Data Transfer, Storage and Communication techniques, a feasibility analysis

In order to secure the trefacta of cybersecurity, our architecture will abide by the following principles:

- *Principle of Least Privilege*

Different authentication levels will exist, so users will be allowed to view/upload/remove only files in their competence. (e.g. admins should be able to remove entries, creators may supply new data and simple users may only view).

- *Principle of Separation of Duties*

The keyword here is: **distribution**. The architecture is fundamentally built in 3 different layers: client, gateway and services. The client's main concern is to offer a way for users to visualise and interact with the data, the gateway will be responsible for directing data between endpoints, finally the services will handle every data operation; such services include: a *CassandraDB* distributed instance for data storage and an ad-hoc authentication system that supports military-grade encryption.

- *Principle of Failing Securely*

Unexpected erorrs are a continuous threat to data integrity, security and confidentiality, therefore mitigation should start in-design. The best way to ensure no data leak could possibly be present, we introduce the idea of a safe fail: whenever the data's chain of custody cannot be verified, or communication from services is being obstacled, the system will provide no data to the endpoint, instead redirecting to a default state (home page / sign-in page).

- *Principle of Open Design*

When it comes to cryptography, the strength and safety of algorithms comes from being open, with no "secret ingredient" or hidden backdoors. We intend to design our encryption system using best practices, including (but not limited to): SHA256 encryption algorithm, salted credentials, 2FA by default, HTTPS-only communication, blockchain-based storage, fault-tolerance and redundancy by design.

- *Principle of Minimal Attack Area*

The fundamental principle of a good defense, is to know your weak spot and defend it against attacks; this becomes exponentially easier if you have very few points of interaction: for instance, databases cannot be queried manually, only by providing authentication tokens on the client, same goes for the middleware. Additionally, registrations and permissions are handled and

approved server-side, as to avoid infiltrations, and an additional security mechanism, in having files being password protected on download, can be additionally included if deemed necessary.

Proposed Architecture:

- CassandraDB for Datastorage
- SHA256 with Salt + Pepper + Hash for encryption
- t3 Tech Stack for web components and middleware
- JWT-based sessions + HTTPS for safe and reliable data transfer