Word Count excluding references: 586

Introduction

This paper details the technical specification for the Dutch Police Internet Forensics system.

Addressing Privacy and Regulatory Compliance

Specific data laws of the Netherlands and European Union were examined to ensure the system is compliant with local laws such as General Data Protection Regulation (GDPR), European Data Protection Board (EDPB), Children's Online Privacy Protection Rule (COPPA) and an Initial Coin Offering (ICO).

The following measures were taken:

1. Encryption of all personal data with AES 256 and Salted MD5.

2. Employment of Hypertext Transfer Protocol Secure (HTTPS).

3. Verification of compliance among partners.

4. Provision for users' consent before collecting and processing personal data through active, unbundled and granular opt-ins as well as an easy opt-out.

5. Cookies consent

Proposed Functional Use Cases of System

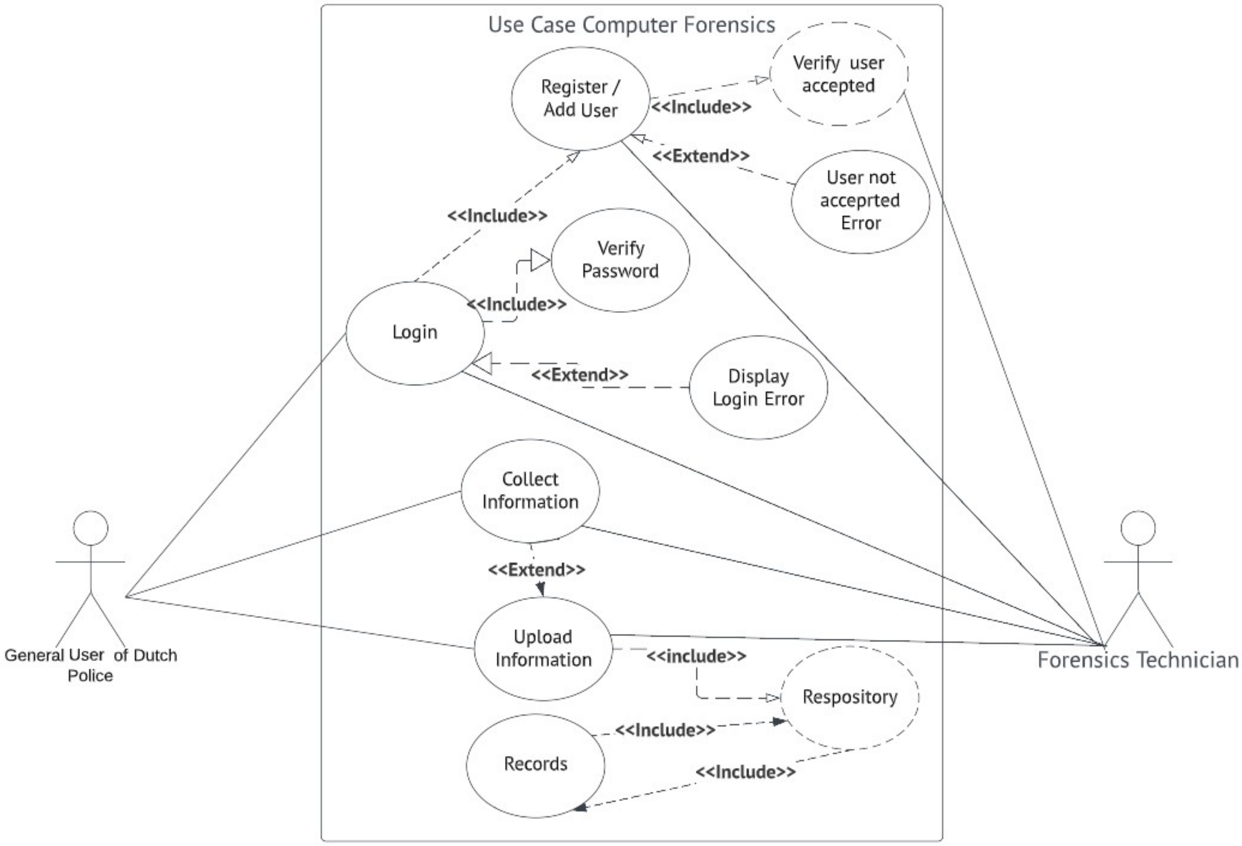

Fig 1 - Use Case Diagram

Threat Modelling Process

Threat modelling has been conducted throughout the early stages of the SDLC for detecting and ranking the potential risks to the system (Desmet, 2005). Dataflow modelling resulted in identification of threats and their potential mitigative countermeasures.

A hybrid methodology consisting of both OWASP and the STRIDE threat modelling framework was utilized (Bygdås 2021).

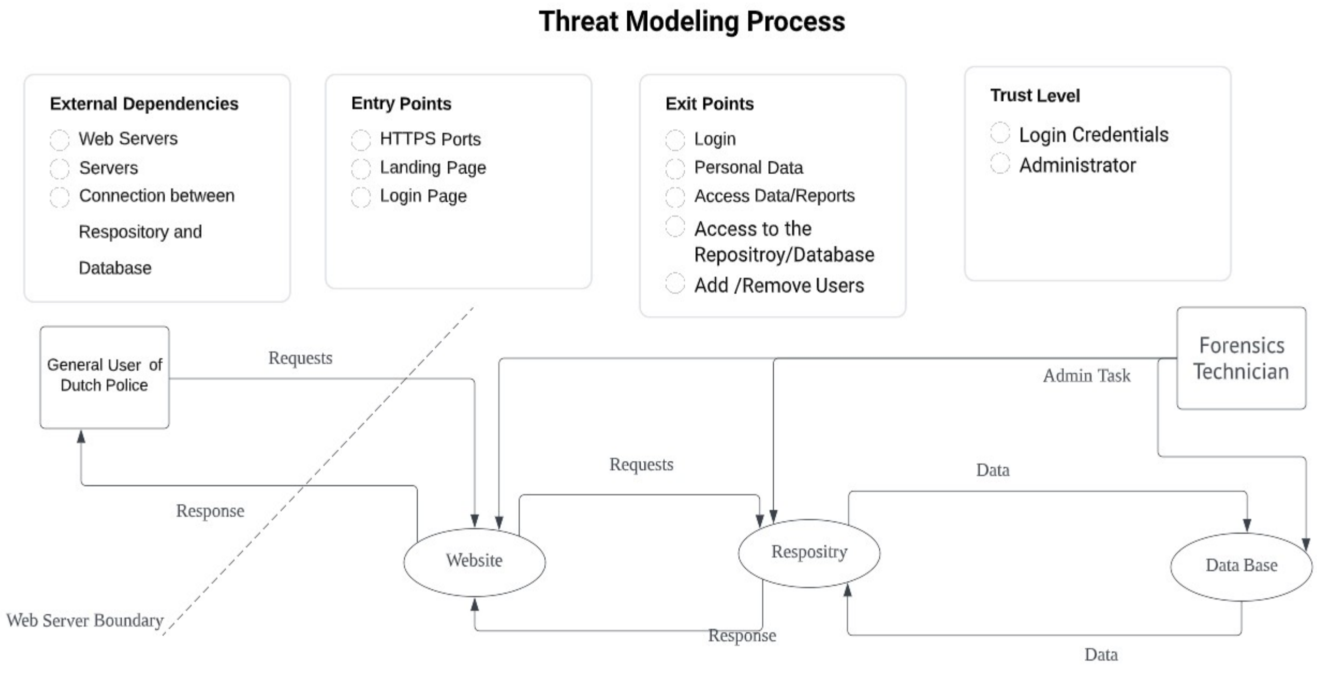

Fig 2 - Data Flow Diagram indicating External Dependencies, EndPoints and Trust levels.

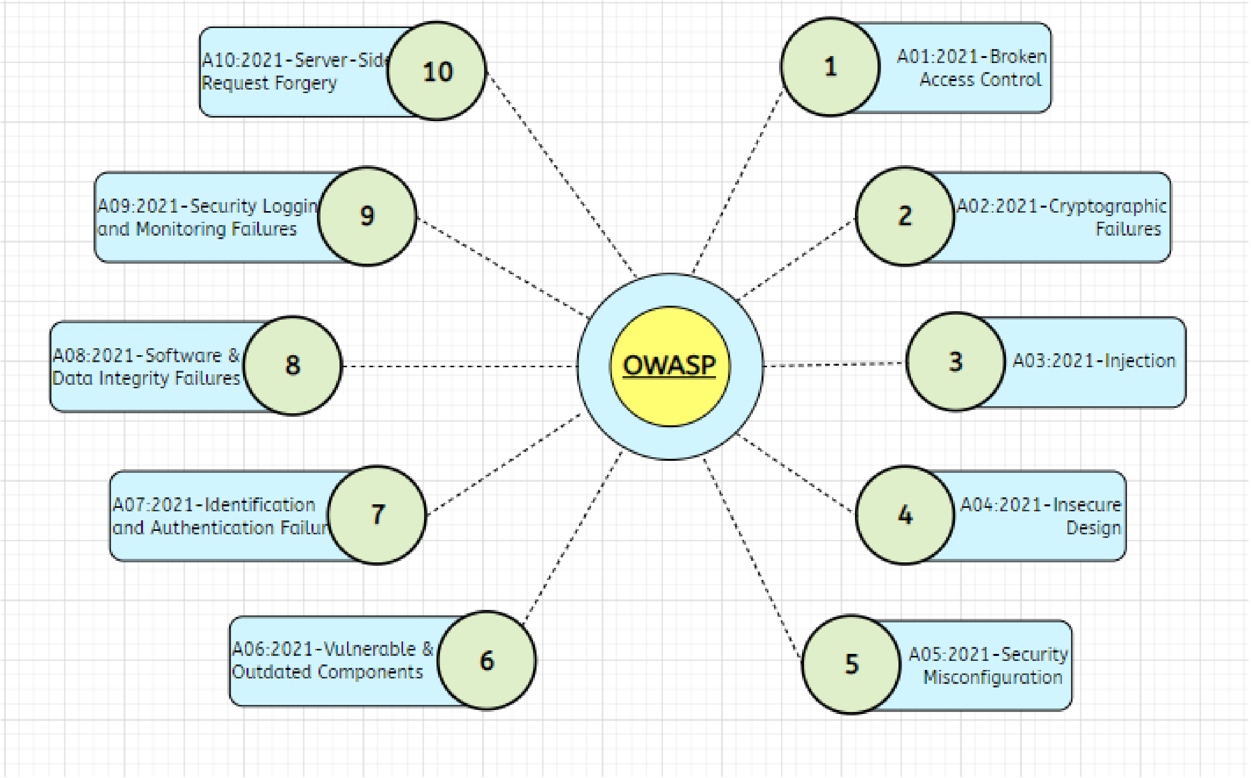
A10:2021-Server-Side Request Forgery
10
1
A01:2021-Broken Access Control
A09:2021-Security Logging and Monitoring Failures
9
2
A02:2021-Cryptographic Failures
A08:2021-Software & Data Integrity Failures
8
OWASP
3
A03:2021-Injection
A07:2021-Identification and Authentication Failures
7
4
A04:2021-Insecure Design
A06:2021-Vulnerable & Outdated Components
6
5
A05:2021-Security Misconfiguration

Fig 3 - OWASP Diagram

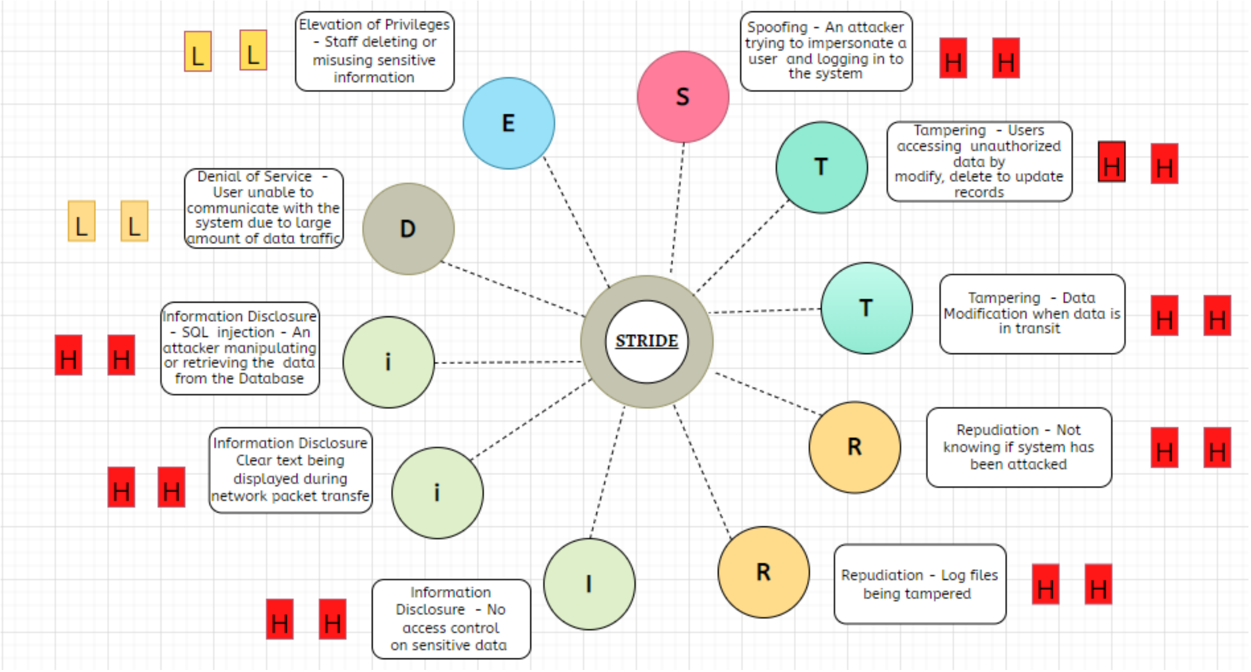
L L
Elevation of Privileges - Staff deleting or misusing sensitive information
E
S
Spoofing - An attacker trying to impersonate a user and logging in to the system
H H
T
Tampering - Users accessing unauthorized data by modify, delete to update records
H H
L L
Denial of Service - User unable to communicate with the system due to large amount of data traffic
D
STRIDE
T
Tampering - Data Modification when data is in transit
H H
H H
Information Disclosure - SQL injection - An attacker manipulating or retrieving the data from the Database
i
R
Repudiation - Not knowing if system has been attacked
H H
H H
Information Disclosure Clear text being displayed during network packet transfe
i
I
R
Repudiation - Log files being tampered
H H
H H
Information Disclosure - No access control on sensitive data

Fig 4 - STRIDE Diagram indicating Probability and Severity of Risk

Mitigation table

STRIDE Reference	OWASP Reference	Mitigation actions
Spoofing	A01:2021-Broken Access Control	● Strong Authentication and Encryption ● Password policy ● Multi Factor Authentication
Tampering	A03:2021-Injection	● Role based access control ● Multi factor authentication
Tampering	A02:2021-Cryptographic Failures	● Ensure data is encrypted using protocols such as the SSL, TLS HTTPS
Repudiation	A09:2021-Security Logging and Monitoring	● Audit logs and production

	Failures	monitoring
Repudiation	A04:2021-Insecure Design	● Ensure the disaster recovery site also maintains backup and replication
Information Disclosure	A02:2021-Cryptographic Failures	● Role based access control
Information Disclosure	A02:2021-Cryptographic Failures	● Secure channel communication
Information Disclosure	A03:2021-Injection	● Access control ● Verification
Denial of Service	A05:2021-Security Misconfiguration	● Network failover recovery ● Increase the network bandwidth ● Create a DOS response plan ● Practise good cyber training

Elevation of Privileges	A01:2021-Broken Access Control	● Ensure all staff have a role based access control, with least privileges as default ● User Audit trails for all tasks

Technical Specifications

Technical Paradigms:

Single Responsibility	(For manageability and easy-to-read)
Open/Closed Principle	(Futureproofing)
Concurrency	(Handling multiple workloads)

Pre-requisite Tools:

Python 3	(Tech stack)
Visual Studio Code	(Integrated Development Environment)
Git for Windows	(Source Control)

Libraries:

Requests Module	(Client listening)
Cachetools Module	(Caching objects)
Pycrypto Module	(Secure cryptography standards)
Pytest Module	(Unit testing)

Endpoint Exposure

Only a single Endpoint will be exposed outside of the Firewall. In this manner, the surface area for attack is minimized.

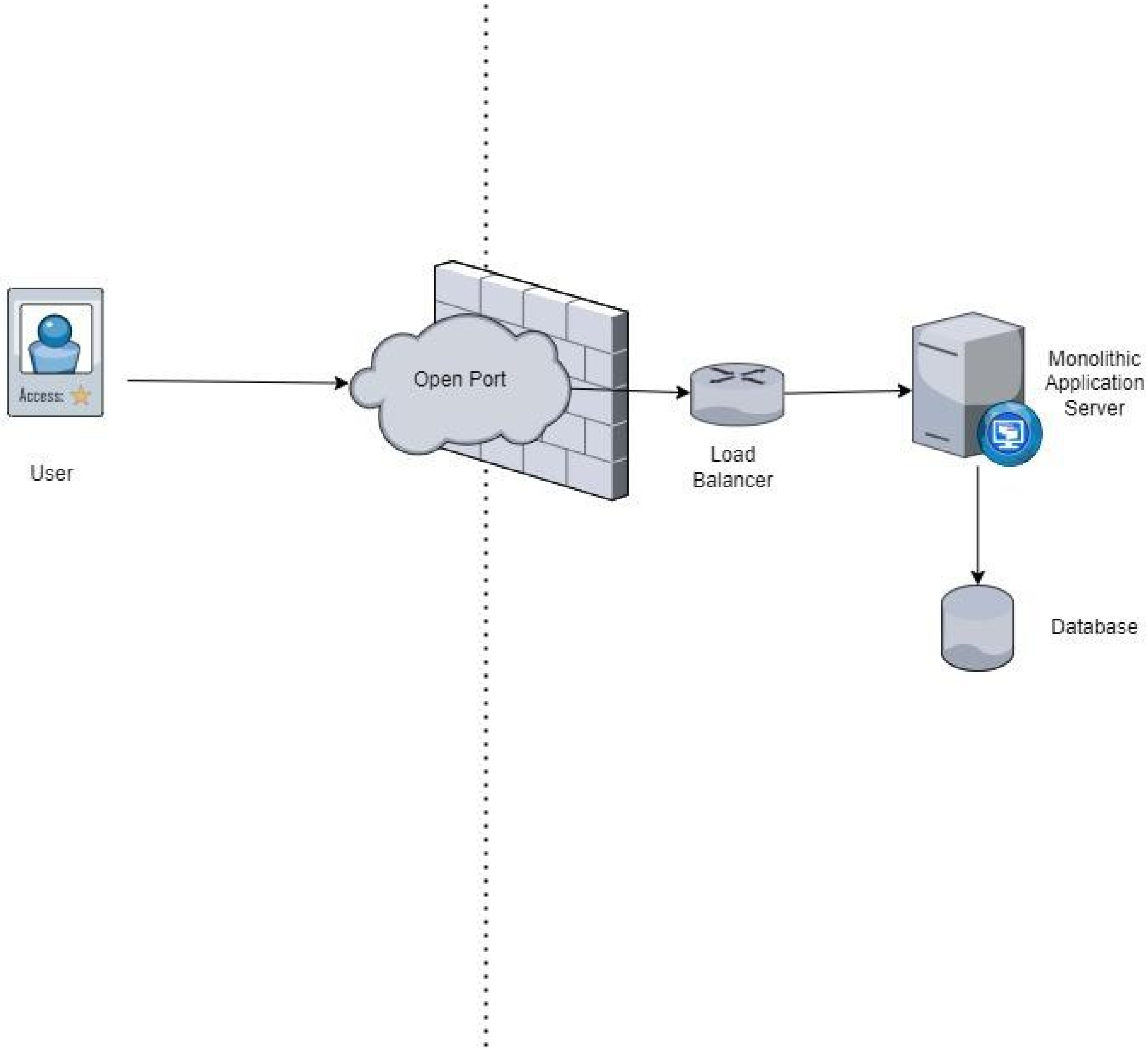

Fig 5 - Network Architecture Diagram

Repository Architecture

An industry best practice approach will be utilized to implement the system’s backend.

All sensitive fields will be Hashed in MD5 or SHA3 (Thiyagarajan, 2008).

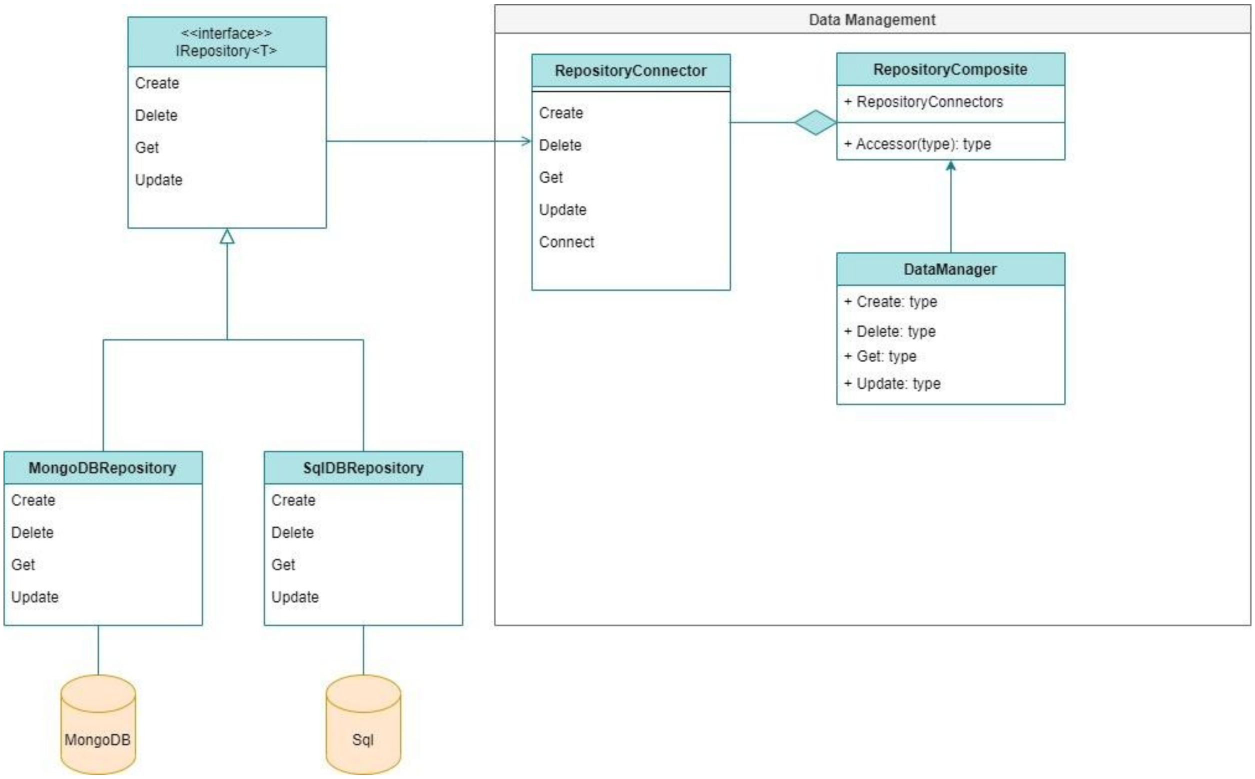

Fig 6 - Class Diagram of Repository

Service Requests

Below is a diagram illustrating the full timeline and lifecycle of an average HttpRequest within the forensics system.

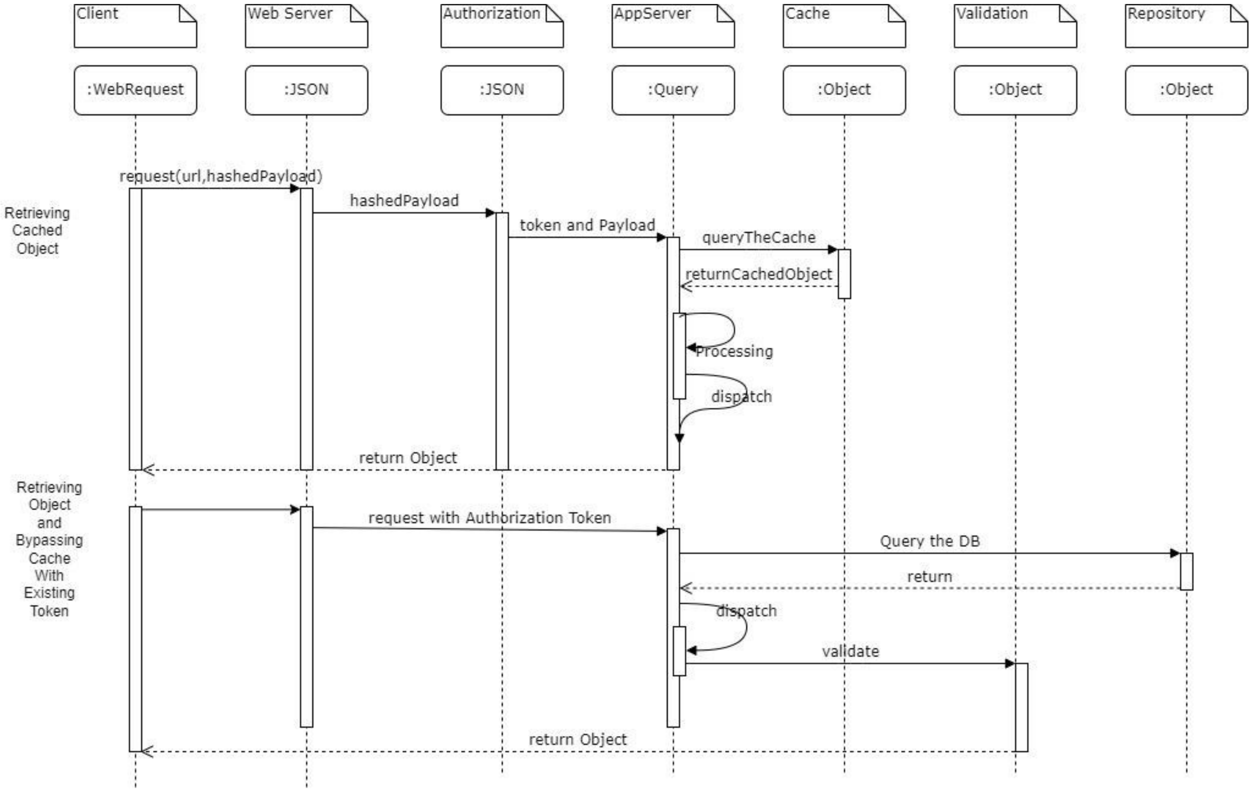

Fig 7 - UML Sequence Diagram of Secure Request

Software Architecture

Figures 8 and 9 illustrate some basic concepts and principles of the intended architecture of the system.

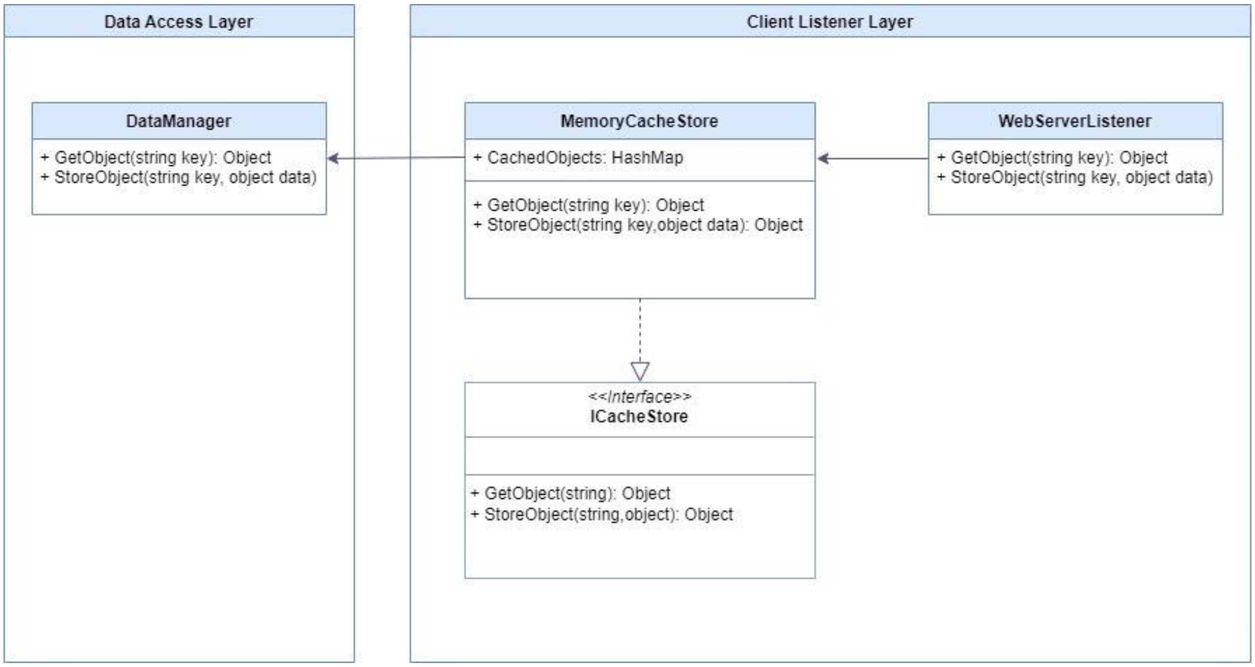
Data Access Layer
DataManager
+ GetObject(string key): Object
+ StoreObject(string key, object data)
Client Listener Layer
MemoryCacheStore
+ CachedObjects: HashMap
+ GetObject(string key): Object
+ StoreObject(string key,object data): Object
WebServerListener
+ GetObject(string key): Object
+ StoreObject(string key, object data)
<<Interface>>
ICacheStore
+ GetObject(string): Object
+ StoreObject(string,object): Object

Fig 8 - Implementation of Cache

Software Development Methodology

A hybrid Software Development Life Cycle methodology will be employed. It will utilize features of both Agile and XP methodologies (Frijns, 2018). Agile caters for dynamic changes of requirements at will, while peer reviews will be used to mitigate against any errors (Thiyagarajan, 2008).

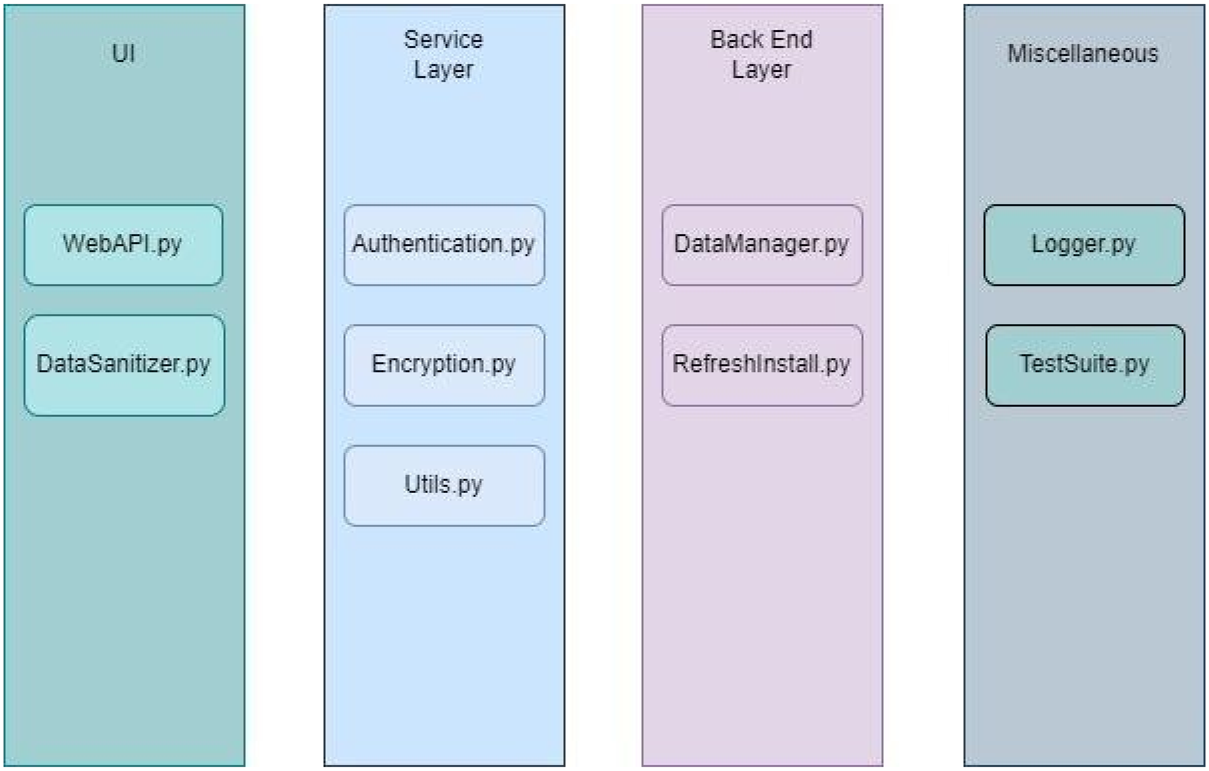

Fig 9 - Example of Software Architecture

References

Bygdås, E., Jaatun, L.A., Antonsen, S.B., Ringen, A. & Eiring, E. (2021) June. Evaluating threat modelling tools: Microsoft TMT versus OWASP Threat Dragon. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)* : 1-7. IEEE.

Desmet, L., Jacobs, B., Piessens, F. & Joosen, W. (2005). Threat modelling for web services based web applications. In *Communications and multimedia security* : 131-144. Springer, Boston, MA.

Frijns, P., Bierwolf, R. & Zijderhand, T. (2018) November. Reframing security in contemporary software development life cycle. In *2018 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*: 230-236. IEEE.

Pittalia, P.P. (2019) A comparative study of hash algorithms in cryptography. *International Journal of Computer Science and Mobile Computing*, *8*(6) : 147-152.

Thiyagarajan, P.S. & Verma, S., (2008) A closer look at extreme programming (XP) with an onsite-offshore model to develop software projects using XP methodology. In *International Conference on Software Engineering Approaches for Offshore and Outsourced Development* :166-180. Springer, Berlin, Heidelberg.