University of Essex

Secure Software Development (Computer Science)

Development Team Project:

Design Document

Group 1

Bernhard van Renssen, Chan Kei Yiu, Hung-Wei Lin, Lai Yin Ping, and Yusuf Fahry

18 April 2022

Word count: 1055

Table of Contents

1. Introduction 3

1.1. Brief overview 3

1.2. Problem statement 3

2. System design 3

2.1. Scope of design 3

2.2. Assumptions 3

2.3. Security consideration 3

2.2.1 Security Vulnerabilities 4

2.4. Design Architecture 5

2.3.1 Two-factor authentication 5

2.3.2 Trust-based authorization 5

2.3.3 Development Tools & Libraries 9

2.5. Implementation 10

3. Tools, Testing and QA 13

3.1. Methodology for testing 13

3.1.1 Static code analysis – Pylint 13

3.1.2 Unit test – Pytest 14

3.1.3 Testing for security flaws – Bandit 14

4. References 14

Table of Figures

Figure 1: Use Case Diagram – 2FA and Trust-based authorization 6

Figure 2: Activity Diagram - 2FA login and Trust Zone update 7

Figure 3: Flowchart Diagram - 2FA and Trust Model 8

Figure 4: Use Case Diagram – Defining access to the database 9

Figure 5: Sequence Diagram – Users access to Database 10

Figure 6: ERD – Database design of the application 11

Figure 7: Sequence Diagram – Create in CRUD operations 11

Figure 8: Sequence Diagram – Read in CRUD operations 12

Figure 9: Sequence Diagram – Update in CRUD operations 12

Figure 10: Sequence Diagram – Delete in CRUD operations 13

1. Introduction

1.1. Brief overview

The arrival of the COVID-19 global pandemic in March 2020 changed the way we work, communicate and live. As social distancing became a regulation, companies had to adapt and digitize virtually overnight, to allow staff to be able to work remotely (Tasheva, 2021). Due to this rapid need for digitization, cyber-attacks increased by over 100% (Norton, 2021).

In 2020, CERN - a technology company specializing in Particle Collision – contracted an external company focusing on cybersecurity to find what internal data has been exposed. Fortunately, only minor findings were exposed and reported which were easily dealt with internally (Computer Security Team, 2020).

1.2. Problem statement

With this as background, CERN has contracted us to develop a secure domain repository that follows the best security practices (as outlined by OWASP, 2021 and GDPR) to minimize the threat of current-day cyber-attacks.

As personal data needs to be secured and protected by any company holding such information (Intersoft consulting, 2018), the secure repository should adhere to all relevant statutory regulations, by providing protection of this personal data against cyber-attacks.

2. System design

2.1. Scope of design

The proposed design will allow users to securely access, upload and download data, to allow secure management and storage of their data. Building this repository as a web-application, a great deal of emphasis will be placed on relevant security best-practices, namely: Encryption of data, Security Monitoring, Propper Logging practices as well as continuously checking for common vulnerabilities – as outlined by OWASP Top 10 (OWASP, 2020) (Krishna, 2021).

Giving focus on the development of a secure WEB application, the framework will be built using AWS (Amazon Web Services). This decision is due to the efficiency in which these SaaS (Software as a Service) can handle scalability and cost, as well as possible tools and add-ons that these services can provide to stay up-to-date on security issues (Krishna, 2021).

2.2. Assumptions

- A working prototype will be built as per the brief, thus relevant fake data will be generated to demonstrate the prototype – stored in a local DB.
- The GDPR legislation will be used as international code standard
- Common python libraries are assumed to be secure

2.3. Security consideration

The STRIDE threat model is a threat modeling framework introduced by (Shostack, 2007), and used and discussed in different industries' case studies (Satar et al, 2021), (Køien, 2021). Also, the OWASP (2021) promotes the top ten risks which are vital for web applications. We apply the STRIDE model and the OWASP list to analyze and evaluate security threats and vulnerabilities of the system

Furthermore, to ensure the system also complies with the ISO 27001 information security standard (confidentiality, integrity, availability, and non-repudiation), the design will focus on following features:

- Authentication verification (verifying valid registered users and generating session to authenticated users, rejecting unauthorized requests/users)
- Ensuring blocking mechanisms are in place
- Role-based access control via secure authorization (verifying user's session before accessing to protected resources, information and files, and grant/refuse a user's access based on permissions)
- Logging and monitoring (investigating and tracing users' activities via digital footprints)

2.2.1 Security Vulnerabilities

With OWASP (2021) defining key security vulnerabilities, our system design will focus on these vulnerabilities by focusing on the following mitigations:

Vulnerability	Mitigation
Broken Access Control	- set deny by default except for public resources - implement access control mechanisms once and re-use them - log access control failures
Cryptographic Failures	- do not store sensitive data unnecessarily - encrypt all sensitive data at rest - ensure up-to-date and strong algorithms, protocols and keys - use proper key management - apply required security controls as per data classification
Injection	- keep data separate from commands and queries - use a safe API - use positive server-side input validation
Identification and Authentication Failures	- implement multi-factor authentication - implement weak password checks - use a server-side, secure, built-in session manager that generates a new random session ID after login
Denial-of-Services	- filter out malicious or non-compliant traffic - access control (authentication, user lockout) - use threading - prevent single point of failure - pooling and caching
Buffer Overflow	- apply latest patches - scan application periodically - review codes from users via HTTP requests

2.4. Design Architecture

2.3.1 Two-factor authentication

Multi-factor authentication (MFA) improves online data security by implementing multiple factors in addition to single factor sign-on. (Das et al. 2019) And two-factor authentication (2FA) is a subset of MFA. (Steven, 2016)

In addition to single-factor authentication (1FA), which is usually the Username and Password, 2FA provides a more secure alternative to 1FA which will require a user to also enter a pin sent via email or SMS.

2.3.2 Trust-based authorization

We are delivering a three-zone trust model (Table 1). Security Administrators would define which source IP networks would fall under which trust zone based on the IP range and log activity. It is possible to request a trust zone upgrade for a user's IP.

All the registered IPs fall under zone B by default, and logons from Trust Zone C (No trust) will be denied. Combined with the 2FA, the trust-based authorization mechanism ensures only non-malicious users' access can be granted.

Policy Item	High Trust	Medium Trust	No Trust
Trust Zone	A	B (default)	C
Login IP	Registered	Registered	Non-registered
Authentication (daily)	1FA	2FA	Logon denied
Authentication (weekly)	2FA	-	Logon denied
Session Timeout	14 hours	1 hour	-

Table 1: Trust-zone model - Define Security Strength

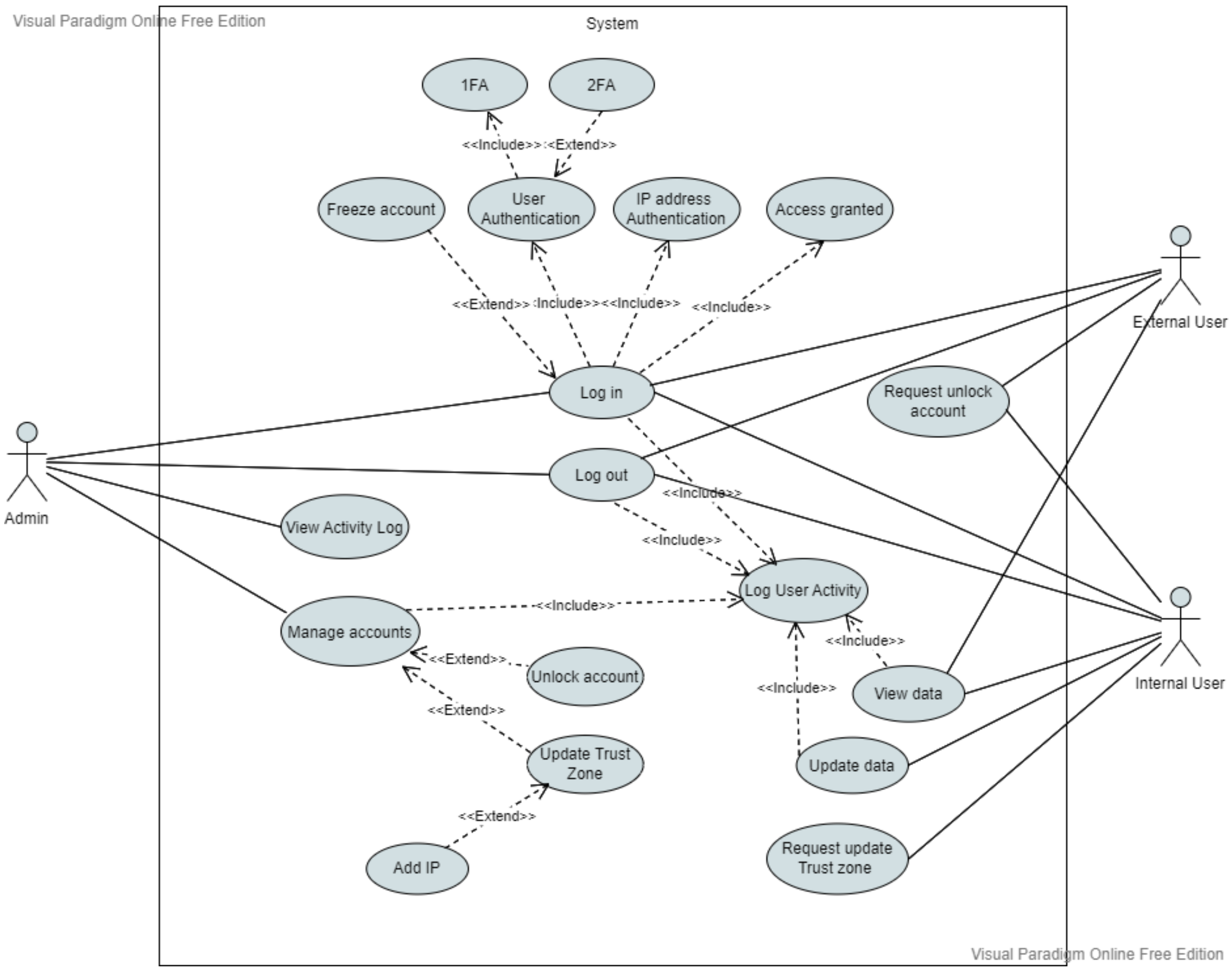


Figure 1: Use Case Diagram – 2FA and Trust-based authorization

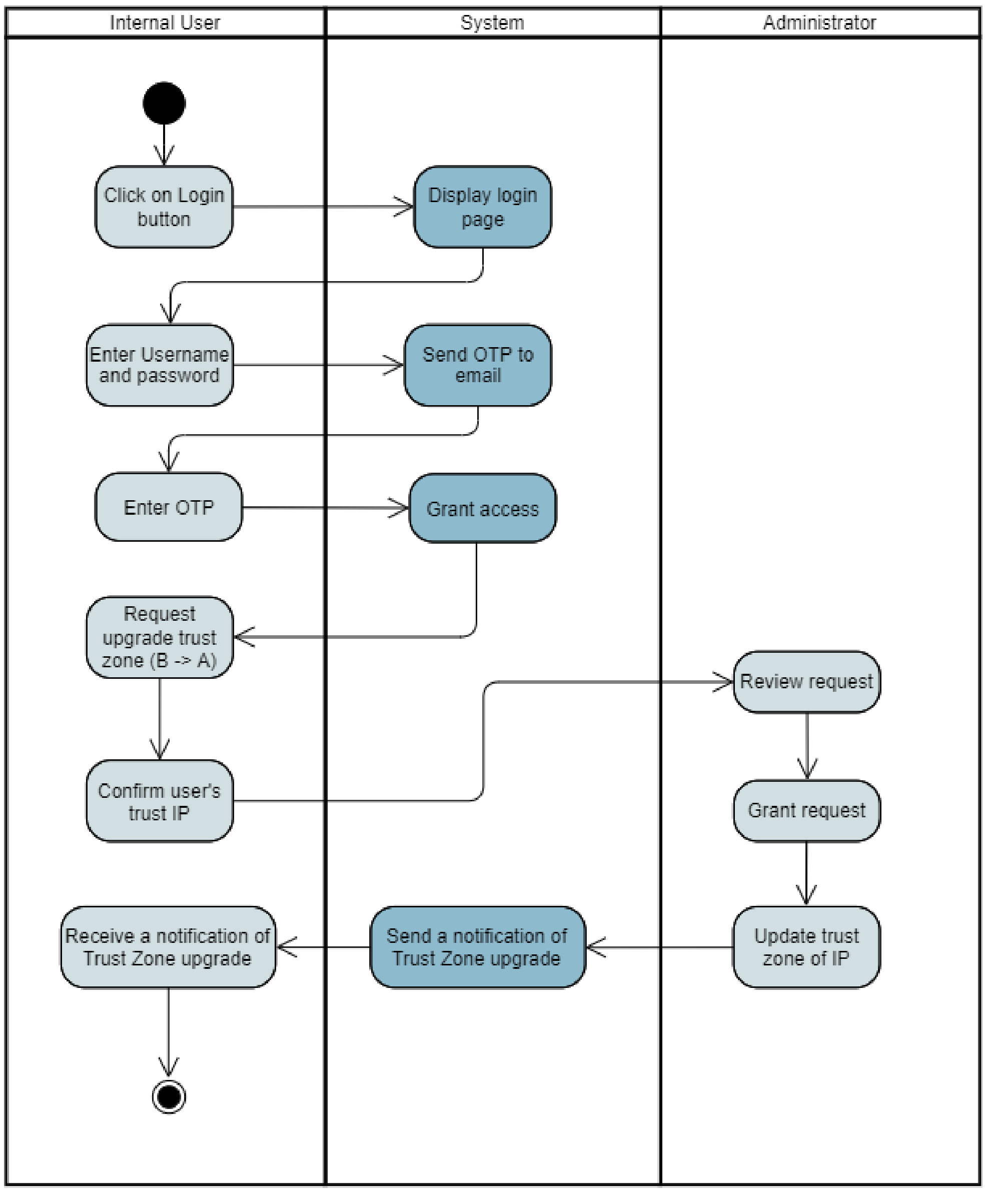
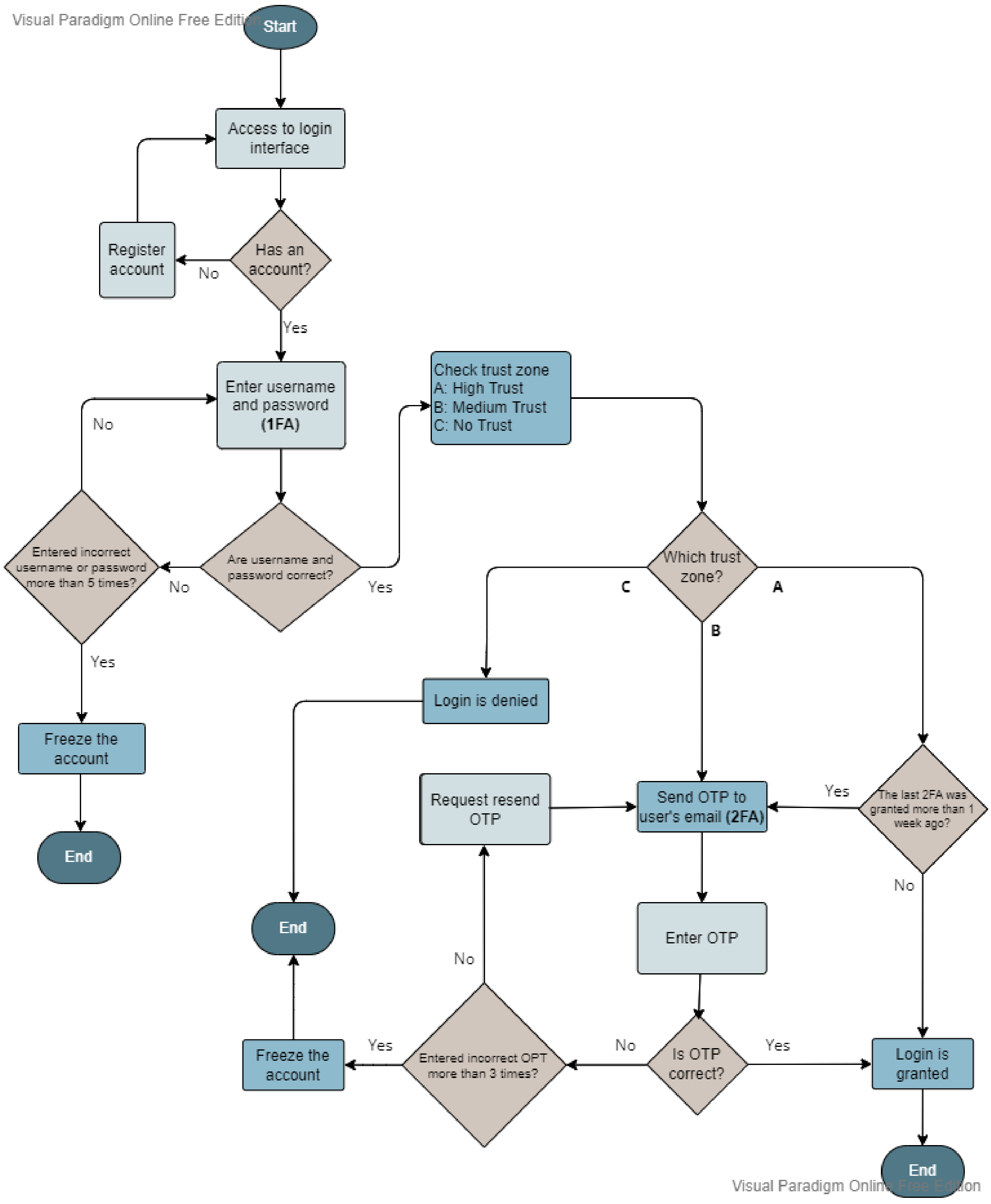


Figure 2: Activity Diagram - 2FA login and Trust Zone update



Visual Paradigm Online Free Edition
Start
Access to login interface
Has an account?
No
Register account
Yes
Enter username and password (1FA)
Are username and password correct?
No
Entered incorrect username or password more than 5 times?
No
Yes
Freeze the account
End
Yes
Check trust zone
A: High Trust
B: Medium Trust
C: No Trust
Which trust zone?
C
A
B
Login is denied
End
The last 2FA was granted more than 1 week ago?
Yes
No
Send OTP to user's email (2FA)
Request resend OTP
Enter OTP
Is OTP correct?
No
Yes
Entered incorrect OPT more than 3 times?
No
Yes
Freeze the account
End
Login is granted
End
Visual Paradigm Online Free Edition

Figure 3: Flowchart Diagram - 2FA and Trust Model

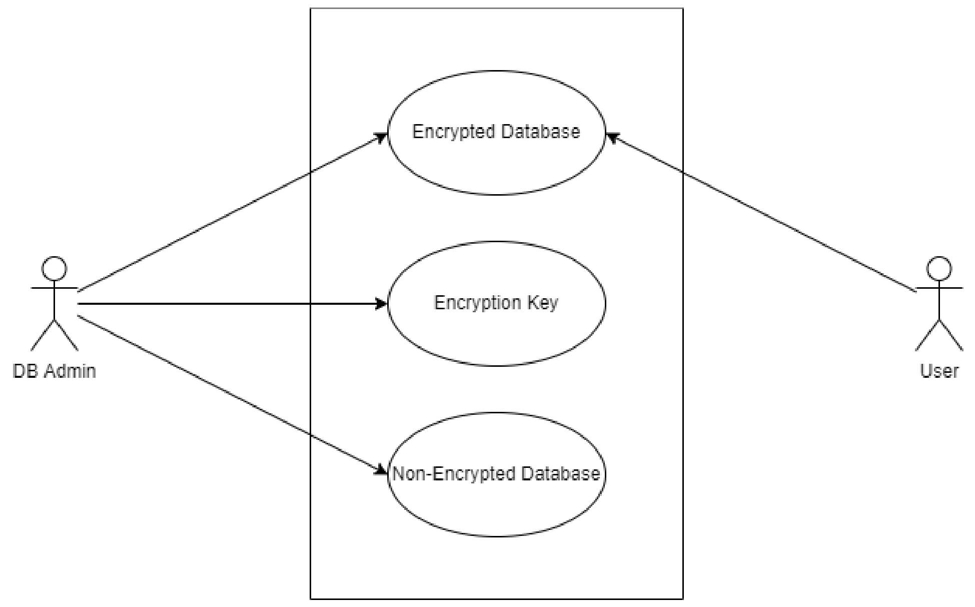


Figure 4: Use Case Diagram – Defining access to the database

2.3.3 Development Tools & Libraries

- Programming Language: Python 3
- IDE: Visual Studio Code
- Framework: Flask and Jinja template engine
- Database: MySQL
- Password Hashing: argon2
- Data Encryption & Decryption: Fernet
- Hashing Algorithms: hashlib
- Source Version Code Control: GitHub
- AWS RDS, EC2

2.5. Implementation

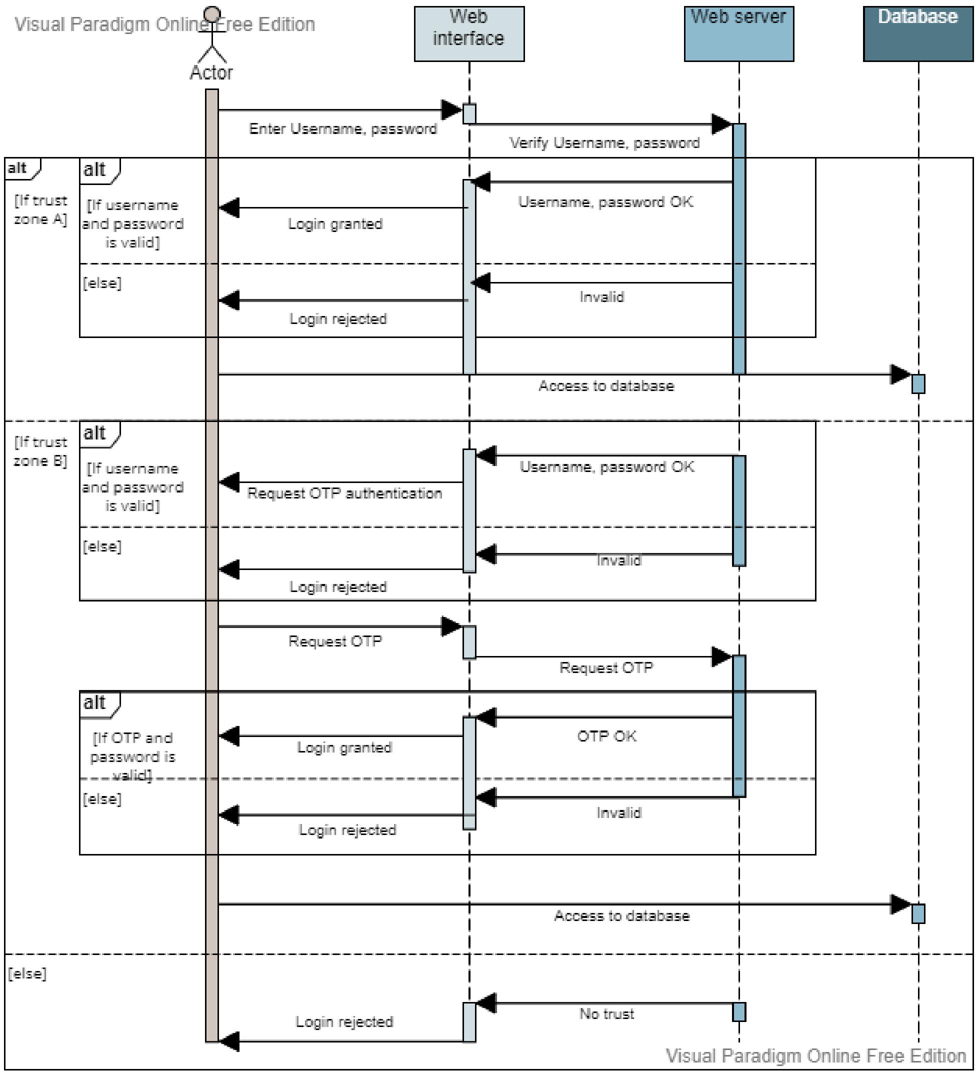


Figure 5: Sequence Diagram – Users access to Database

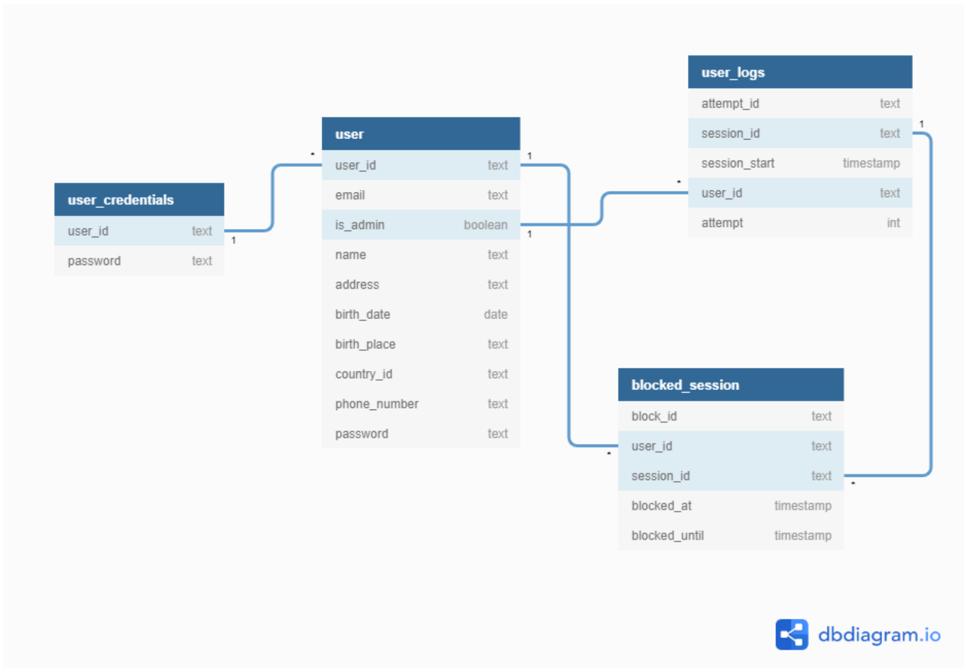


Figure 6: ERD – Database design of the application

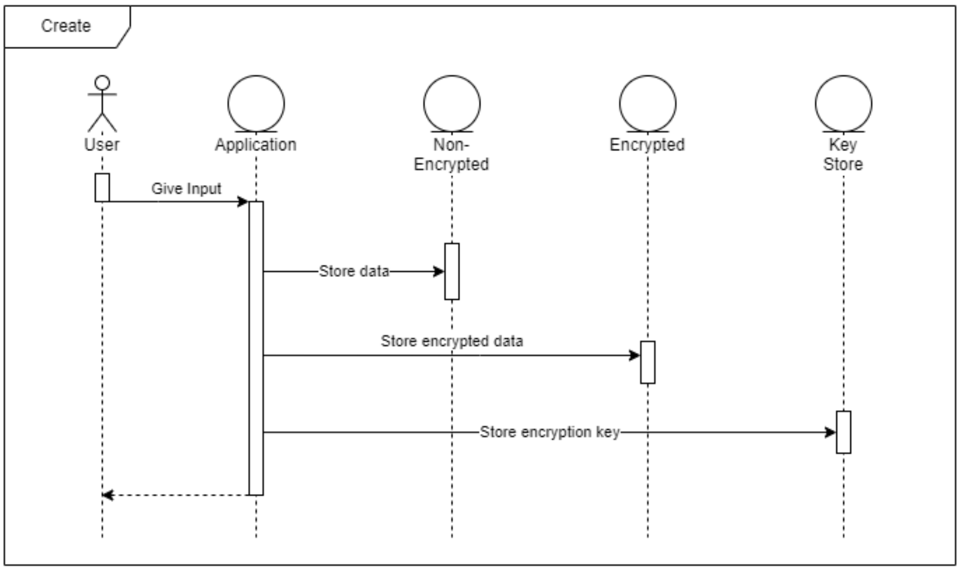


Figure 7: Sequence Diagram – Create in CRUD operations

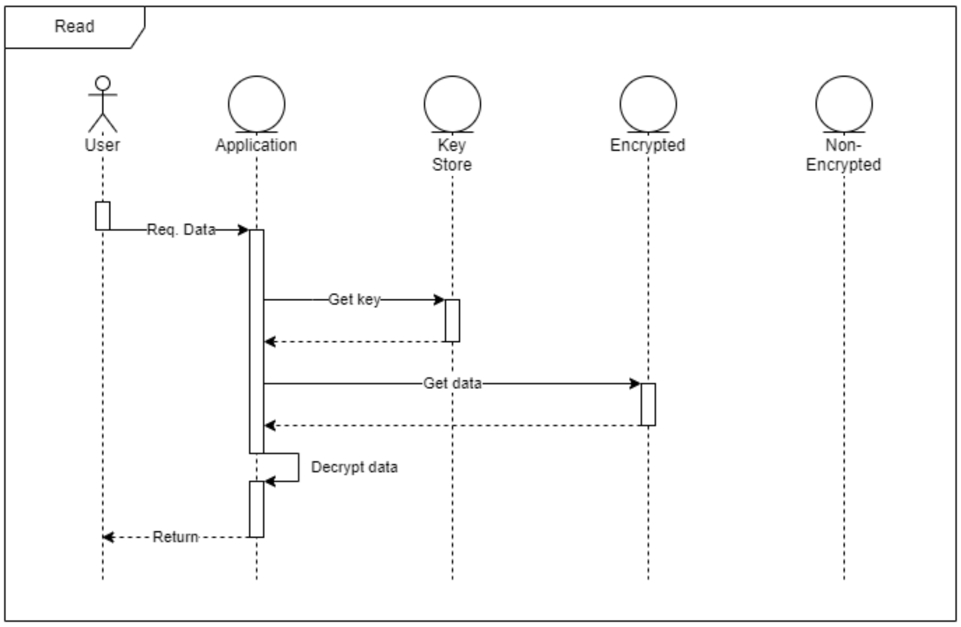


Figure 8: Sequence Diagram – Read in CRUD operations

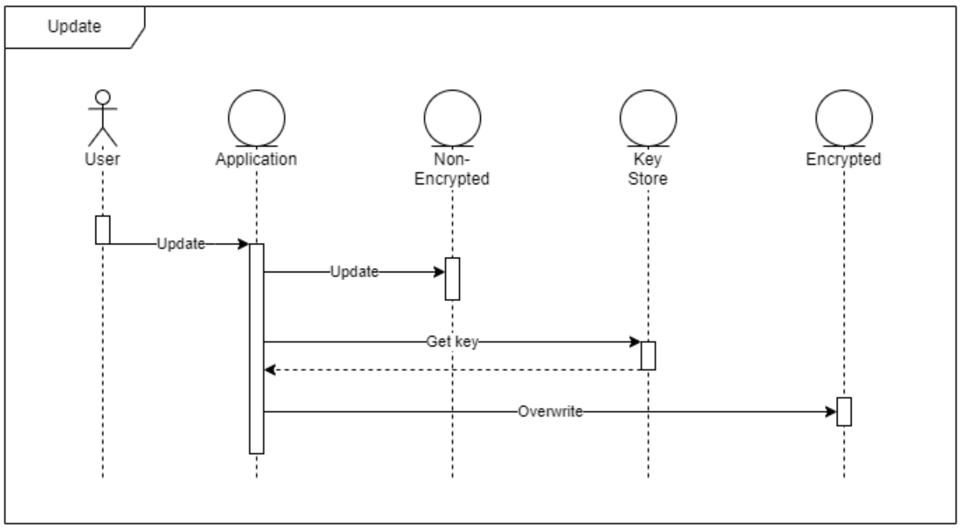


Figure 9: Sequence Diagram – Update in CRUD operations

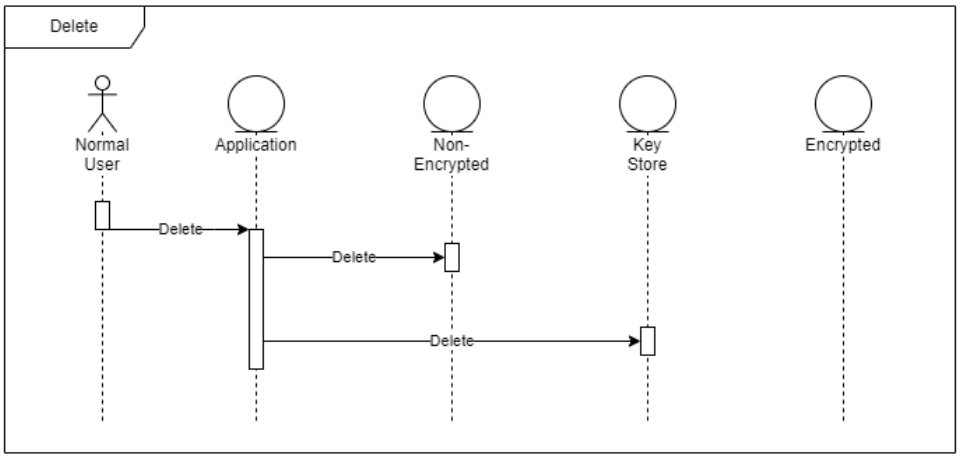


Figure 10: Sequence Diagram – Delete in CRUD operations

3. Tools, Testing and QA

To ensure that the system is performing against the requirements set-out in this document, the following list of testing and code-analysis will be used for Quality Assurance:

3.1. Methodology for testing

3.1.1 Static code analysis – Pylint

Static code analysis can help programmers identify errors without executing the code, and an initial indication of correctness of the code can be ensured (Louridas, 2006). At the first step, Pylint is used to check for errors in the Python code.

3.1.2 Unit test – Pytest

The second step is to perform unit test on code blocks. Pytest is a robust Python testing tool that can be used for many types of software testing (Okken, 2017). A proper unit test can ensure the code meets the standard and defects are identified at earlier stages.

3.1.3 Testing for security flaws – Bandit

To cope with the security standards in development, Bandit will be applied to check for common security issues in the Python code before deployment.

4. References

Ali, S., Waseem Anwar, R., & Khadeer Hussain, O. (2015) Cyber security for cyber physical systems: a trust-based approach. *Journal of Theoretical and Applied Information Technology* 20(2): 145. Available from: http://www.jatit.org/volumes/Vol71No2/1Vol71No2.pdf

Anna, K. Olena, K. Mykhailo, K. Svitlana, K. Olena, S and Rostyslav, Z. (2020) Methods of security authentication and authorization into informationals systems. *2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT*) 270-274. DOI: 10.1109/ATIT50783.2020.9349333.

Computer Security Team. (2020) Computer Security: Digital stolen goods of CERN?. Available from: https://home.cern/news/news/computing/computer-security-digital-stolen-goods-cern [Accessed 06 April 2022].

Das, S., Wang, B., Tingle, Z., & Camp, L. J. (2019) Evaluating User Perception of Multi-Factor Authentication: A Systematic Review. *arXiv preprint arXiv*:1908.05901. DOI: 10.48550/arxiv.1908.05901

Intersoft consulting. (2018) GDPR - Personal Data. Available from: https://gdpr-info.eu/issues/personal-data/ [Accessed 06 April 2022].

King, J. (2013) Measuring the forensic-ability of audit logs for nonrepudiation. *2013 35th International Conference on Software Engineering (ICSE)* 2013: 1419-1422. DOI: 10.1109/ICSE.2013.6606732.

Krishna, A. (2021). 7 Web Application Security Practices You Can Use. Available from: https://www.thesslstore.com/blog/web-application-security-practices-you-can-use/ [Accessed 06 April 2022].

Køien, G.M. (2021) A Threat Analysis of Human Bond Communications. *Wireless Pers Commun* 118**:** 1987–2013.

Logilab & PyCQA. (N.D.) Pylint 2.14.0-dev0 documentation. Available from: https://pylint.pycqa.org/en/latest/ [Accessed 8 Apr 2022]

Norton. (2021) 115 Cybersecurity statistics and trends you need to know. Available from: https://us.norton.com/internetsecurity-emerging-threats-cyberthreat-trends-

cybersecurity-threat-review.html
_[Accessed 06 April 2022].

Okken, B. (2017). *Python testing with Pytest: simple, rapid, effective, and scalable.* 2nd ed. Pragmatic Bookshelf.

Okta. (N.D.). Authentication vs. Authorization. Available from: https://www.okta.com/identity-101/authentication-vs-authorization/. [Accessed 5 Apr 2022].

OWASP. (2021) OWASP.
Available from: https://owasp.org/ _[Accessed 06 April 2022].

OWASP. (2021) OWASP Top Ten. Available from: https://owasp.org/www-project-top-ten/. [Accessed 5 Apr 2022].

Sattar, D. Vasoukolaei, A, H.Crysdale, P. and Matrawy, A. (2021) *A STRIDE Threat Model for 5G Core Slicing*, 2021 IEEE 4th 5G World Forum (5GWF) 247-252. DOI: 10.1109/5GWF52925.2021.00050.

Shostack, A. (2008) Experiences Threat Modeling at Microsoft. *MODSEC@MoDELS.*

Steven, F. (2016) Single-factor Authentication (SFA) vs. Multi-factor Authentication (MFA). Available from: https://www.centrify.com/blog/sfa-mfa-difference/. [Accessed 5 Apr 2022].

Tasheva, I. (2021) Cybersecurity post-COVID-19: Lessons learned and policy recommendations. *Sage Journals* 20(2): 140-149.