Questions”

Full Brief

For this assessment, you are advised to position yourself and your team as IT Software Consultants and Developers.

You are required to develop an application that provides a **secure repository** for an organisation with domain-specific requirements. A domain refers to a group of users with similar application and hardware requirements. A domain may be characterised by having:

- periodic pressure on resources (CPU, storage, or the network).
- interactive response requirements between request and reply.
- substantial data download requirements.

Such requirements place specific demands on a system’s operations, and this influences the way that it should be operated and managed.

Domains that you should consider for the purpose of this assignment include (see the Unit reading list for the appropriate links):

- the International Space Station (NASA, 2007).
- the Dutch Police Internet Forensics (Government of the Netherlands, n.d.).
- the Hadron Collider at CERN (The Computer Security Team, 2020).

You can read more about the computer requirements of these systems using the references given. You should choose a single domain to focus your development on, and your system should be tailored to the requirements of this domain. In all domains, a user will need to be able to upload, download and share data.

Application requirements:

The agreed criteria for successful development are:

- The solution should ensure that all privacy and security regulations are met, including those specified by the General Data Protection Regulation (GDPR) (ICO, n.d.).
- Mechanisms should be deployed to minimise the attack surface of the solution.
- The organisation would like to see a monolithic version of the application because they have concerns about security, scalability and supportability, and they may wish to extend use to partner organisations on a worldwide basis.

Development conditions: Open-source libraries must be used in your development to ensure that the code may be run for its assessment.

Your full brief is to:

1. Create a comprehensive design proposal report, describing how you will meet the requirements, and the design and implementation of the solution.
2. You should create a working prototype of your design based on a monolithic approach.

Part 1: The Design Proposal Document

Your team is expected to prepare and deliver a design proposal report of your intended development work for the organisation. Note that the associated grading criteria are highlighted in the requirements below, to be reviewed alongside the criteria grid (Module Resources).

Your report should detail the system requirements and assumptions (such as local or remote access, magnitude of storage required, CPU capacity, etc), design decisions (such as encryption algorithms, approach to data storage, use of databases, use of frameworks) and approaches that you have adopted to create your secure software solution. Build up a rationale where appropriate supported by literature **(Knowledge and Understanding weighted at 25%).**

It should list security challenges you have identified/ expect to encounter (such as those in the STRIDE model and/or those forming the OWASP principles). Highlight (briefly) what paradigm(s), pattern(s), theories and practices you intend to utilise on this project to address security, technical and business challenges. Justify your approaches supported by literature. **(Application and Understanding weighted at 25%).**

You should produce a graphical design, based on UML, that illustrates your approach via a number of views which should include, at a minimum, sequence diagrams, class diagrams, and activity diagrams. AND

You should also state any tools, libraries and models that you will use in your solution, justified by academic research. AND

Ensure that your system considers the functionality that similar applications provide, such as user registration, user roles, and CRUD functionality: which should you include? You will need to explain your reasoning and justifications for any omissions during your demonstrations **(Criticality weighted at 25%)**.

Presentation and Structure of your work (weighted at 25%) includes spelling, style, evidence of proofreading, correct use (and format) of citations and references.

It is recommended you use tables and bullet-point lists to stay within the word count limit.

Checklist for the assignment:

- Bulleted list of system requirements and assumptions, design decisions and approaches you will use based on background information and additional academic research (ensure you include any references you have used);
- Bulleted list of security risks/ vulnerabilities you have identified, including reference to frameworks used (e.g. STRIDE, OWASP) with potential mitigations and references;
- UML design of solution using multiple diagrams (e.g. class, sequence, activity) with references;
- Bulleted list of tools (including development and test tools), libraries and models you will use;
- Remember to use a spell checker and proofread your work before submission.
- **You should get your design outline reviewed and approved by your tutor BEFORE you submit the final version in this Unit.** You are invited to arrange a meeting between your team and the tutor to get feedback.

Answer"

Design Document Proposal

Introduction

The following proposal focuses on the development of a secure repository system for NASA. Upon successful implementation of this software, it can be used for the management and tracking of vulnerabilities plus threats along the space shuttle. The system will follow the development techniques standardised by NASA and OWASP. The system also follows general data protection regulations (GDPR) and NIST Security framework. The front end will securely capture User login details and the back end to store Information while maintaining Confidentiality, Availability and Integrity.

System Requirements Assumptions

According to online references, NASA uses IBM AP-101 central processing units in their space systems, coupled with custom-built input/output processors. NASA uses five of these computers (NASA, gov, 2021) in its space systems. For other hardware, NASA (NASA, 2022) has complete data management on its servers with a robust security architecture. These servers

work and operate via cloud flare. Our proposed application is monolithic to run via NASA servers and will contain all the security repositories.

- The application will be built for local access as a Proof of concept (POC).
- All Security controls will be implemented by NASA, adhering to NIST (800, 1800) for cybersecurity and GDPR are implemented and confirmed.
- NASA to evaluate the feasibility of hosting the application as either IaaS or SaaS.
- Storage requirements will start at a minimum and expand with users growth and data logs.

The following are user system requirements:

	Windows requirements	Mac requirements	Linux requirements
Operating system	Windows 8 or later	macOS Sierra 10.12 or later	64-bit Ubuntu 14.04+, Debian 8+, openSUSE 13.3+, or Fedora Linux 24+
Processor	Intel Pentium 4 or later	Intel	Intel Pentium 4 or later
Memory	2 GB minimum, 4 GB recommended		
Screen resolution	1280x1024 or larger		
Application window size	1024x680 or larger		
Internet connection	Required		

Fig 1: User System Requirements: Adopted from support.google.com. (n.d.)

Requirement type	Feature
Functional Requirements	· Allow CRUD operations · Query-able data repository · Manage user data · Manage user access · Make data available on RESTful API · Register users · Security regulations should meet GDPR guidelines · Database security at rest and in transit
Non-Functional Requirements	· Page must load within 2 seconds · Users cannot log in without two-factor authentication · The system must meet WCGA 2.1 guidelines

Fig 2. Functional and non-functional requirements

Tech Stack/ Tools

- Python
- Django
- SQL Database
- OWASP Zap Testing tools

Frameworks

The NIST Special Publication 800-218 framework will be used to ensure Secure Software Design in the Software Development Life Cycle : (Souppaya, Scarfone and Dodson, 2022)

Cryptography libraries in Python like the fernet module will be used. Both Asymmetric encryption (RSA) and Symmetric(AES) will be used in this case to ensure data is secured at rest and in transit. (Franklin, 2020)

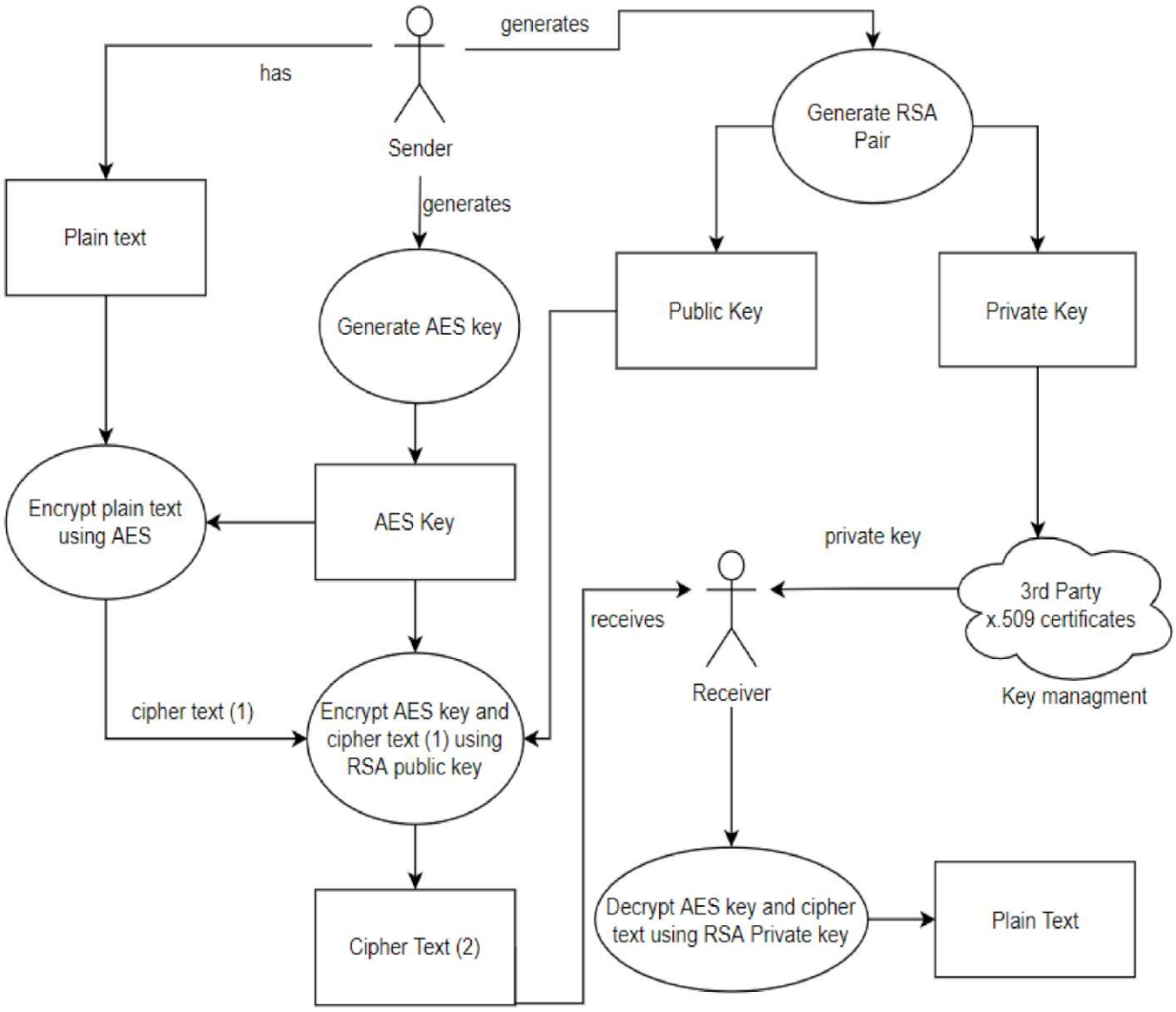


Fig 3: Asymmetric and Symmetric Cryptography; Adopted from (Adee and Mouratidis, 2022)

For our design, we will be using Secured Hashing Algorithm SHA-256 for hashing passwords to be stored in the database,This is due to its improved resistance against brute force attacks (Avast.com, n.d.).

SECURITY CONSTRAINTS

The OWASP top 10 identifies significant vulnerabilities as highlighted below are the security constraints identified

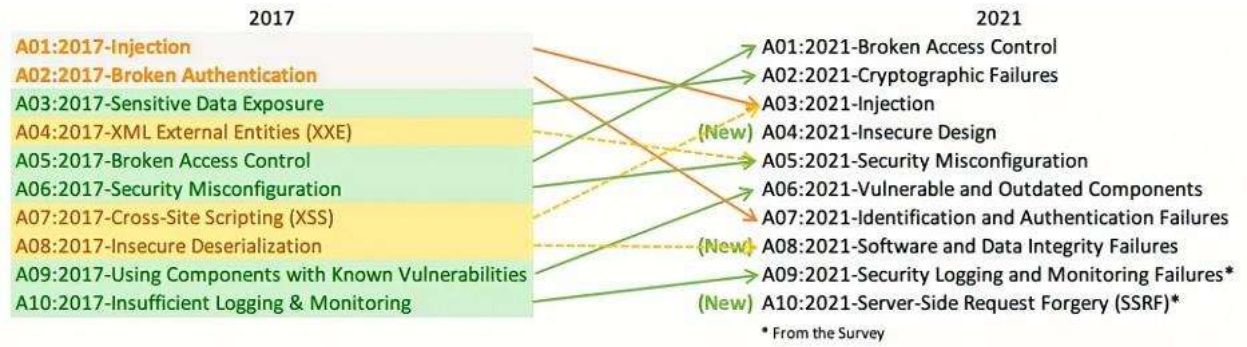


Fig 4 OWASP TOP 10. Adopted from OWASP TOP 10 (OWASP (2021)

S/N	Strategy	How it helps
1	Validate inputs	Inputs should validated especially from sites that are not trusted. This will help defend against injection attacks
2	Simplistic Design	Complex designs might lead to errors and consequently security misconfigurations or insecure designs. Keeping a design simple will help avoid errors
3	Principle of least privilege	Design should adopt the deny by default approach. Least previllages should be considered at all times. The helps prevent against security misconfigurations
4	Clean data	Cleaning data for communications between third-party systems such as databases, command shells, COTs components, third-party middlewares,will reduce the chances of injection attacks
5	Control and Authorise access	Segregation of roles and ensuring proper authentication will help prevent identification and Authentication failures. Also prevents software and data integrity failures
6	Perform effective QA	Comprehensive and regular security testing such as Fuzz testing, penetration testing, and source code audits will prevent monitoring failure, broken access control
7	Practice defense in layers	Multilayered approach to security when coding
8	Define security requirements	Prioritising security in the lifecycle of the design will help identify security constraints. Adopting a risk based approach with regular security checks will ensure vulnerabilities as identified and mitigated
9	Model threats	Use threat modeling like STRIDE to anticipate the threats to which the software will be subjected to
10	Architect and design for security policies	Uniformity should be maintained in the Architecture such that security policies go in line with the various systems of the design.

Fig 5: Security best practices Adopted from (Pillai, 2017)

Threat Type	Mitigation Techniques
Spoofing Identity	1. Control and Authorise Access 2. Principle of least Privilege 3. Simplistic design
Tampering with data	1. Control and Authorise Access 2. Validate Inputs 3. clean Data 2. Defence In Layers 3. Define Security requirements
Repudiation	1. validate Inputs 2. Control and Authorise access 3. Define Security requirements
Information Disclosure	1. Control and Authorise Access 2. Validate Inputs 3. Define Security requirements 4. Simplistic design 5. Defense in Layers
Denial of Service	1. Control and Authorise Access 2. Appropriate authorization 3. Validate Inputs 4. perform Effective Quality Assurance
Elevation of privilege	1. Principle of least Privilege

Fig 6: STRIDE Threat Modelling & Mitigation Techniques

Design Decision and Approaches:

The project emphasis will be on security and usability. According to NASA development guidelines (NASA, 2016), the repository should be compliant with federal regulations and NASA repository security requirements (NASA, 2022).

Detailed System Design

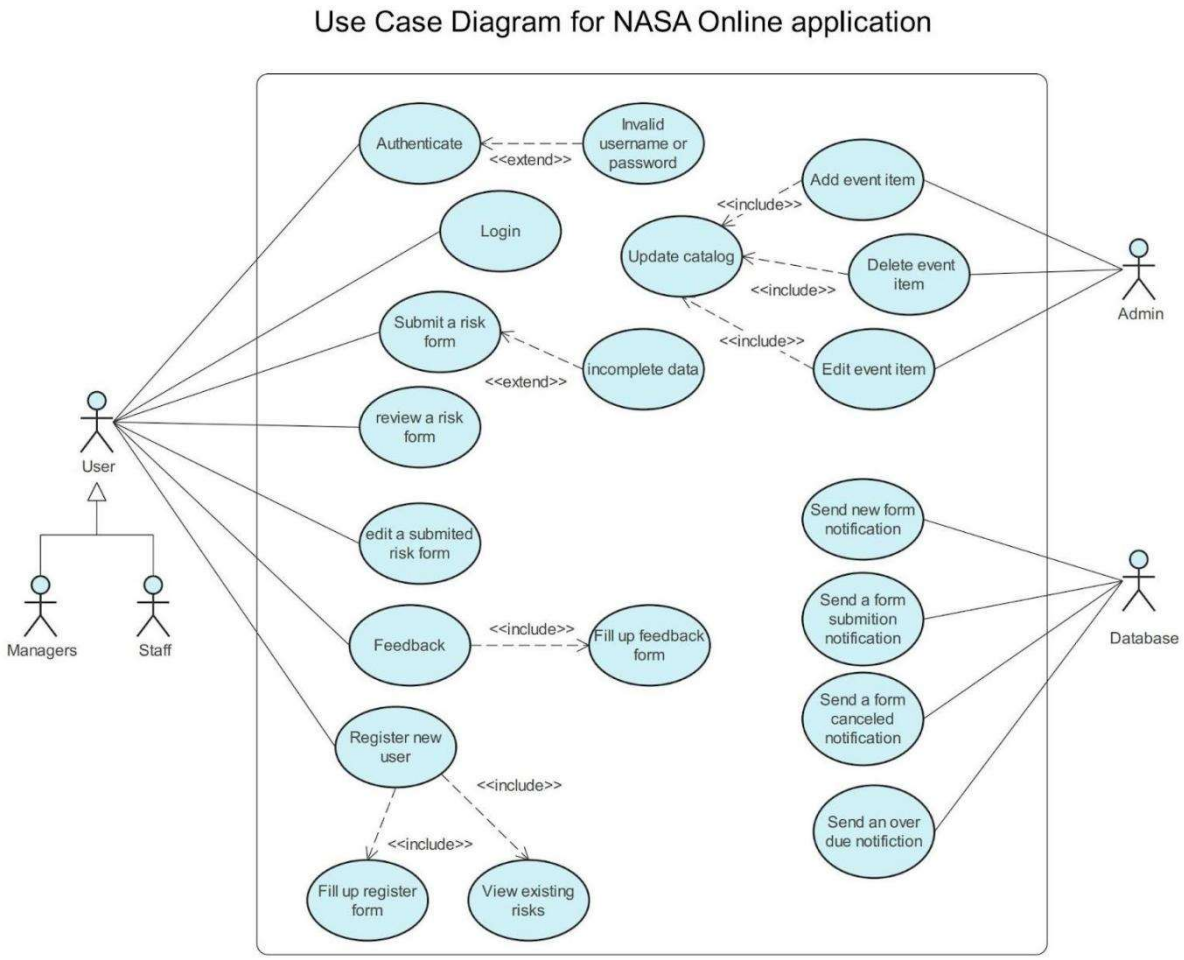


Fig 7: Use case Diagram

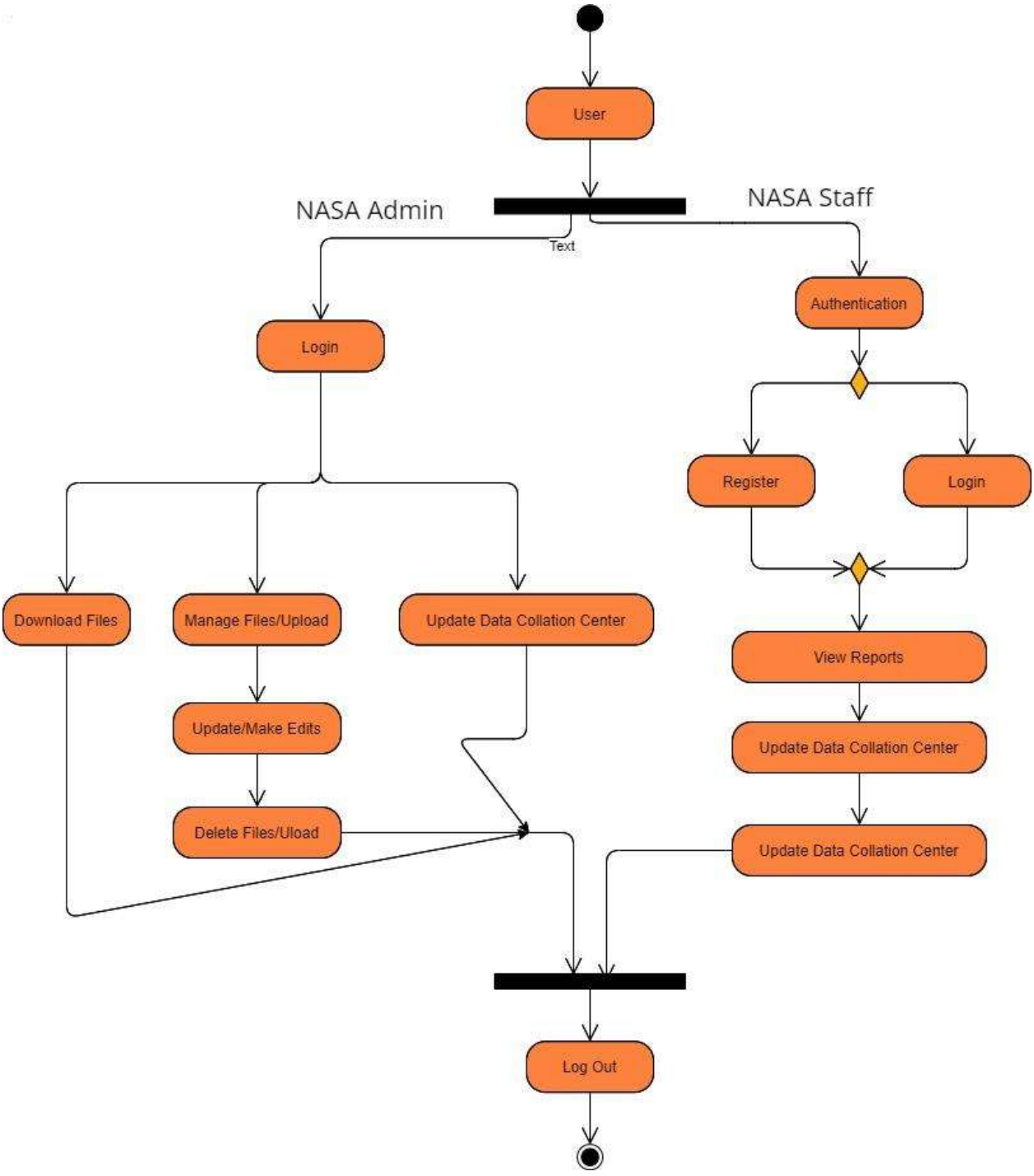
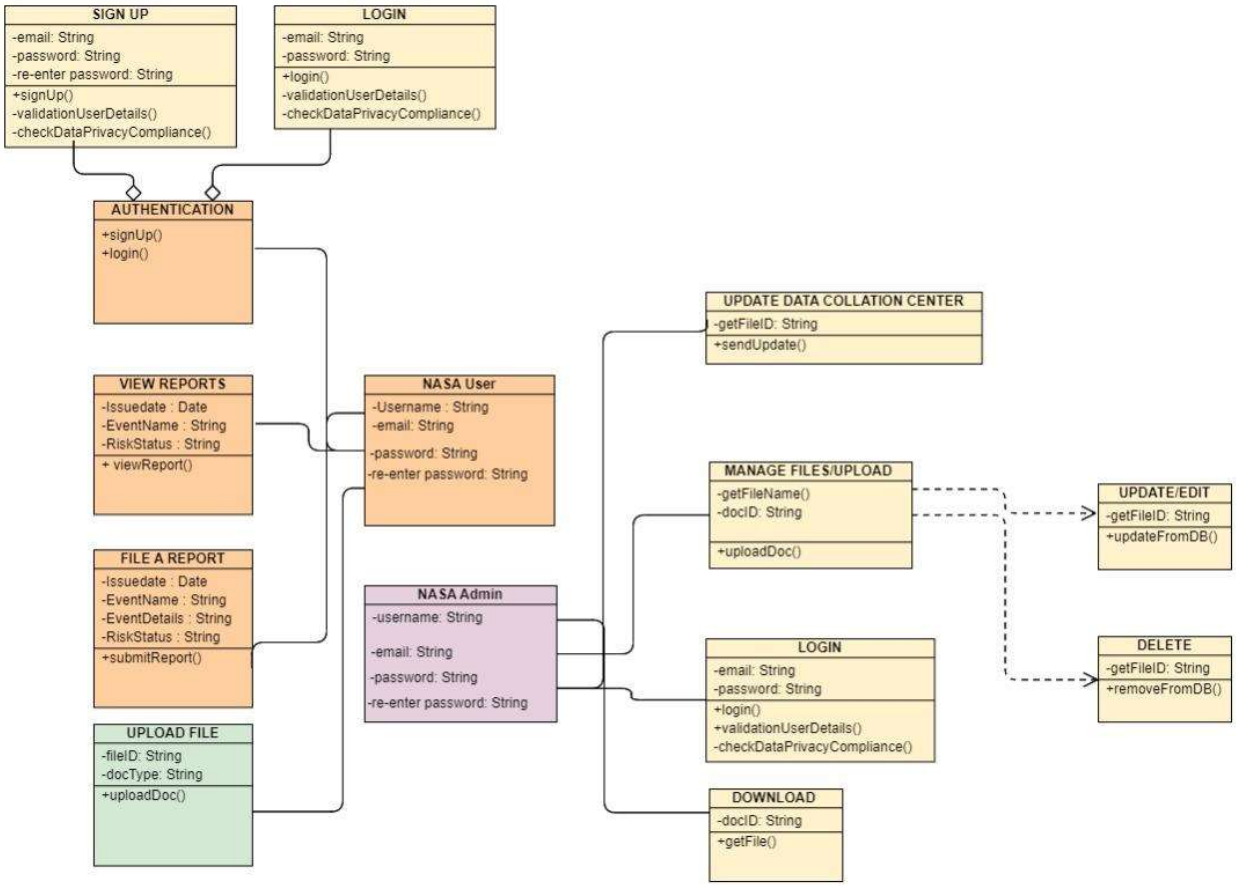


Fig 8: Activity Diagram



Note: to illustrate compliance with GDPR, the + shows public attributes while - private attributes

Fig 9: Class Diagram

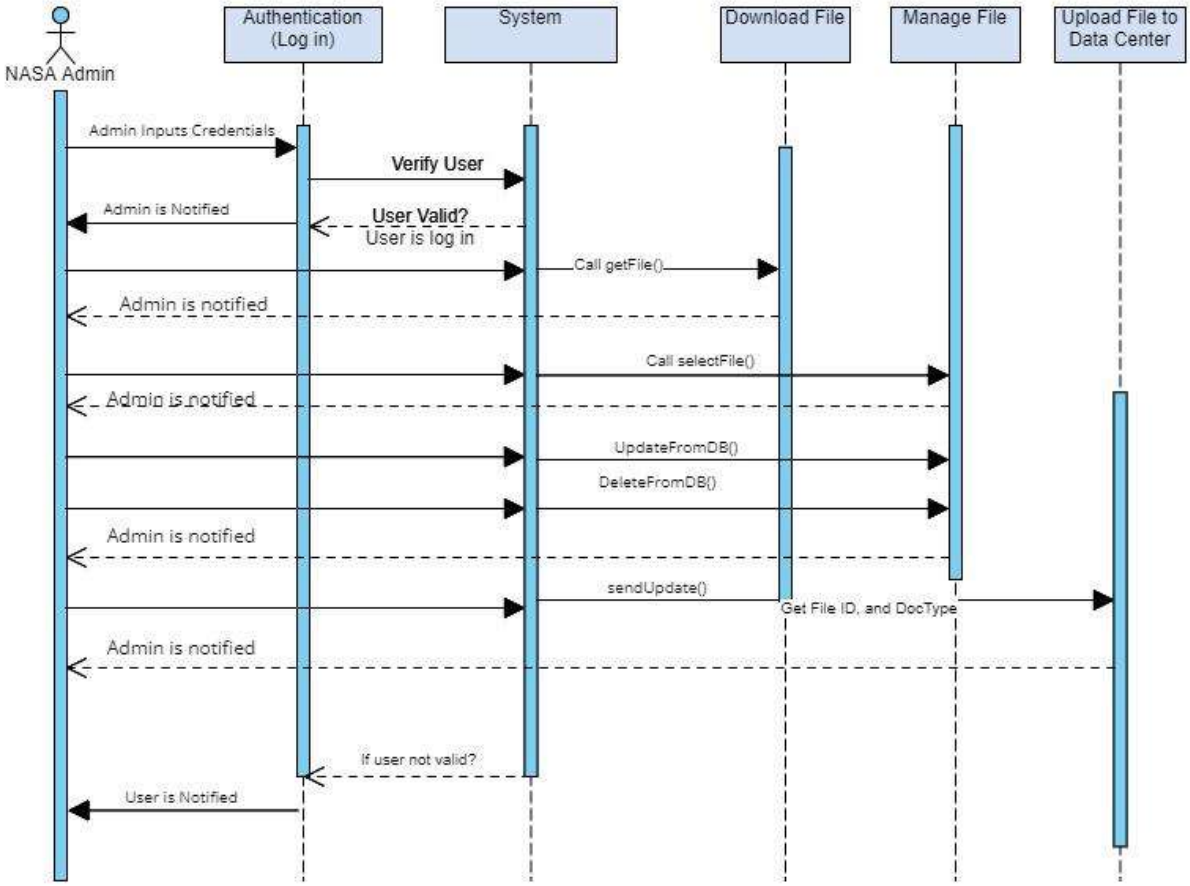


Fig 10: Sequence Diagram for Admin

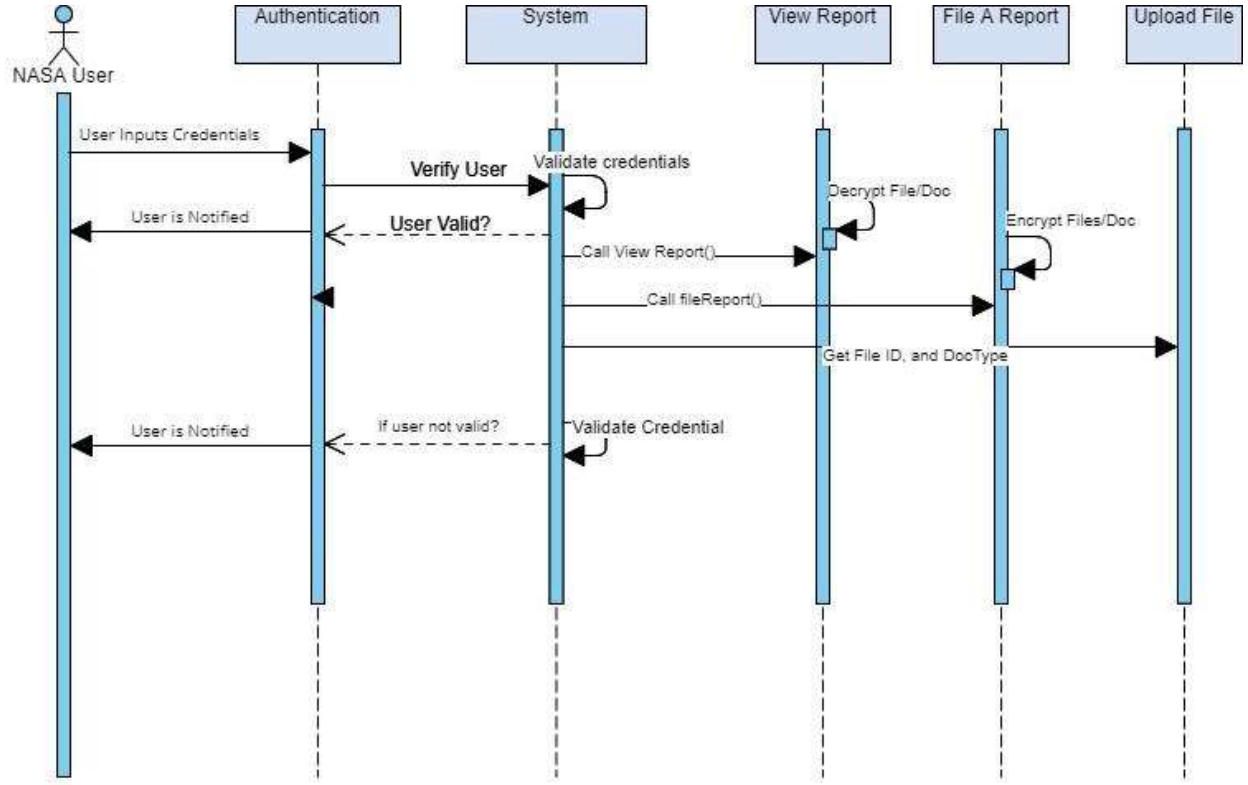


Fig 11: Sequence Diagram for Users

The Web application Architecture Using the Django Framework

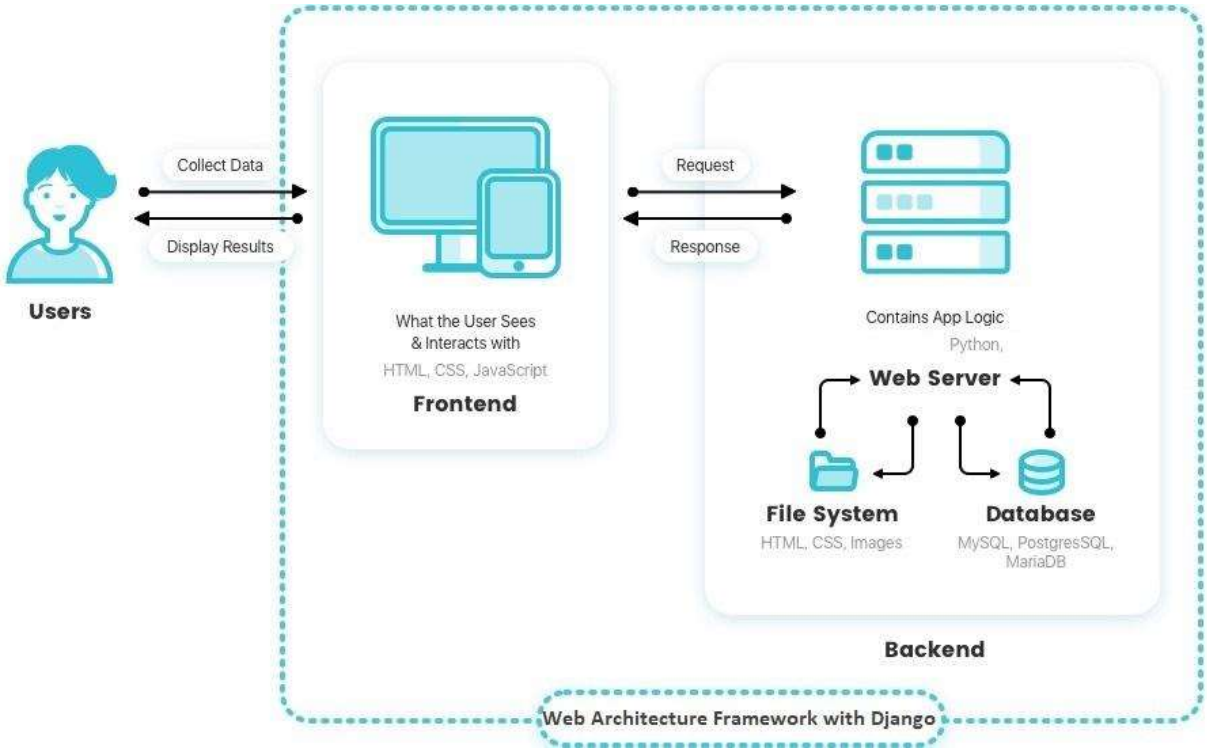


Fig 12: Web app system design Adopted from: Miyawaki, 2020

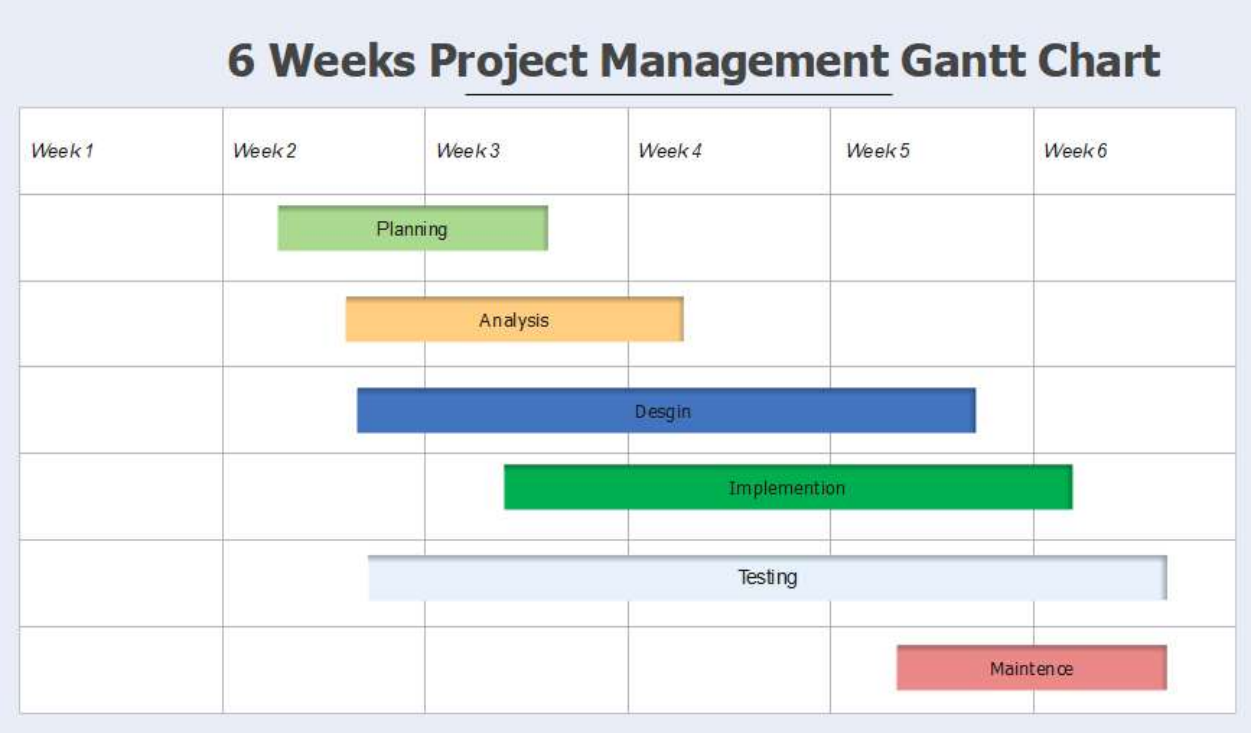


Fig 13: Development phases following secure SDLC

Conclusion

The models mentioned above will be implemented in our system to create and operate the security repository system.This system will be monolithic following the OWASP vulnerability testing and solutions to minimise the attack surface for the solution. The previous diagrams show all the user interaction diagrams for the system to develop a secure application for NASA.

References

Adee, R. and Mouratidis, H. (2022). A Dynamic Four-Step Data Security Model for Data in Cloud Computing Based on Cryptography and Steganography. Sensors, 22(3), p.1109. doi:10.3390/s22031109.

Avast.com, (n.d.). What Is the MD5 Hashing Algorithm and How Does It Work? [online] Available at: https://www.avast.com/c-md5-hashing-algorithm#:~:text=The%20MD5%20hashing%20algorithm%20uses.

Franklin, R. (2020). AES vs. RSA Encryption: What Are the Differences? [online] Precisely. Available at: https://www.precisely.com/blog/data-security/aes-vs-rsa-encryption-differences.

MDN Web Docs. (n.d.). Django introduction. [online] Available at: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction. [Accessed 26 August 2022]

Miyawaki, S. (2020). System Architecture Diagram For Web Application. [online] Available at: https://tovatiz.blogspot.com/2020/04/system-architecture-diagram-for-web.html [Accessed 27 Aug. 2022].

NASA,gov. (2021). *Computers in the space shuttle avionics system*. https://history.nasa.gov/computers/Ch4-3.html#:~:text=NASA%20uses%20five%20general%2Dpurpose,throughout%20the%204Pi%20series29.

NASA. (2016). *Security of Information Technology* (NPR 2810; p. 40). NASA.

NASA. (2022). Software Formal Inspections Standard. *NTSS*. Available from: https://standards.nasa.gov/standard/nasa/nasa-std-87399 [Accessed on 28th August 2022]

NASA. (n.d.). NASA Software Engineering Procedural Requirements, Standards, and Related Resources. *Intelligent Systems Division*. NASA Software Engineering Procedural Requirements, Standards, and Related Resources

OWASP Proactive Controls. [online] Available at: https://owasp.org/www-project-proactive-controls/

OWASP Top 10:2021. [online] owasp.org. Available at: https://owasp.org/Top10/

Pillai, A.B. (2017) Software Architecture with Python : Architect and Design Highly Scalable, Robust, Clean, and Highly Performant Applications in Python. Birmingham, UK: Packt Publishing. Available at: https://search-ebscohost-com.uniessexlib.idm.oclc.org/login.aspx?direct=true&db=nlebk&AN=1513359&site=ehost-live (Accessed: 27 August 2022).

Souppaya, M., Scarfone, K. and Dodson, D. (2022). Secure Software Development Framework (SSDF) Version 1.1: (Draft). [online] doi:10.6028/nist.sp.800-218 [Accessed 27 August 2022]

support.google.com. (n.d.). System requirements - Google Web Designer Help. [online] Available at: https://support.google.com/webdesigner/answer/3232604?hl=en.