Development Team Project: Design Document

1. Introduction

The oversight of digital security in the Netherlands is the responsibility of the National Cyber Security Centre (NCSC), which has the primary objective of increasing the country's resilience to internet-related crime (NCSC, n.d.). The NCSC gathers information on cyber vulnerabilities on key national systems from the public.

Table 1: Coordinated Vulnerability Disclosure (CVP) online form data fields.

Field	Field Type	Size Limit	Required
First name	Text input	40	N
Surname prefix	Text input	40	N
Surname	Text input	40	N
E-mail	Text input	40	Y
Telephone number	Text input	40	N
Type of vulnerability	Radio Buttons	–	Y
Step-by-step explanation of vulnerability	Text area	None	Y
Explanation of why the vulnerability requires reporting	Text area	None	Y
Domain name(s) or IP address(es) related to report	Text input	40	Y
User's PGP key	Text input	40	N
Acceptance of terms of CVD	Radio button	–	Y
Acceptance of personal data processing agreement	Check box	–	Y

The NCSC website provides an online form (NCSC CVD page, n.d.) which members of the public can use to make Coordinated Vulnerability Disclosures

(CVDs) which are captured, assessed and actioned on. Apart from information about the vulnerability, the form only requires for a valid email address, but the user is free to provide additional personal information. Table 1 summarizes the information captured by the online form, which especially important because it specifies the information that the intended web application should capture, given a lack of access to officials in the NCSC.

The online form also provides a PGP key (Al-Kabi and Ahmad, 2012; Van Oorschot and Wiener, 2005) which the user can optionally use to manually encrypt the information they intend to provide before pasting them into the text fields. Furthermore, the form allows the user to provide their own PGP key which the NCSC will use to encrypt any responses to the user. Therefore, in addition to CVDs, the NCSC may require secure communication to and from the informant. The intended web application described in this document will automate this process by allowing and encrypting messages to and from users, which will not only significantly enhance security but also make it natural and effortless to do so.

2. Requirements and Assumptions

1. **Access**: The system should be accessible both locally and remotely, and therefore it is necessary to deploy it on a cloud server. According to W3Techs, Apache is the most popular web server used for hosting websites, with a market share of 32.2% as of March 2023 (W3Techs, 2023) and is therefore a good choice.
2. **Storage and CPU capacity**: Given that vulnerability reports are not frequently received, and the data is in any case text-based, a medium-sized storage capacity along with a typically powerful CPU with sufficient RAM will suffice for this application. This is in exclusion of malicious traffic which

should be monitored and controlled by means of an Intrusion Detection and Prevention System.

3. Design Approach

1. Encryption: All data will be encrypted for confidentiality and data integrity. AES is a widely-used symmetric encryption algorithm that provides strong encryption (Papadimitriou & Giannetsos, 2016) and is supported in Python (Daemen & Rijmen, 2002).
2. Data Storage: The data captured by the system should be stored securely and in compliance with the GDPR. An SQLite database will be used as a proof-of-concept, as it is lightweight and easy to deploy (Arockiam & Chandrasekaran, 2017) but should be replaced by a secure RDBMS such as Oracle or MySQL in the final production.
3. Frameworks: Flask will be used which is a lightweight and flexible framework suitable for small to medium-sized applications (Grinberg, 2018). SQLAlchemy will be used to store and retrieve information from the database, providing robustness to SQL injection and XSS attacks.
4. Libraries: The Fernet library will be used to allow for encryption and decryption of data. Also, the bcrypt library will be used to securely generate password hashes.

4. Potential Security Issues and Mitigation

Table 2: Potential security threats and vulnerabilities and mitigation.

Threat	Description	Mitigation
Cross-Site Scripting	Attacker injects client-side scripts into web pages viewed by other users.	Effective input validation and sanitization, output encoding, and the use of Content Security Policy (CSP) can mitigate this (Kirda et al., 2006)
SQL Injection	Attacker injects malicious SQL statements into input fields, potentially giving them access to sensitive data.	Use of parameterized queries, input validation and sanitization (Halfond et al., 2006)
Cross-Site Request Forgery	An attacker executes actions on behalf of a victim who is logged in.	The use of CSRF tokens, input validation, and limiting the number of requests from a single IP address (Barth et al., 2008)
Session hijacking	Attackers steal a user's session ID, potentially allowing them to take control of the user's account	Secure session management techniques, such as expiring sessions after a certain amount of time or after a user logs out (Sekar et al., 2001)
Insufficient authentication and authorization	The attacker is given unauthorized access to sensitive information or functionality	Strong password policies, two-factor authentication, and role-based access control (Juels et al., 2006)
Denial-of-service	Attackers may use (D)DOS attacks to take down the service.	Make use of an Intrusion Detection and Prevention System
Third-party software vulnerabilities	All third-party libraries, frameworks, APIs and software, if not updated regularly, can introduce vulnerabilities into the system	Regularly update all such components of the system (Frei, Christey, & Mohan, 2002)
Phishing and pretexting attacks on admin users	The attacker may use various social engineering techniques to obtain elevated (admin) access to the system	Conduct regular cybersecurity awareness training (Levy et al., 2016) and instil a security-aware culture (Lai et al., 2019)

5. Proposed System Design

The proposed system will allow users to securely submit vulnerability reports. It was decided to require the user to create an account and log in before submitting a report but, consistent with the NCSC's current workflow, the user will only be required to provide an email address. Once logged in, they will also be able to view their previous reports, or delete their personal details (but not their reports). Users will also be able to view/send secure messages from/to an admin user. Admin users will be able to manage users, manage vulnerability reports, and view/send secure messages to users based on a specific vulnerability report.

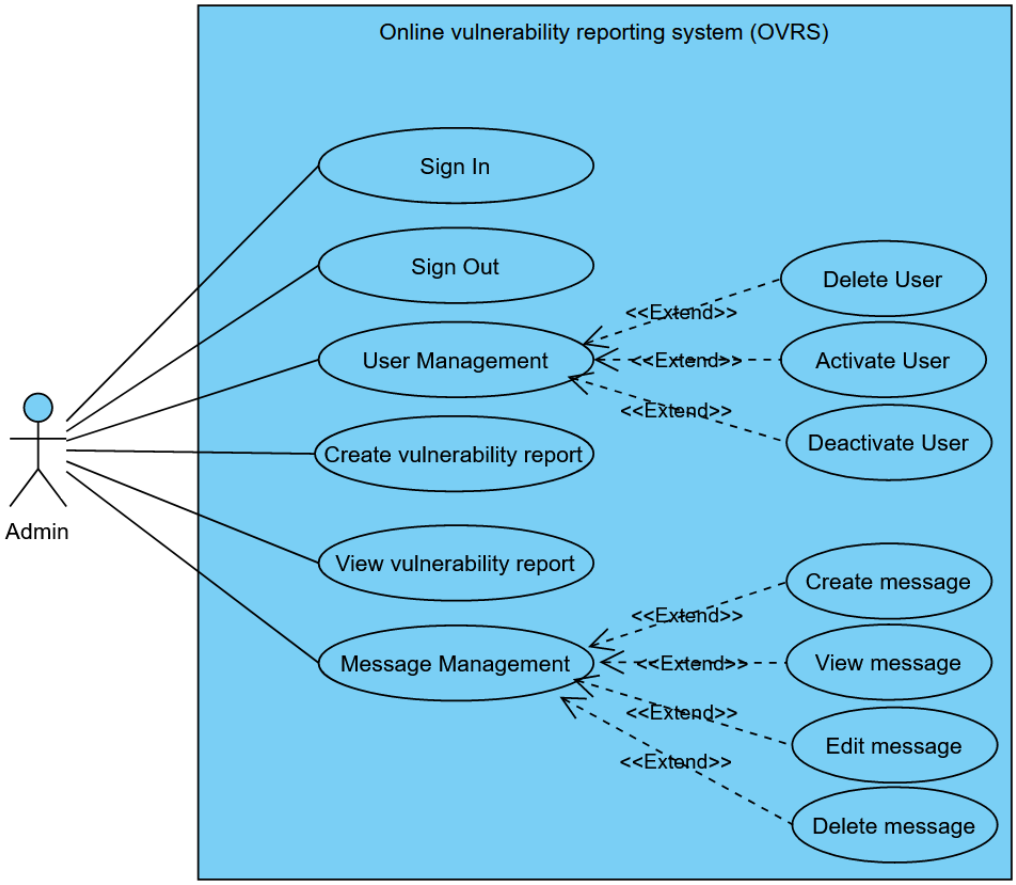

Figure 1: Use case diagram for admin user.

The actions available to admin and normal users are summarized in Figures 1 and 2. Figure 3 is a class diagram summarizing the main classes in the system, namely, a User, Report, Message and Vulnerability class representing the main data elements in the proposed system. Figure 4 is an activity diagram demonstrating key actions carried out by a user and the method of handling these actions by the web application.

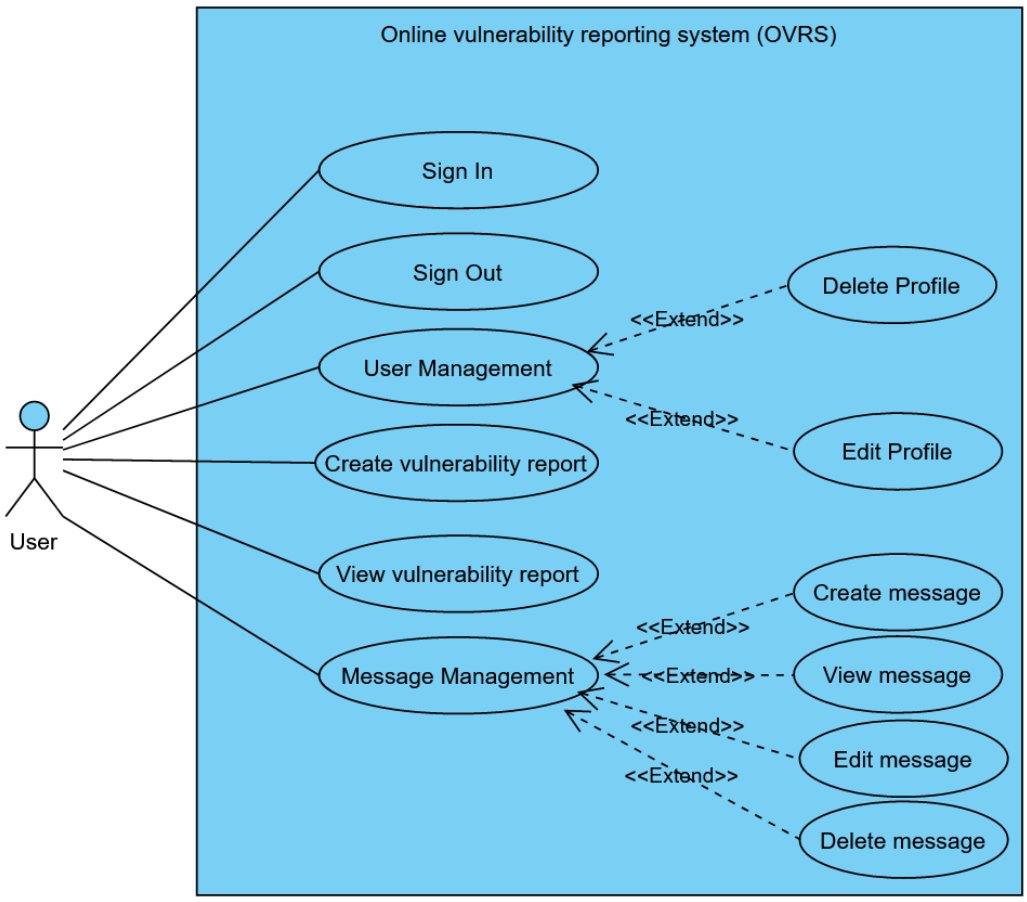

Figure 2: Use case diagram for user.

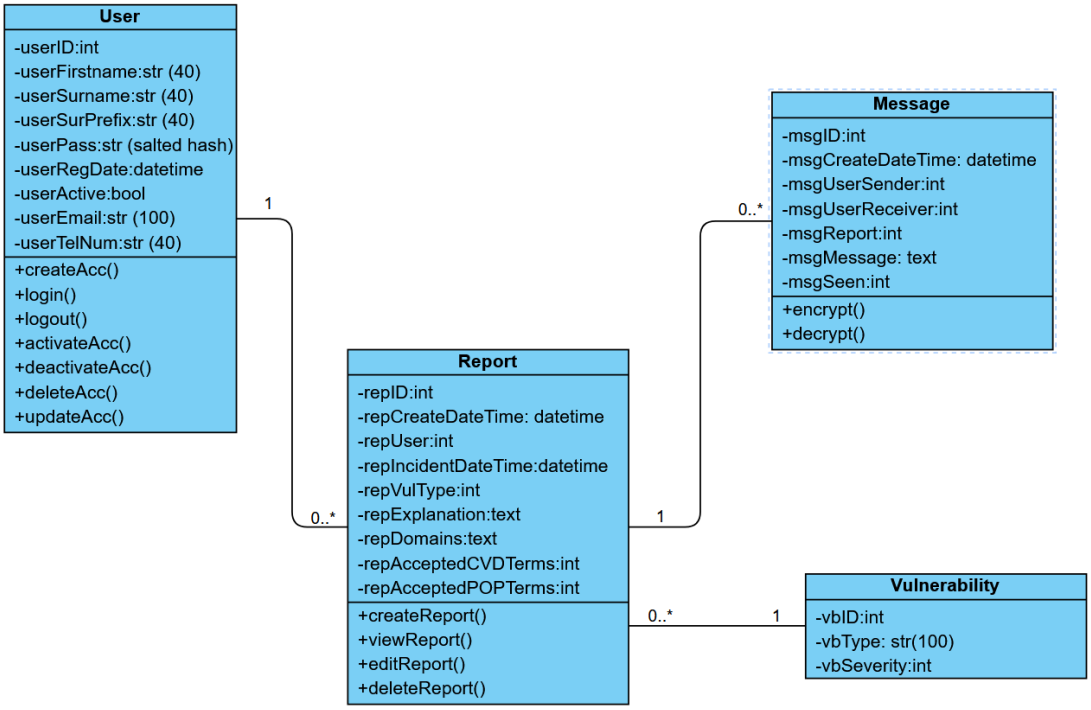

Figure 3: Class diagram of the proposed system.

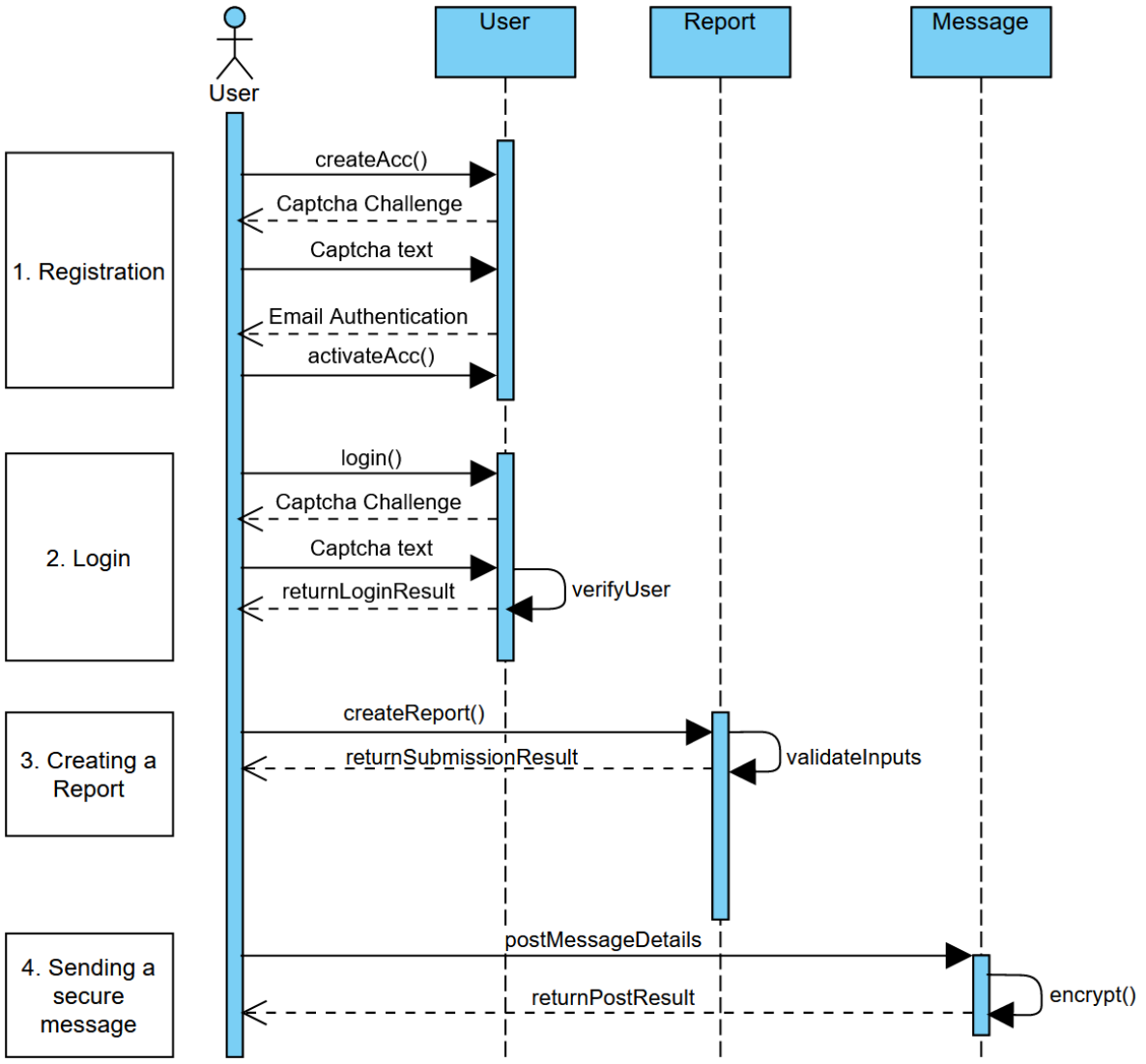

Figure 4: Activity diagram for four key activities.

6. References

Al-Kabi, M. H., & Ahmad, Y. F. (2012). A survey of PGP encryption technology. International Journal of Advanced Computer Science and Applications, 3(7), 11-16.

Arockiam, L., & Chandrasekaran, R. M. (2017). Comparative study of SQLite, MySQL, and Oracle. International Journal of Applied Engineering Research, 12(9), 2297-2301.

Barth, A., Jackson, C., & Mitchell, J. C. (2008). Robust defenses for cross-site request forgery. In Proceedings of the 2008 IEEE Symposium on Security and Privacy (pp. 75-89).

Frei, R., Christey, S., & Mohan, N. (2002). Analysis of third-party component vulnerabilities in commercial software. Proceedings of the 2002 ACM workshop on Computer security. New York, NY, USA: ACM.

Grinberg, M. (2018). Flask web development: Developing web applications with Python. O'Reilly Media, Inc.

Halfond, W. G., Orso, A., & Manolios, P. (2006). AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation (pp. 120-129).

Juels, A., Rivest, R. L., & Szydlo, M. (2006). The blocker tag: Selective blocking of RFID tags for consumer privacy. In Proceedings of the 8th international conference on Ubiquitous computing (pp. 103-116).

Kirda, E., Kruegel, C., Vigna, G., & Jovanovic, N. (2006). Noxes: A client-side solution for mitigating cross-site scripting attacks. In Proceedings of the 15th international conference on World Wide Web (pp. 931-940).

Lai, C. H., Chen, C. W., & Chang, C. L. (2019). The impact of information security culture on information security awareness and behavior. Journal of Information Privacy and Security, 15(1), 1-25.

Levy, N., Ramim, M. M., & Schurr, A. (2016). The effect of awareness and training programs on employees' knowledge of information security policy and compliance intentions. Journal of Information Privacy and Security, 12(3-4), 161-176.

National Cyber Security Centre (NCSC) | About NCSC. National Coordinator for Security and Counterterrorism, Government of the Netherlands, www.ncsc.nl/english/about-ncsc. Accessed: 17th March 2023.

National Cyber Security Centre (NCSC) | CVD report form. National Coordinator for Security and Counterterrorism, Government of the Netherlands, https://english.ncsc.nl/contact/reporting-a-vulnerability-cvd/cvd-report-form. Accessed: 17th March 2023.

Papadimitriou, P., & Giannetsos, T. (2016). Cryptography techniques and security assessment mechanisms for cloud computing environments. Journal of Cloud Computing, 5(1), 1-19.

Sekar, R., Bendre, M., Dhurjati, D., Bollineni, P., & Shi, J. (2001). A fast automaton-based method for detecting anomalous program behaviors. In Proceedings of the 2001 IEEE Symposium on Security and Privacy (pp. 144-155).

Van Oorschot, P., & Wiener, M. (2005). Pretty Good Privacy (PGP) and GNU Privacy Guard (GPG) for email security and digital signatures. Proceedings of the IEEE, 93(3), 547-560.

W3Techs. (2023). Usage statistics and market share of Apache. https://w3techs.com/technologies/details/ws-apache . Accessed: 20th March 2023.