation	Yes	Yes
Encryption	• Encryption of data in transit and at rest	Yes	Yes
Security Misconfiguration	• Implementation of the MVC architecture pattern	Yes	No

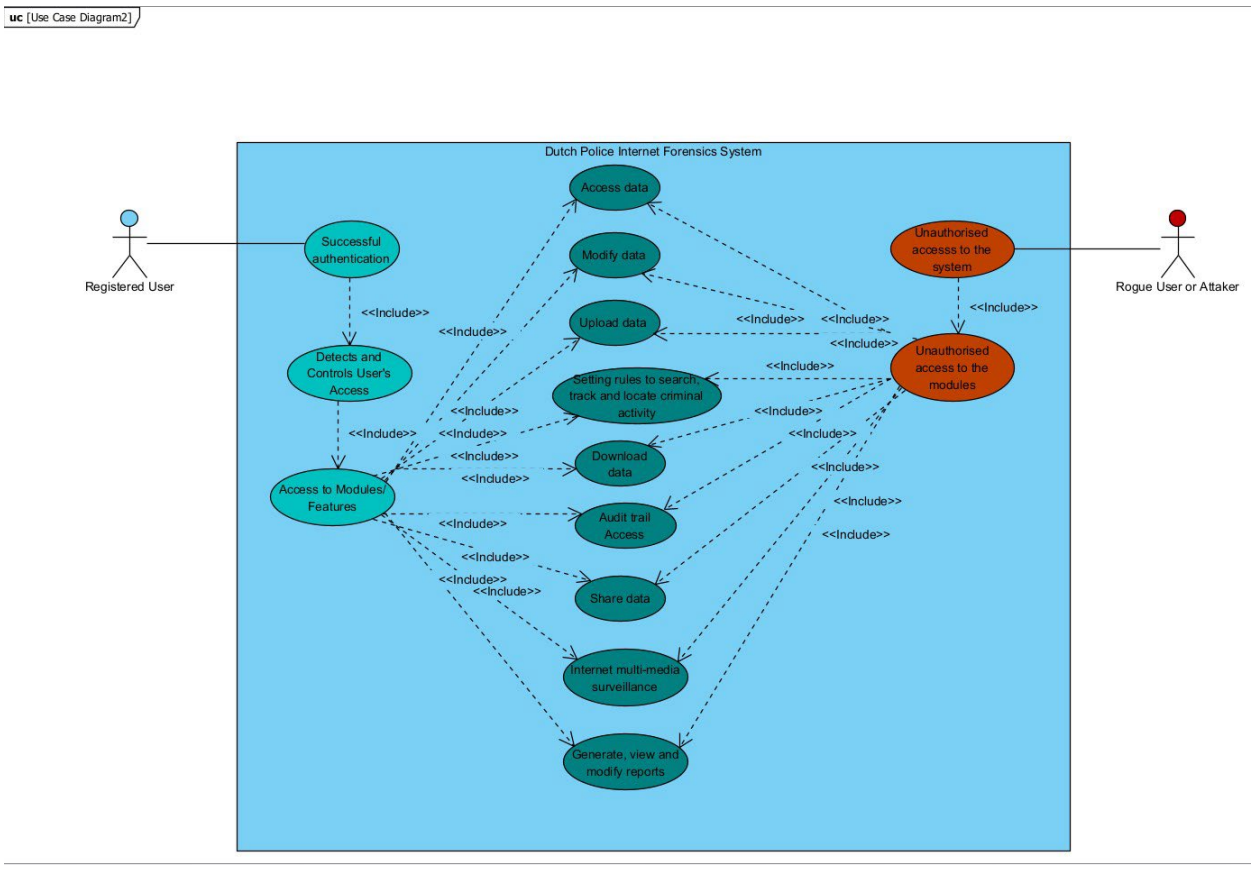
10. The proposed tools, models and libraries for the proof of concept are :

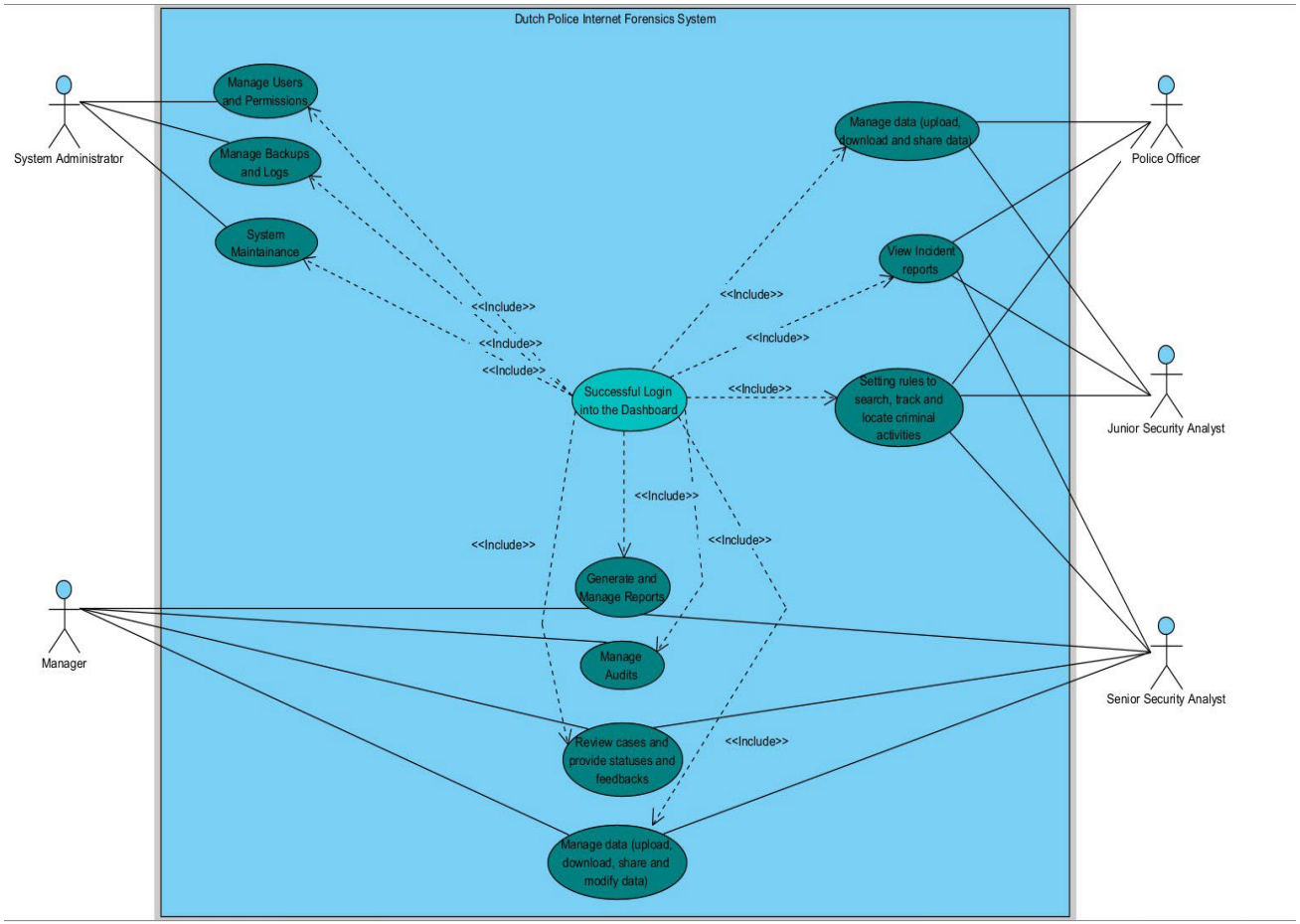
Development Environment	Development tools, libraries, models, and languages
Development tools	IDE – Codio
	GitHub
	JIRA
Test Tools	Katalon studio
Libraries	Python encryption, OTP library, Requests library (See Appendix D)
Architectural Model	MVC Django
Front End Languages	HTML CSS Java

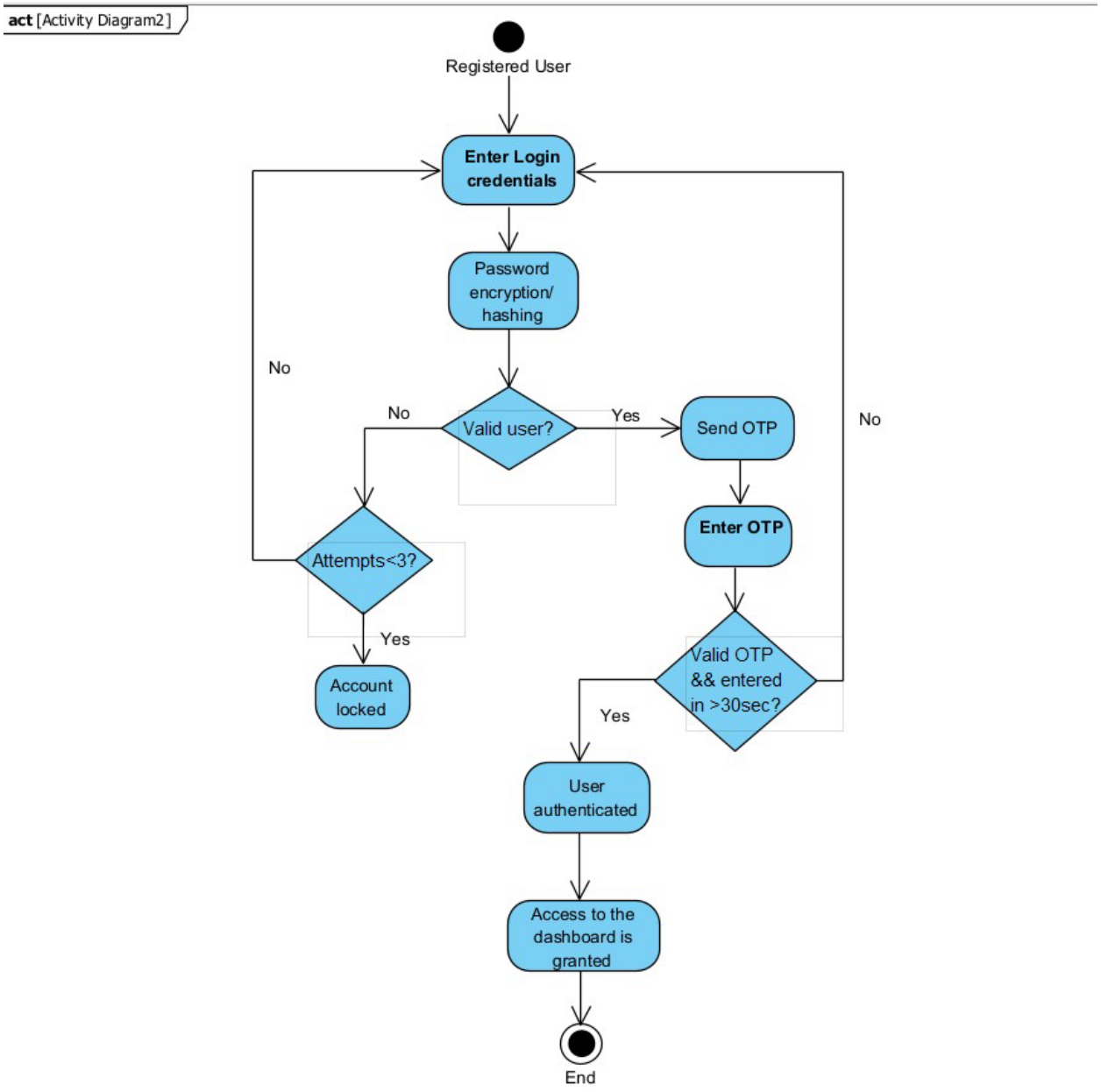
	Javascript
Back-end Languages	Python
Database	MySQL

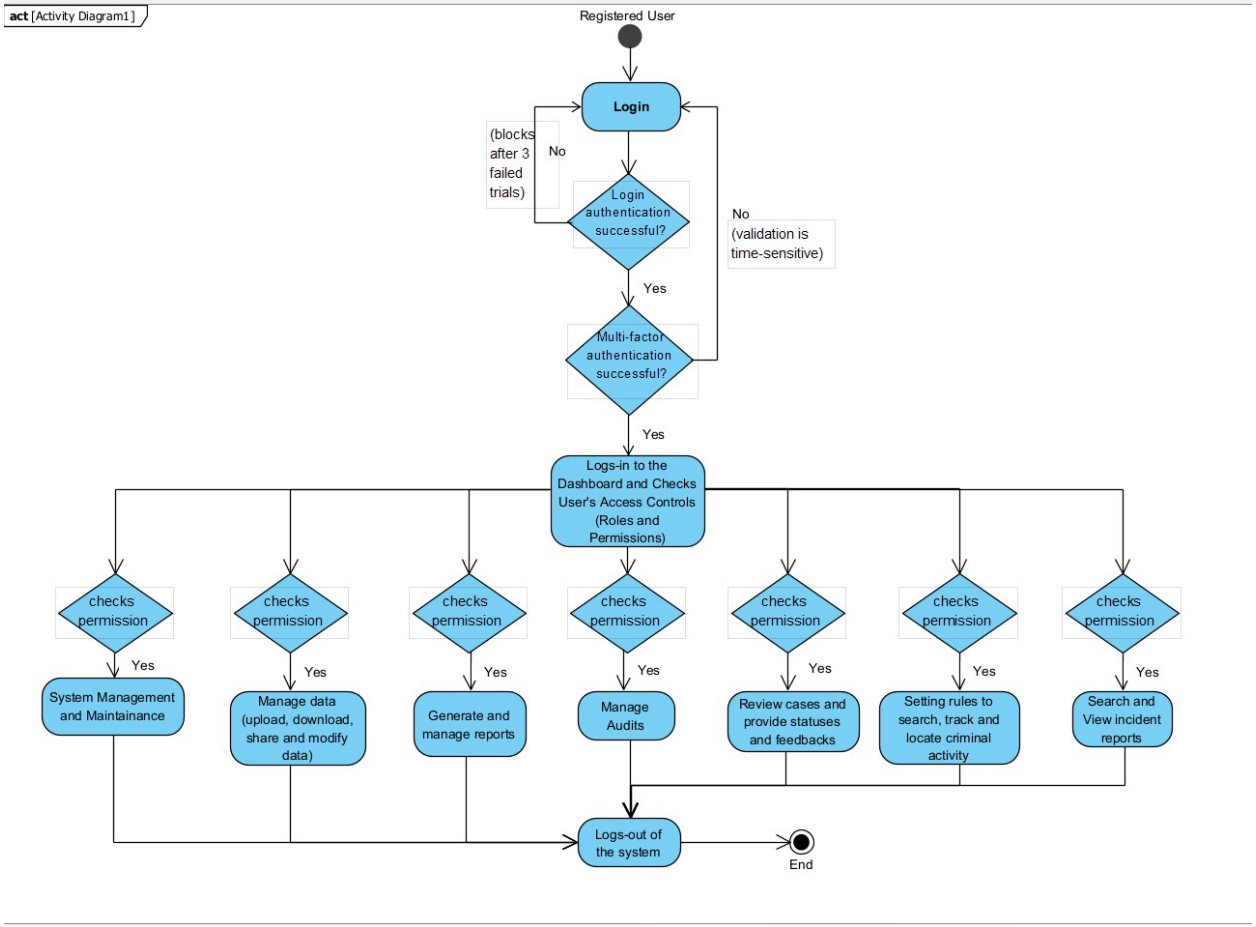
11. UML Artefacts for POC Sprint One

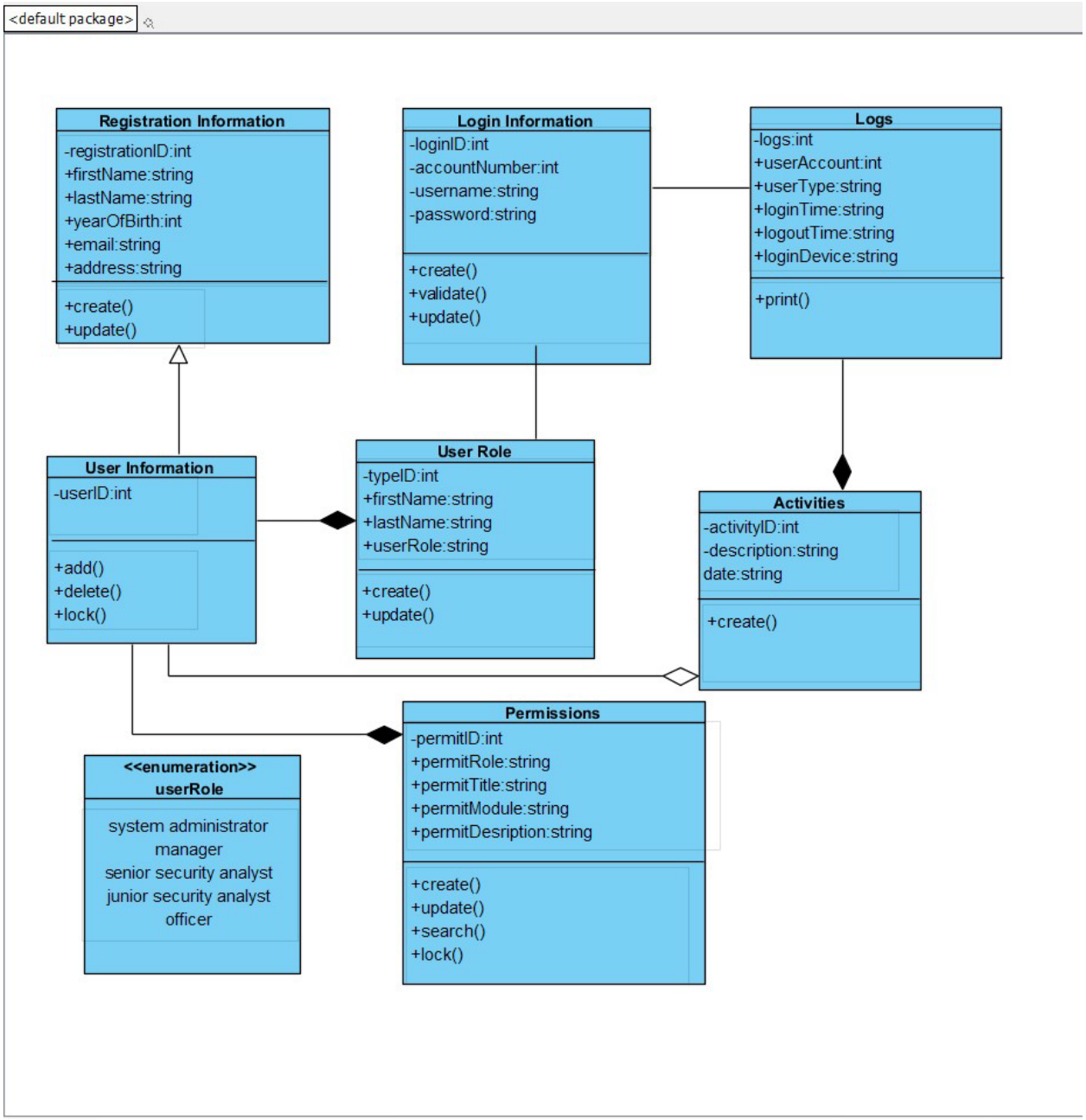

uc [Use Case Diagram1]
Dutch Police Internet Forensics System
Registered User
Successful authentication
Detects and Controls User's Access
Access to Modules/ Features
Access data
Modify data
Upload data
Setting rules to search, track and locate criminal activity
Download data
Audit trail Access
Share data
Internet multi-media surveillance
Generate, view and modify reports
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>

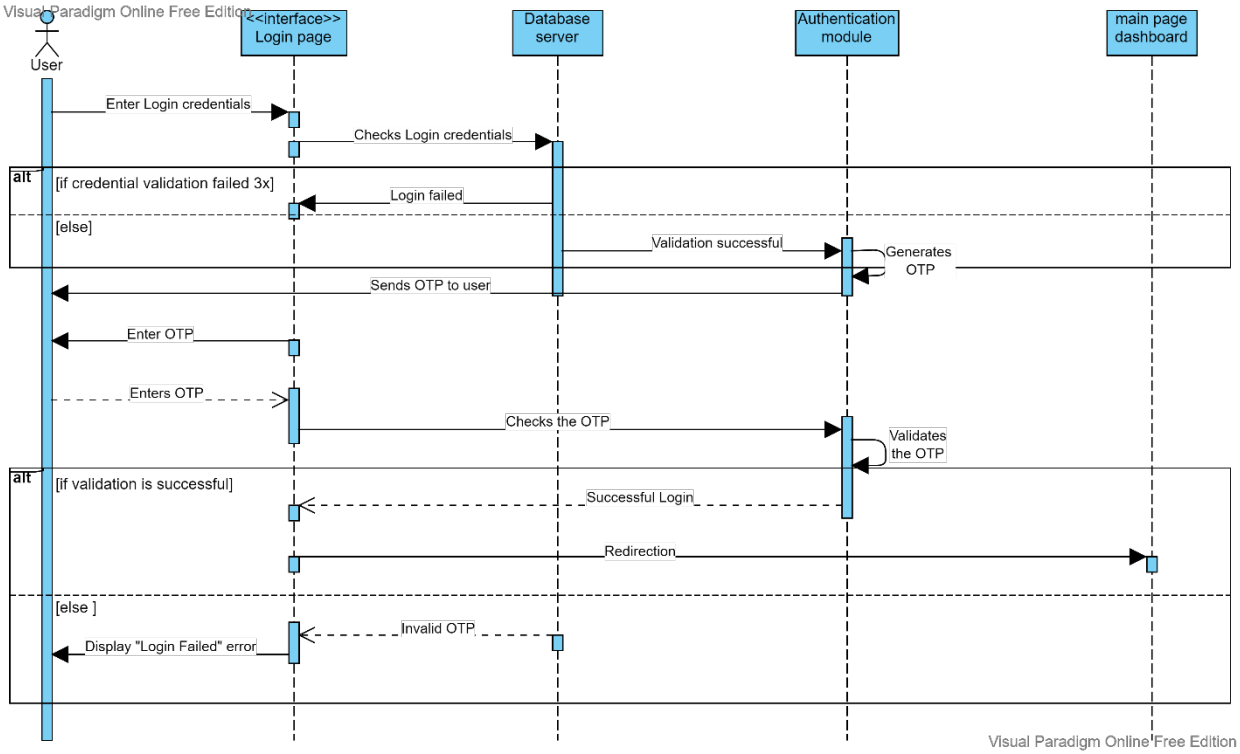

uc [Use Case Diagram2]
Dutch Police Internet Forensics System
Registered User
Rogue User or Attaker
Successful authentication
Unauthorised accesss to the system
Access data
Modify data
Upload data
Detects and Controls User's Access
Setting rules to search, track and locate criminal activity
Unauthorised access to the modules
Download data
Access to Modules/ Features
Audit trail Access
Share data
Internet multi-media surveillance
Generate, view and modify reports
<<Include>>


Dutch Police Internet Forensics System
System Administrator
Manage Users and Permissions
Manage Backups and Logs
System Maintainance
Manager
Police Officer
Junior Security Analyst
Senior Security Analyst
Manage data (upload, download and share data)
View Incident reports
Setting rules to search, track and locate criminal activities
Successful Login into the Dashboard
Generate and Manage Reports
Manage Audits
Review cases and provide statuses and feedbacks
Manage data (upload, download, share and modify data)
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>

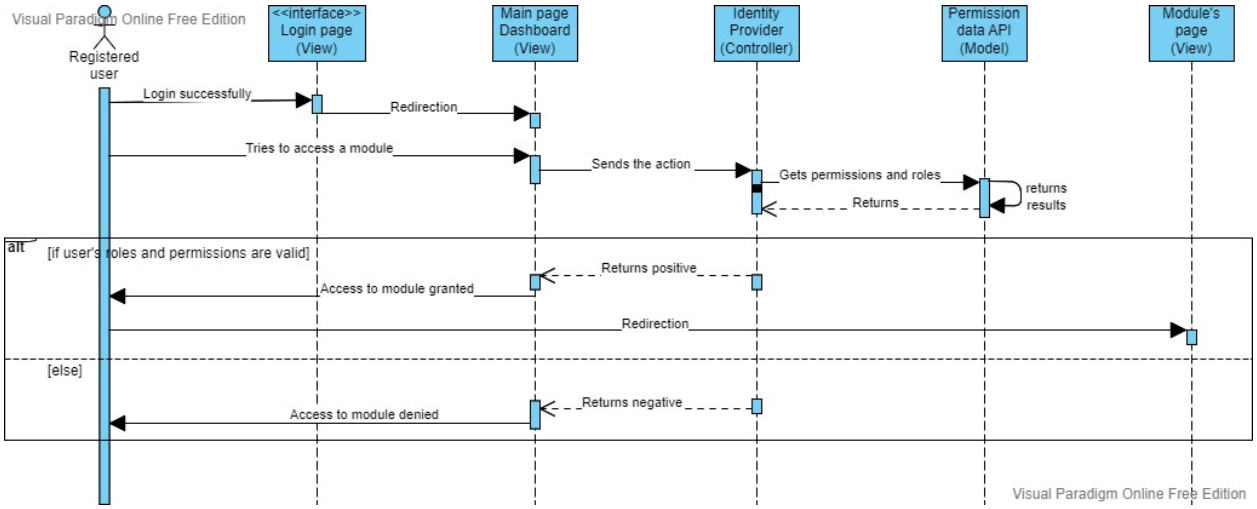

act [Activity Diagram2]
Registered User
Enter Login credentials
Password encryption/ hashing
Valid user?
No
Yes
No
Send OTP
Enter OTP
Attempts<3?
No
Yes
Account locked
Valid OTP && entered in >30sec?
No
Yes
User authenticated
Access to the dashboard is granted
End


act [Activity Diagram1]
Registered User
Login
(blocks after 3 failed trials)
No
Login authentication successful?
No
(validation is time-sensitive)
Yes
Multi-factor authentication successful?
Yes
Logs-in to the Dashboard and Checks User's Access Controls (Roles and Permissions)
checks permission
Yes
System Management and Maintainance
checks permission
Yes
Manage data (upload, download, share and modify data)
checks permission
Yes
Generate and manage reports
checks permission
Yes
Manage Audits
checks permission
Yes
Review cases and provide statuses and feedbacks
checks permission
Yes
Setting rules to search, track and locate criminal activity
checks permission
Yes
Search and View incident reports
Logs-out of the system
End


<default package>
Registration Information
-registrationID:int
+firstName:string
+lastName:string
+yearOfBirth:int
+email:string
+address:string
+create()
+update()
Login Information
-loginID:int
-accountNumber:int
-username:string
-password:string
+create()
+validate()
+update()
Logs
-logs:int
+userAccount:int
+userType:string
+loginTime:string
+logoutTime:string
+loginDevice:string
+print()
User Information
-userID:int
+add()
+delete()
+lock()
User Role
-typeID:int
+firstName:string
+lastName:string
+userRole:string
+create()
+update()
Activities
-activityID:int
-description:string
date:string
+create()
Permissions
-permitID:int
+permitRole:string
+permitTitle:string
+permitModule:string
+permitDesription:string
+create()
+update()
+search()
+lock()
<<enumeration>>
userRole
system administrator
manager
senior security analyst
junior security analyst
officer



Sequence diagram showing a user who is trying to access one of the modules after successfully logging in to the main page dashboard :


Visual Paradigm Online Free Edition
Registered user
<<interface>> Login page (View)
Main page Dashboard (View)
Identity Provider (Controller)
Permission data API (Model)
Module's page (View)
Login successfully
Redirection
Tries to access a module
Sends the action
Gets permissions and roles
returns results
Returns
alt
[if user's roles and permissions are valid]
Returns positive
Access to module granted
Redirection
[else]
Returns negative
Access to module denied
Visual Paradigm Online Free Edition

References

Beek, van H.M.A., Eijk, van E.J., Baar, van R.B., Ugen, M., Bodde, J.N.C. (2015). *Digital Forensics as a Service: Game on.* Digital Investigation.

Dauzon, S., Bendoraitis, A. & Ravindran, A., 2016. *Django: Web Development with Python.* Mumbai: Packt Publishing Ltd.

Dermott, Mac AM., Baker, T., Buck, P., Iqbal, F., Shi, Q. (2019). *The internet of things: challenges and considerations for cybercrime investigations and digital forensics.* International Journal of Digital Crime and Forensics.

Lenka, C., 2020. *Python program check validity Password.* [Online]
Available at: https://www.geeksforgeeks.org/python-program-check-validity-password/
[Accessed 28 May 2022].

Melé, A., 2020. *Django 3 by example.* 3rd ed. UK: Packt Publishing Ltd.

Oliphant, T. E., 2007. Python for Scientific Computing. *Computing in Science and Engineering,* 9(3), pp. 10-20.

OWASP (2022). OWASP Top Ten. Available from:
https://owasp.org/www-project-top-ten/
[Accessed 15th November 2022]

Pillai, A. (2017) *Software Architecture with Python.* 1st ed. Birmingham: Packt Publishing Ltd.

Microsoft. (2003) *Improving Web Application Security Threats and Countermeasures.* 1st ed.

Seyyar, Bas M., Geradts, Z.J.M.H. (2020). *Privacy impact in large scale digital forensic investigations.* Forensic Science International.

Appendix A – Business Requirement – High Level functionality

Dutch police internet forensics

- Users must be able to upload, download and share data
- Searching across the internet for specific criminal activity, searching for voice, emails, text, photo's etc.. so multi media surveillance
- Need a fast response – within 3 seconds
- Need to be able to logon to the system on their mobile devices, when away from the main office
- Need to be able to search for patterns and fuzzy logic
- Need to be able to identify the location of any data which is deemed to be of interest
- Need an audit trail of who is searching for what and controls around searches so that no one is looking for data outside of the project that they are not working on
- The audit trail should also be available for review by permissioned senior users
- Need to be able to create a case, and associate the data found with that case, and also their findings need to be persisted as well as any relevant reports.

- There also needs to be a workflow where a senior officer can review the case and details and provide feedback, so the status would be ‘not started’, ‘in progress’, ‘submitted for review’, ‘closed’ – and when submitted for review the status could then also revert back to ‘in progress’.
- There should be a visible audit trail of the status history
- Ability to produce reports showing trends, and also a summary for specific cases for senior management
- Controls must be in place so that no data which has been downloaded can be altered, because it may be used in court
- Automated surveillance across the internet, looking for criminal behaviour, which would then need to be investigated and a case created
- Needs to be a global system – in what location will the data be persisted ?
- Language translation for multiple languages
- User registration – onboarding new users and removing access for leavers
- User roles – what are the different user roles?
- Create, Read, Update and Delete (CRUD). Can details be deleted while the status is in ‘draft’ if deleted just mark as deleted, no physical deletion, could abuse the search functionality
- Ability to extend operations to partner organisations on a worldwide basis, therefore permissions and roles need to take into account what other organisations can view, create, update, amend and delete

Appendix B

Mitigations highlighted in blue will be included within the scope of the first sprint :

Id	OWASP Threat Category	Proposed design mitigations for the internet surveillance system based on OWASP threats and mitigations
1	Broken Access Control	• Deny by default. • Implement access control mechanisms • Access controls should be granular on the basis of create, read, update, or delete records. • Disable web server directory listing and ensure file metadata and backup files are not present within web roots. • Log access control failures, and alert admins • Limit API and controller access • Stateful session identifiers should be invalidated on the server after logout. • A privilege management system
2	Cryptographic Failures	• Identify which data is sensitive • Don't store sensitive data unnecessarily.

Id	OWASP Threat Category	Proposed design mitigations for the internet surveillance system based on OWASP threats and mitigations
		• Encrypt all sensitive data at rest and in transit. • Ensure up-to-date and strong standard algorithms, protocols, and keys are in place • Encrypt all data in transit with secure protocols such as TLS • Enforce encryption using directives like HTTP Strict Transport Security (HSTS). • Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor) • Enforce passwords to be changed on a regular basis • Always use authenticated encryption. • Keys should be generated cryptographically randomly and stored in memory as byte arrays.
3	Injection	• API's, which provide a parameterized interface. • Server-side input validation • Use LIMIT and other SQL controls within queries to prevent mass disclosure of records in case of SQL injection.
4	Insecure design	• Establish and use a library of secure design patterns • Write unit and integration tests to validate that all critical flows are resistant to the threat model.

Id	OWASP Threat Category	Proposed design mitigations for the internet surveillance system based on OWASP threats and mitigations
5	Security Misconfiguration	• Development, QA, and production environments should all be configured identically, with different credentials used in each environment. • An automated task to review and update system configurations as part of the patching process • The application architecture will be segmented which provides secure separation between components (MVC)
6	Vulnerable and Outdated Components	• Continuously monitor sources like Common Vulnerability and Exposures (CVE) and National Vulnerability Database (NVD) for vulnerabilities in the components. • Only obtain components from official sources over secure links. • Monitor for libraries and components that are unmaintained
7	Identification and Authorization Failures	• Implement multi-factor authentication • Implement weak password checks • Align password length, complexity, and rotation policies with National Institute of Standards and Technology (NIST) 800-63b's guidelines in section 5.1.1 • Limit failed login attempts. • Log all login failures and alert administrators. • Use a server-side, secure, built-in session manager that generates a new random session ID after login.

Id	OWASP Threat Category	Proposed design mitigations for the internet surveillance system based on OWASP threats and mitigations
		• If no activity for a period of time, log the user out
8	Software and Data Integrity Failures	• Verify that software or data is from the expected source • Use OWASP Dependency Check to verify that components do not contain known vulnerabilities • Review process for code and configuration changes
9	Security Logging and Monitoring	• Maintain logs for firewalls, workstation, servers, any asset which shows inbound and outbound traffic, which includes monitoring and logging capabilities for endpoints and network infrastructure • Errors and warning should generate clear log messages • Logs should be monitored for suspicious activity • Logs should be backed up and stored remotely off line, as a second line of defence • Real time log monitoring process should be in place which provides alerts if suspicious activity is detected • Implementing software that monitors the network for suspicious activity, and endpoint detection tools and patch management tools
10	Server Side Request Forgery	• Segment the network and network traffic • Enforce “deny by default” firewall policies or network access control rules to block all but essential intranet traffic

Id	OWASP Threat Category	Proposed design mitigations for the internet surveillance system based on OWASP threats and mitigations
		• Validate all client input data • Enforce the URL schema, port, and destination with a positive allow list • Disable HTTP redirections

Appendix C

The mitigations highlighted in blue will be included in the first sprint :

Threat	STRIDE Countermeasures
Spoofing user identity	Strong authentication Do not store data in plaintext, such as passwords Do not pass credentials in plaintext Protect authentication cookies with Secure Sockets Layer (SSL)
Tampering with data	Use strong authorization Use data hashing and signing Use digital signatures Use tamper resistant protocols across communication links Secure communication links with protocols that provide message integrity
Repudiation	Create Secure audit trails Use digital signatures

Information Disclosure	Use Strong authorization Use Strong encryption Secure communication links with protocols that provide message confidentiality Do not store secrets in plain text
Denial of Service	Use resource and bandwidth throttling techniques Validate and filter input
Elevation of Privileges	Principle of least privilege Use least privileged service accounts to run processes and access resources

Appendix D – Library list

Custom-designed Input Validating Code

Regular Expressions (RegEx or re) library (to enable the use of 're.search()' method to search and detect a series of strings within a given criteria (Lenka, 2020).

One-Time Pin/Password (OTP) Generator and Password Generator for MFA

Importation of Random library (to allow generation of random numbers) and also Math library (to give access to common mathematical functions) (Oliphant, 2007).

Django forms library: it joins 3 main components of the framework: database fields, html form tags and ability of the user's input to be validated and display error messages (Dauzon, et al., 2016).

Python library: to interface with SQLite (Dauzon, et al., 2016).

Javascript library: allows a faster creation of client-side functionality (Dauzon, et al., 2016)**.**

Django-allauth: used for authentication (local and social).

Requests library: allows HTTP basic authentication (Melé, 2020)

Python Cryptographic library