Design Proposal

Background

Cyber security aims to protect the confidentiality, integrity, and availability of IT systems (Veale & Brown, 2020). New vulnerabilities continuously emerge as applications evolve and systems change therefore cybersecurity is a never-ending battle (Lin et al., 2007). The National Centre for Cyber Security (NCSC) in the Netherlands was established in 2012 to coordinate national cyber security policies (Boeke, 2017). The NCSC provides a channel for reporting vulnerabilities in ICT systems belonging to government bodies in the Netherlands. This is called a responsible disclosure and alerts the NCSC allowing them to remedy the flaw, ideally within 60 days, before making it public. (Government of the Netherlands, N.D.).

This application will provide an interface for the general public to submit vulnerabilities they have discovered to be remedied by the appropriate government body.

User roles

- General Public:
    - No account required.
    - Submit vulnerabilities.
    - View fixed vulnerabilities.
- Admin:
    - Account required.
    - Create and manage user accounts.
- Operator
    - Account required.
    - Verify/escalate submitted vulnerabilities.

Software architecture

Front-end:

- Allow users to interact with website user interface.
- Create a fast and responsive user interface.
- Dynamic checking for user input.

Backend:

- Create a REST API for the front end to send requests through.
- Allow third party systems to send requests using the API.
- Check the IP address of the user by login.
- Log events in the system for debugibility and security reasons.
- Sanitize user input of the vulnerability.
- Send confirmation email after submitting a vulnerability.

Application description

The general public can submit vulnerabilities with anonymous data entry or data entry to receive updates for their report. This feature is the main functionality demonstrated by the first activity diagram “Vulnerability submission by a general public user”. Furthermore, they can control their personal data with the ability to request the deletion of data through the application as per the GDPR directive (ICO, 2022), which is represented by the use case diagram from the perspective of the general public actor. The application will allow an operator to log in to verify, approve and reassign reported vulnerability records which is depicted in the second activity diagram “Operator Features”. Fixed and published vulnerability records can be viewed by anyone without a user account. A further user type called admin is responsible for account management duties. To demonstrate the above features, the actors in the use case diagram have been divided into operator, admin and the general public. In addition, the role of the app as a secondary actor was made clear. Finally, the class diagram represents the application's core with the required classes, subclasses and associations.

Design decisions

Design patterns:

- Factory method:
    - Creational design pattern.
    - Provides an interface for creating objects in a superclass while enabling subclasses to change the object type (Refactoring Guru, 2022).

Architecture patterns:

- Model-Template-View (MTV) as architecture design pattern:
    - Slightly different to the Model-View-Controller (MVC).
    - Model is responsible for the interaction with the database and the logical structure.
    - Template provides the user interface.
    - View handles the HTTP requests (Djangoproject, 2022).
- REST API:
    - Client-server communication for web application over HTTPS.
    - Implement GET, POST, PUT, and DELETE methods.
    - Main design principles are addressability, uniform interface and statelessness.
    - Works on the principle of CRUD (Create, Read, Update, Delete) (Prayogi et al., 2020: 2).

System requirements and assumptions

- Local or remote access.
- Storage required:
    - Disk space: 1 TB (it can scale up to 281 TB if needed) (SQLite, 2022).
- Initial server requirements:
    - CPU: Intel Ice Lake or AMD EPIC 4 Cores
    - RAM: DDR4 8GB

Software Quality:

- **Readability:** README file and comments are provided for clarity. PEP8 guideline is followed and a violation against coding standards is reviewed by Flake8.
- **Maintainability:** Through the Django system, updates can be made easily. Implement the use of the Git version control system.
- **Reusability:** Django provides methods and modules that are already created and ready to use. The principle of “Don’t Repeat Yourself” is also followed.
- **Modularity:** Django provides a framework that separates views, models, templates, urls, etc. from the start.
- **Usability:** The user interface is easy to use, as the user functions are limited only to submitting and viewing vulnerabilities.

Security challenges Identified

In order to avoid the common security risks to which the web applications are susceptible, the 2021 edition of OWASP Top Ten (N.D.) is taken as the basis. The mitigation of each security challenge identified can be seen below:

Security risk	**Mitigation**
Broken Access Control	Supported by a role matrix, proper authorization scheme will be prepared before the development so that each user type can only access the intended portion of the software.
Cryptographic Failures	Passwords will be hashed by BCRYPT instead of fast hashes like SHA1, SHA256 or SHA512.
Injection	Data sanitation will be handled by Django (N.D.)
Insecure Design	Database connection will not be hardcoded into the source code. Sensitive information will be added to

	config/env files and these files will never be committed to Github (will be added to gitignore). Github bot will be enabled to monitor unsafe commits.
Security Misconfiguration	Dynamic testing (such as Veracode Dynamic Analysis software) will be used.
Vulnerable and Outdated Components	A monthly security update will be scheduled.
Identification and Authentication Failures	Proper authentication and encryption of credentials will help avoid this type of failure.
Software and Data Integrity Failures	A database backup will be scheduled every weekend.
Security Logging and Monitoring Failures	An event logging system will be created to keep track of the logged in users and their IP addresses and device type. In case of an unknown login, an alert will be sent via email.

UML diagrams

1) Use case diagram

Appendix 1

2) Class Diagram

Appendix 2

3) Activity diagram - Vulnerability submission by a General public user

Appendix 3

4) Activity Diagram - Operator features

Appendix 4

Tools and Libraries

Programming Language	Python 3.10.8
Framework	Django 4.1.2, Bootstrap
Database	SQLite
Testing	Unittest
Linter	Flake8
Deployment platform	Microsoft Azure
Other tools/libraries	Bcrypt, HTML5, CSS

References

Boeke, S. (2017) National cyber crisis Management: Different European approaches. *Governance*, 31(3): 449-464.

Djangoproject (2022) Django appears to be a MVC framework, but you call the Controller the "view", and the View the "template". How come you don't use the standard names? Available from: https://docs.djangoproject.com/en/4.1/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names [Accessed 29 October 2022].

Django (N.D.) Form and field validation. Available from: https://docs.djangoproject.com/en/4.1/ref/forms/validation/ [Accessed 30 October 2022]

Government of the Netherlands (N.D.) Responsible disclosure. Available from: https://www.government.nl/topics/cybercrime/fighting-cybercrime-in-the-netherlands/responsible-disclosure [Accessed 23 October 2022].

I.C.O. (2022) Guide to General Data Protection Regulation. Available from: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/ [Accessed 29 October 2022].

Lin, H., Spector, A., Neumann, P. & Goodman, S. (2007) Toward a safer and more secure cyberspace. *Communications of the ACM* 50(10):128.

OWASP (N.D.) OWASP Top Ten. Available from: https://owasp.org/www-project-top-ten/ [Accessed 29 October 2022]

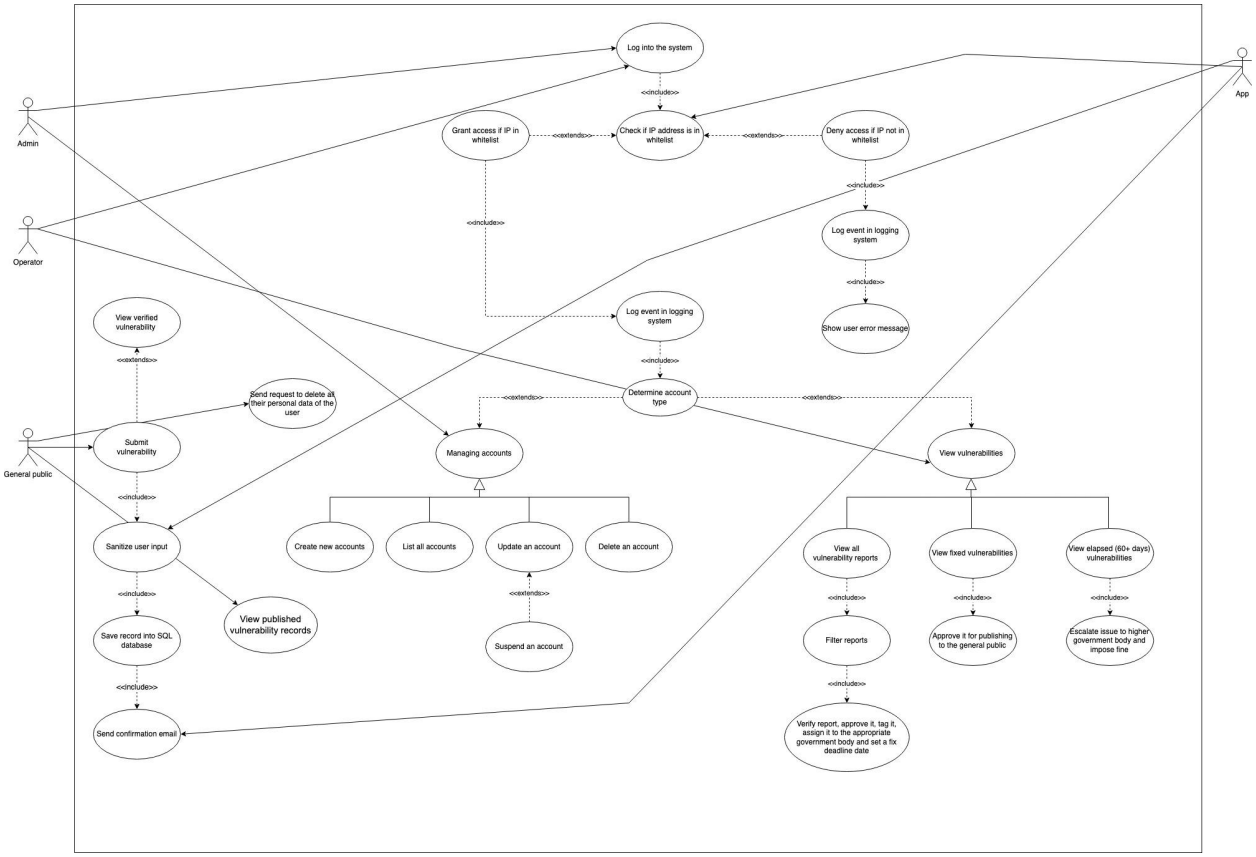
Prayogi, A A., Niswar, M., Indrabayu & Rijal, M. (2020) Design and Implementation of REST API for Academic Information System. I*OP Conference Series: Materials Science and Engineering* 875(1): 1-7. DOI: 10.1088/1757-899X/875/1/012047

Refactoring Guru (2022) Factory Method. Available from: https://refactoring.guru/design-patterns/factory-method [Accessed 29 October 2022].

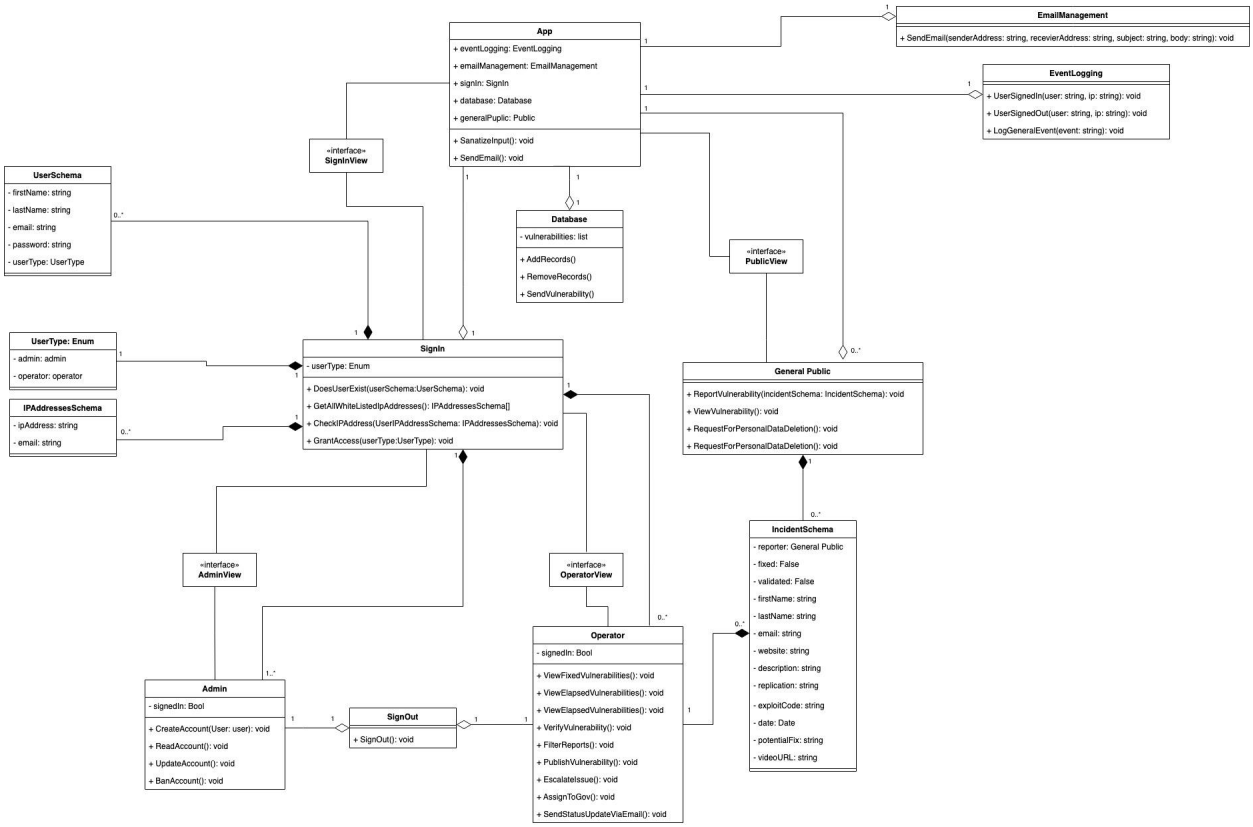
SQLite (2022) Limits In SQLite. Available from: https://www.sqlite.org/limits.html#:~:text=Maximum%20Number%20Of%20Pages%20In%20A%20Database%20File&text=The%20largest%20possible%20setting%20for,this%20limit%20at%20run%2Dtime. [Accessed 29 October 2022].

Veale, M. & Brown, I. (2020) Cybersecurity. *Internet policy review* 9(4): 1-22.

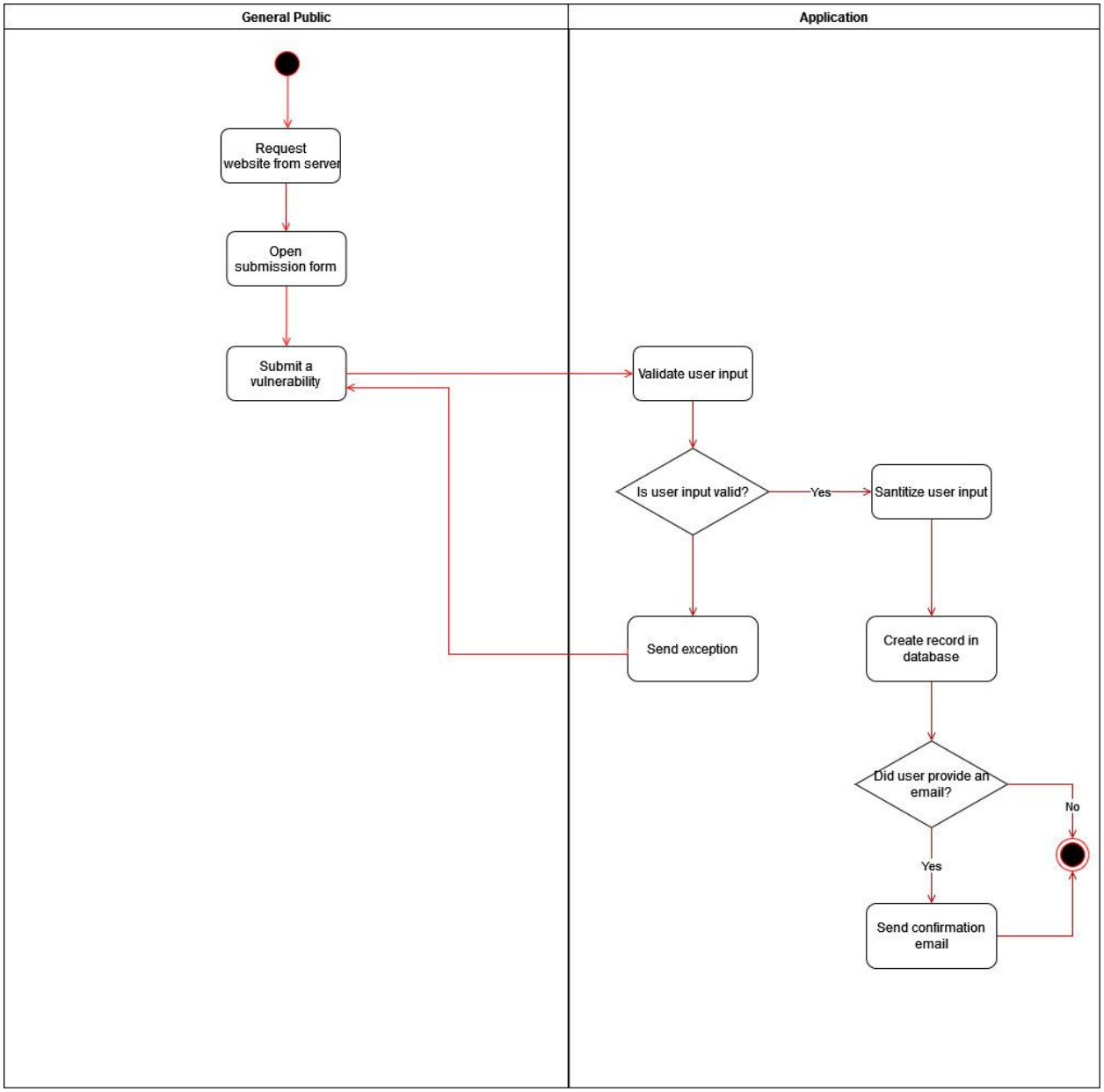
Appendix 1: Use Case Diagram



Appendix 2: Class Diagram



Appendix 3: Activity Diagram - Vulnerability submission by a General public user



Appendix 4: Activity Diagram - Operator Features