Report: Secure (Digital) Repository for the Dutch National Cyber Security Center

University	*Essex University*
Place & date:	*Remote; 20-07-2022*
Name, initials:	*Nicolas Haas, Mohammad Atieh, Djordje Savanovic, Roberto Cappiello*
Study:	*Computer Science*
Supervisor:	*Dr. Cathryn Peoples*
Word Count:	*1038*

Defense in Depth

Software Development Proposal Report

EXECUTIVE SUMMARY

This proposal outlines a digitally secure repository solution for the National Cyber Security Center (NCSC) as a department under the National Coordinator of Counterterrorism and Security of the Government of the Netherlands to manage cases of suspected internet crime in accordance with the national counterterrorism and cybersecurity policy (Government of the Netherlands, N.D.). In addition, the NCSC helps public authorities in providing information about new IT software and hardware trends and updates, and this will also be a part of the solution. In general, the focus will be on functionality and security. Functionality will be ensured through guaranteeing simple CRUD operations. Security relies on the principle of “Defense in Depth”. The solution uses Firebase (Firebase Realtime Database for textual data, Firebase cloud for documents, Firebase Authentication and Firebase events) in addition to other tools for implementation and testing.

PROJECT OVERVIEW & SYSTEM REQUIREMENTS

The tool will allow authorized employees (Cyber specialists) of the NCSC to access a database management system via a homepage in order to add, delete, update, and read cases of internet crime. Moreover, they will be able to perform CRUD operations for IT-Updates (software and hardware) and for the list of authorities collaborating with the NCSC. Security is an essential part of the system functionality, which will be covered in the next section.

The system will be built using the following assumptions:

1. The system is primarily intended for use by Cyber Specialists of the NCSC.
2. Members of the general public and restricted users (i.e. unauthorized NCSC employees) can send information (reports/advices) to the NCSC via a general, publicly accessible form (without logging in).
3. Cyber Specialists will use the application for creating, reading, updating and deleting reports/advices on cybercrimes.
4. Cyber specialists can maintain the table of IT-Updates and the table of authorities (also all CRUD operations are included).

In addition, the system allows users to perform the following tasks:

1. Adding, deleting and updating a new report/new authority/new IT-Update
2. Querying cases/IT-Updates/authorities using search terms and IDs
3. Providing report statuses (to do, in progress, completed, declined)

Figure 1 depicts the entire system in an end-to-end diagram.

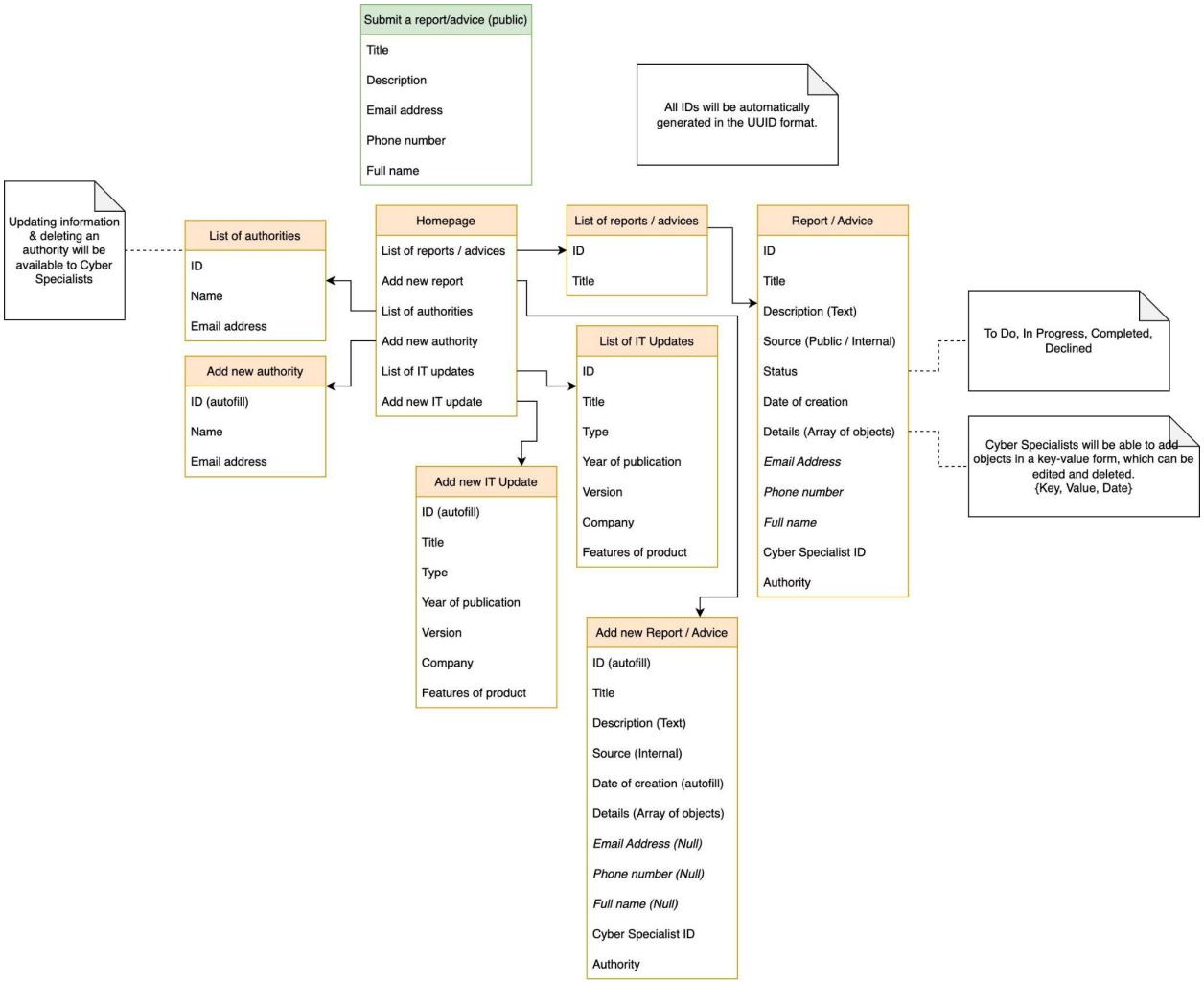

Fig 1. Site Map. The private application has eight web pages (where only the cyber specialist has access to) so that on the main page the Cyber Specialist starts by choosing one of the 6 options (list of reports/advice, add new report/advice, list of IT Updates, add new IT Update, List of authorities, add new authority). After that, the actions can be divided into two kinds: One where a new entry will be added, this requires information from the Cyber Specialist, which will be checked before sending it to the DB. The other option is simply searching/selecting something in/from a list (of reports, IT Updates, authorities) and then retrieving the data from the DB and showing it to the user. Aside from the private application, there is also the possibility to always submit reports/advices over a public form (in green) which will be available for the general public to reach out to the NCSC.

The following activity diagrams (Figure 2 and Figure 3) focus on key functional requirements.

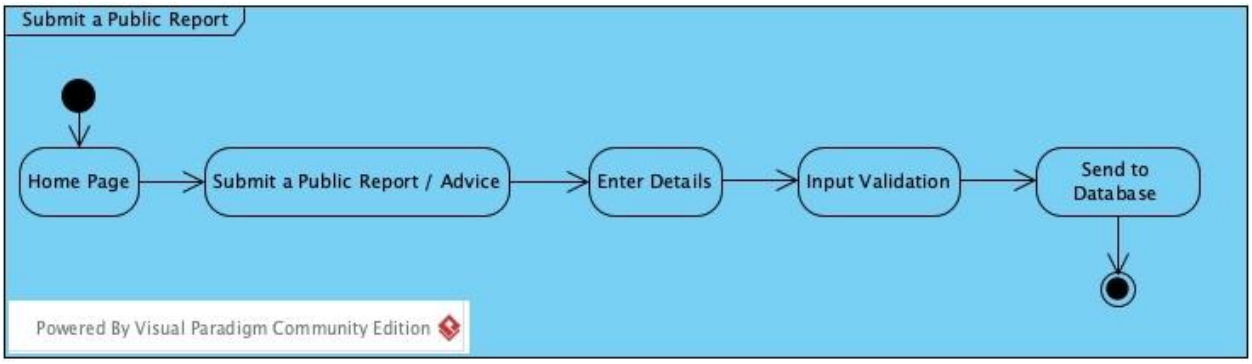

*Fig 2. Submit a Public Report. This activity diagram is showing the steps of submitting a **public report** by a **general public user**. This process doesn't need a login.*

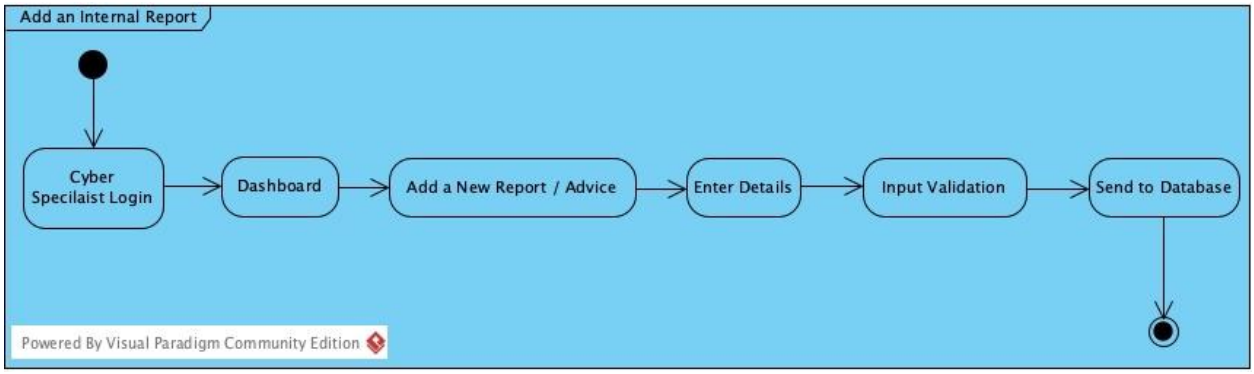

*Fig 3. Submit an Internal Report. This activity diagram is showing the steps of adding an **internal report** by a **cyber specialist**.*

For running the application in a Chrome Browser, there are non-functional requirements that need to be met (regarding OS and CPU), which are listed in Table 1.

Table 1

System Requirements			
	Windows	**Mac**	**Linux**
Operating System*	Windows 7 or later	OSX 10.11 or later	64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, or Fedora Linux 32+
Processor*	Intel Pentium 4 processor		
Memory	8 GB		
Internet speed	High-speed internet (25 Mbps or more)		
Other	Javascript support		

*Operating system and processor requirements are based on the minimum system requirements suggested for running the Chrome browser (Google, n.d)

The application itself will be decoupled in a few ways. The application's authentication and authorization will run through Firebase Authentication, whereas the actual data will be stored in Firebase Realtime Database (textual data) and Firebase Cloud (documents). Users will be added directly in the Firebase Authentication control panel. The code for the user interface of this app will be hosted on GitHub, but the actual deployment will run with the help of Netlify, based on every code change in the master branch. This software will run on any modern internet browser that supports JavaScript (reactjs.org, n.d.). The connection to our database will only be available from our app, as it requires the usage of an API key, as well as other API configurations known only to us (developers).

The decision of choosing this set-up is simply explained by two reasons:

- Firebase products provide necessary functionalities, as well as proper security. They are trusted by the developer community and allow us to develop the system in a shorter amount of time.
- The decentralized structure for the storage of data files through Firebase Realtime Database and Firebase Cloud leads to more efficient requests (CRUD). Furthermore, Firebase provides several tools for handling concurrent threads (ThreadManager in FirebaseApp or Cloud Firestore) for which a decision will be taken when the application already has some fundamental structure.

SECURITY VULNERABILITIES

Since this is a web based application, it is inherently exposed to the OWASP top 10 vulnerabilities which will be dealt with using strict security requirements (see next section) (OWASP, 2021).

Those vulnerabilities are in Table 2:

Table 2

A01-2021	Broken Access Control
A02-2021	Cryptographic Failures
A03-2021	Injection
A04-2021	Insecure Design
A05-2021	Security Misconfiguration
A06-2021	Vulnerable and Outdated Components
A07-2021	Identification and Authentication Failure
A08-2021	Software and Data Integrity Failures
A09-2021	Security Logging and Monitoring Failures
A10-2021	Server-Side Request Forgery

SECURITY REQUIREMENTS

Layer of Defense (Pillai, 2017; Bernard Institute of Cybersecurity, 2020; OWASP, 2021):

a. Secure firewall on the servers where the application and the data is stored (on the Firebase server).

b. Secure firewall on local machines of different (remote) users (**A01-2021).**

c. Firebase ensures (because of its NoSQL approach) that the data is collected on a decentralized basis (i.e. on different machines). This makes the whole system more secure because it is only partially vulnerable (**A01-2021).**

d. Restrict a user in always accessing only a small part of the data at a time. In other words, if there is a request command/statement to access data, then make sure that firebase only gives limited access to data in the way that is needed (i.e. use more precision in defining rules, avoid global rules) (**A01-2021)**.

e. Prefer the use of local variables rather than global ones (remember that global variables aren't restricted by any permission rights).

f. Use validation before sending a request to the database. For example, use a type check when input of a user has been sent (**A03-2021 and A10-2021)**.

g. Access, Authentication and Authorization (**A01-2021 and A07-2021)**, see Table 3:

Table 3

Layer	Name	Description
1	Uniquely identify system users	Give every user an ID
2	rights / role-based access controls (use of "allow" method in firebase)	Differentiate the access to data depending on the user
3	Passwords and password controls	Log in to application with multi-factor authentication Strong password policy (length and character; rotation policy) Limit and record failed logins Use generic failed login messages to prevent enumeration attack Manage session security
4	List/hierarchy of roles	Superadmin: Oversight users permissions, facilitate transmission of sensitive data from NCSC to authorities, set up accounts for NCSC employees. Cyber Specialist: inform local/government authorities regarding possible suspects of security flaw, log reported vulnerabilities, monitor suspect sources on the internet, monitor software developments in digital technology and update security systems.

h. Encryption (**A02-2021)**

Certain parts of the stored data will be encrypted with AES encryption, through the help of crypto-js library (npm, 2016). This data will be decrypted once it is fetched in the front-end.

i. Security event logging (**A09-2021)**

j. Produce-Consumer concurrency

In addition, vulnerabilities **A04-2021,A05-2021,A06-2021 and A08-2021** will be managed through careful design, code review, application settings and permissions review, and the use of up-to-date libraries and tools.

DATA PROTECTION REQUIREMENTS

The application adheres to GDPR policies consenting the users to amend/request the deletion of their personal information (ico.org.uk, 2021). However, this may not apply to cases that contain important information relating to criminal investigation or proceedings. Moreover, a logging system monitors potential inappropriate access and disclosure of data (ico.org.uk, 2021). The application will also specify different categories of personal data (ico.org.uk, 2021).

TOOLS

Table 4

I	*Reactjs + reactstrap for frontend*
II	*Netlify for deployment*
III	*Firebase Realtime Database for textual data*
IV	*Firebase Cloud Storage for documents.*
V	*Firebase Authentication.*
VI	*Firebase Events.*
VII	*Moment-js*
VIII	*Crypto-js*

DESIGN

The following use case diagram shows the different types of user interactions (Fig 4.).

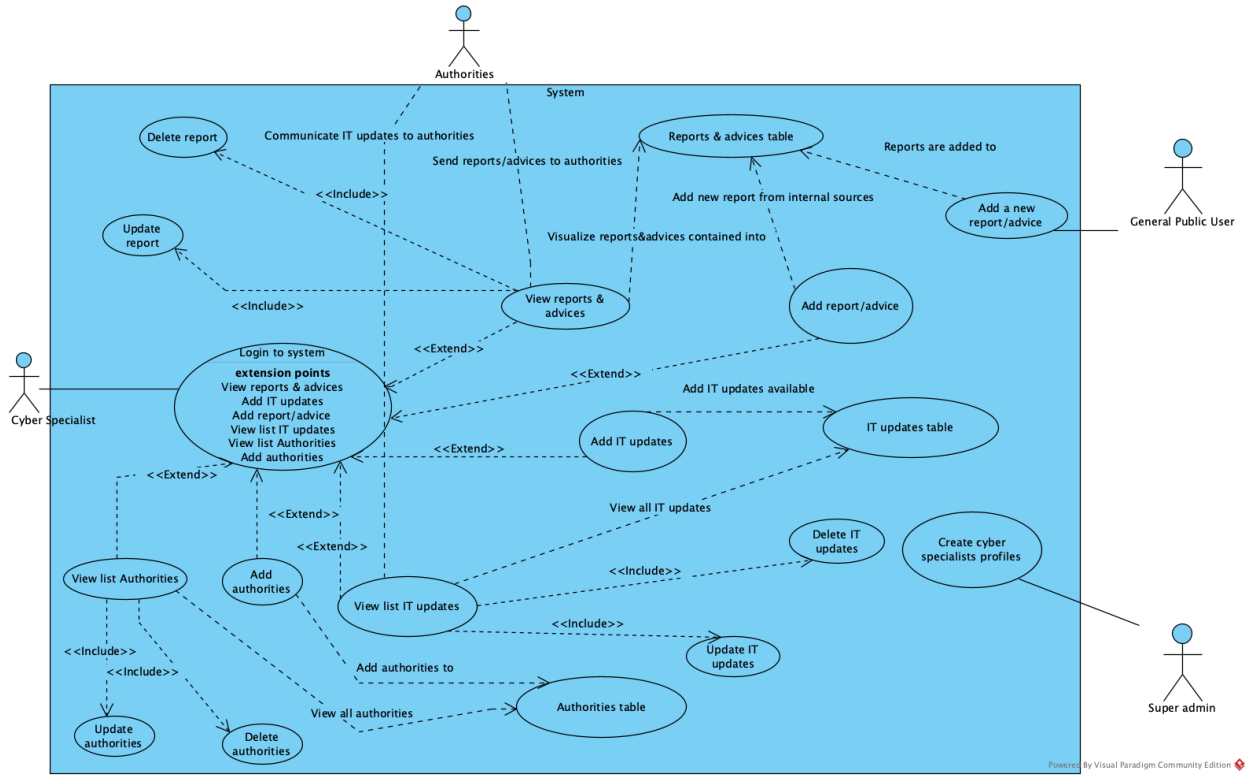

*Fig 4. Use Case Diagram. This use case diagram gives an overview of the functionalities offered by the system and the role of the different actors. The **super admin** is responsible for setting up the **cyber specialist** profiles to grant permission to login into the system. Through a home page menu the cyber specialist is able to access different functionalities to perform a series of actions such as viewing, adding and updating the reports /advices, IT updates, and the list of **authorities** to inform when a security flaw is reported on an ICT system. When the **general public** accesses the system, they are able to access only the "submit report/advice" function to report any system vulnerabilities to the NCSC or give advice in case of a new cyber threat on the Internet.*

References

Bernard Institute of Cybersecurity (2020) *Cybersecurity Architecture based on a Defense-In-Depth Design.* Available from: https://www.youtube.com/watch?v=6l1nz5y6jrk [Accessed 08 July 2022]

Google (n.d.). Chrome Browser system requirements - Google Chrome Enterprise Help. Available from: https://support.google.com/chrome/a/answer/7100626?hl=en [Accessed 11 Jul. 2022].

Government of the Netherlands (n.d.) Fighting Cybercrime in the Netherlands.Available from: https://www.government.nl/topics/cybercrime/fighting-cybercrime-in-the-netherlands [Accessed 17 July 2022]

ico.org.uk. (2020). *Logging*. Available from: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-le-processing/accountability-and-governance/logging/ [Accessed 10 July 2022].

ico.org.uk. (2020). *The right to erasure and the right to restriction*. Available from: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-le-processing/individual-rights/the-right-to-erasure-and-the-right-to-restriction/ [Accessed 10 July 2022].

ico.org.uk. (2021). *Categorisation of individuals*. Available from: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-le-processing/accountability-and-governance/categorisation-of-individuals/ [Accessed 10 July 2022].

npm. (2016) crypto-js. Available from: https://www.npmjs.com/package/crypto-js [Accessed 18 July 2022].

OWASP (2021) OWASP Top Ten. Available from: https://owasp.org/www-project-top-ten/ [Accessed June 25 2022].

Pillai, A.B. (2017) *Software Architecture with Python*. Birmingham, UK. Packt Publishing Ltd.

reactjs.org. (n.d.)JavaScript Environment Requirements – React. Available from: https://reactjs.org/docs/javascript-environment-requirements.html [Accessed 10 July 2022].