IDEA-FAST

Identifying Digital Endpoints to Assess FAtigue, Sleep and acTivities in daily living in Neurodegenerative disorders and Immune-mediated inflammatory diseases.

Grant Agreement No. 853981

WP5 – Data Management

D5.2: Data Management Platform Specification

Lead contributor	P9 - ICL
Other contributors	P20 - PLUR P35 - Janssen P44 - SARD

Due date	31 Oct 2020
Delivery date	30 Oct 2020
Deliverable type	R
Dissemination level	PU

Document History

Version	**Date**	**Description**
V0.1	10 Oct 2020	First Draft
V0.2	21 Oct 2020	Additional Content
V0.3	26 Oct 2020	Additional Content
V0.4	27 Oct 2020	Additional Content
V1.0	30 Oct 2020	Final Version

PARKINSON'S UK
CHANGE ATTITUDES.
FIND A CURE.
JOIN US.

Table of Contents

1 Abstract

This document details the specification of the IDEA-FAST Data Management Platform (DMP). It defines the functionality requirements of the DMP, with detailed explanations on the DMP components including Data Storage and Sharing, Authentication, Authorisation and User Management, APIs and Analytical Environment. It also describes the technical specifications of the underlying infrastructure of the DMP. System security features and requirements are also outlined.

2 Introduction

The IDEA-FAST project aims to identify novel digital endpoints for fatigue and sleep disturbances that will provide more reliable, objective and sensitive evaluation of the severity and impact of these symptoms in the person's normal surrounding. Data on fatigue, sleep, selected activities of daily living and other relevant contextual information will be collected during the project, including pseudonymised participant demographic, clinical and patient-reported outcome data, device data from participants during the periods of device use, as well as data generated from analyses of the clinical and/or device datasets. Such datasets are expected to provide an invaluable resource for clinical practice, well-being of the patients, drug development, healthcare provision, health and wealth of the society and research and innovation.

It is essential to have a shared work environment where all the project data are curated to high quality, integrated for analysis and robustly managed with security and privacy assurance. The IDEA-FAST DMP is being designed and developed to address these needs. All data generated in the IDEA-FAST project will be integrated and stored on the DMP using a secure and robust process that is fully compliant with European data privacy and security legislation and standards, in particular the General Data Protection Regulation (GDPR).

This document details the specification of the IDEA-FAST DMP. Each section defines a set of requirements that describe the functionality of the DMP as well as the current and future development of the DMP to meet these requirements, include:

- **Infrastructure and General Environment**, where technical specifications of the DMP's underlying infrastructure are described, along with the infrastructure strategy and security rationale (section 3);
- **Data Storage and Sharing,** where we give an overview of the IDEA-FAST data flow, as well as the functionalities of the DMP in terms of data integration, storage, sharing and exploration (section 4);
- **Authentication, Authorisation and User Management**, where mechanisms used for Identity and Access Management are described (section 5);
- **Application Programming Interfaces (APIs) Specifications**, where APIs for authentication, and file transfer are defined (section 6);
- **Analytical Environment,** where we describe the architecture and components of the DMP's analytical environment (section 7);
- **Security and Regulatory**, where the security features and requirements are defined, along with the proposed approach to cybersecurity (section 8).

All the above components form a coherent platform for large-scale healthcare and medical data management and analysis. The production version of the DMP is deployed at https://data.ideafast.eu. The latest version of the DMP (as of October 2020) is v0.6.0. Documentation and user manuals of the DMP will be publicly accessible from the project GitHub repository[1], as well as the source code for the software developed during the project.

[1] https://github.com/ideafast

3 Infrastructure and General Environment

3.1 Physical Environment

As part of its plan to host IDEA-FAST data for the duration of the project and beyond, Imperial College London (ICL) will be utilising equipment and tools it owns, located in an ISO 9000 certified and ISO 27001 compliant data centre. The physical security, facility maintenance and emergency operations are run by “VIRTUS Data Centres”. The data centre caged pods utilised are located at: VIRTUS LONDON4 Slough Campus, Data Hall 3, 13 Liverpool Road, Slough, SL1 4QZ, United Kingdom.

The list of data centre certifications includes:

- BS EN ISO 9001:2015 – Quality Management
- BS EN ISO 14001:2015 – Environmental Management
- ISO/IEC 27001:2013 – Information Security Management
- BS EN ISO 50001:2011 – Energy Management
- ISO/IEC 20000-1:2011 - Service Management
- ISAE3402 Certification compliance
- PCI DSS v3.2 Compliant
- BREEAM Excellent
- Uptime Institute - Management & Operations

3.2 Execution Environment

The IDEA-FAST DMP is designed to be run in cloud-native environment in a way that it can be made resistant to environmental perturbations. To support WP5, will use a mutualised cloud installation based on Canonical MAAS and OpenStack.

The underlying infrastructure of the DMP uses reproducible deployments by employing metal-as-a-service provisioning for the installation of its OpenStack components (Table 1). This allows for a zero-touch centralised configuration of the various pieces of software used to power the platform.

Table 1: A list of the components used in the underlying infrastructure of the DMP.

Component	Description
aodh	AODH is the Alarming service, it enables the ability to trigger actions based on defined rules against metric or event data collected by Ceilometer or Gnocchi.
ceilometer	Ceilometer is a data collection service that provides the ability to normalise and transform data across all OpenStack core components it is a component of the Telemetry project. Its data can be used to provide resource tracking and alarming capabilities across all OpenStack core components.
ceilometer-agent	The ceilometer agent is a data gathering component that is executed alongside computational workloads to provide monitoring metrics.
ceph-mon	The Ceph monitor is the cluster monitor daemon for the Ceph distributed file system. One or more instances of ceph-mon form a Paxos part-time parliament cluster that provides extremely reliable and durable storage of cluster membership, configuration, and state. It provides management and health status checks for the block and object storage underlying Ceph deployment.
ceph-osd	The CEPH OSD is the object storage daemon for the Ceph distributed file system. It is responsible for storing objects on a local file system and providing access to them over the network.
ceph-radosgw	A utility for interacting with a Ceph object storage cluster (RADOS), part of the Ceph distributed storage system.
cinder-ceph	This is a bridge connector for using Ceph in Cinder environments.
cinder	Cinder is the OpenStack Block Storage service for providing volumes to Nova virtual machines, Ironic bare metal hosts, containers and more.
glance	Image service project which provides a service where users can upload and discover data assets that are meant to be used with other services.
gnocchi	Operational data integration and analysis tooling.
horizon	Web-based management user interface provider.
keystone	Service that provides API client authentication, service discovery, and distributed multi-tenant authorization.
keystone-ldap-ict	Connector for LDAP support in Keystone.
memcached	An in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.
mysql-innodb-cluster	InnoDB is a storage engine for the database management system supported for MySQL.
neutron-api	Controller service to provide “network connectivity as a service” between interface devices in cloud fabric.
neutron-api-plugin-ovn	Neutron driver plugin for Open Virtual Network in Linux kernel.
nova-cloud-controller	Provider for compute instances management. Nova supports creating virtual machines, it is not planned to support IDEA-FAST ontop baremetal servers as it would require Ironic but MAAS is already used.
nova-compute	Provider for compute instances libvirt engine and execution.
ntp	Time synchronisation daemon.
ovn-central	Service for management software defined network layout in Open Virtual Network environments.
ovn-chassis	Compute instance connector agent for Open Virtual Network.
placement	Service responsible for tracking inventory of resources available in a cloud and assisting in choosing which provider of those resources will be used when creating a virtual machine.
rabbitmq-server	Messaging service to broker communication across all other infrastructure services.
vault	Software enclave for securing, storing and tightly controlling access to tokens, passwords, certificates, encryption keys for protecting secrets and other sensitive data.

3.3 Infrastructure Strategy

In order to minimise maintenance cost in terms of person-resource, the underlying infrastructure of the IDEA-FAST DMP is implemented in autonomous deployment. The DMP utilises Canonical MAAS, which is driven by Canonical Juju, to orchestrate the deployment of a fully high-availability capable OpenStack Infrastructure.

High availability of the Juju management controller is enabled, and all configuration of OpenStack services is operated via Juju. Operation data are stored in a three-node MongoDB cluster.

The OpenStack infrastructure features a replicated 4 instances MySQL cluster for storage of operation data.

3.4 Security Rationale

In order to ensure maximum security as well as drive compliance up, servers are deployed with dynamically generated SSH keys and hardened prior to being placed online, rendering it impossible for even the infrastructure administrator to access the servers.

To prevent physical attack on the server and data exfiltration from hard drives, all systems are already deployed with LUKS locking full drive encryption. Token for the encryption are generated dynamically, one per drive and stored in a multi-component Shamir key secured Vault with rotating master. The Vault is instantiated in-memory and requires 3 components of the key to be restarted or recovered from outage.

Large block and object storage are secured in a similar fashion for data safety powered by a three-tier data replication across multiple rack in our data hall. Note this means that IDEA-FAST data is not protected against area-wide natural disaster.

All inter-service communication is done over TLS v1.3 using rotating certificates provided dynamically by the Vault instance. The Vault instance derives its signing right from an Intermediate X509 certificate from the Data Science Institute of ICL. The X509 certificate itself derives from a 10-year valid CA Root certificate from the Data Science Institute of ICL.

4 Data Storage and Sharing

The DMP will provide secure large-scale data storage and data sharing environment to host data generated in the IDEA-FAST project as well as extant datasets provided by the academic and EFPIA partners. Such environment is essential to ensure data conforms to the FAIR (findable, accessible, interoperable and reusable) data principles. Data managed by the DMP will be mirrored at all times to prevent accidental loss. A snapshot of the DMP will be created and achieved on a bi-weekly basis for backup purpose.

Diverse datasets which are expected to be stored and shared by the DMP, include:

- Pseudonymised participant demographic, clinical and patient-reported outcome data, generated from the Feasibility Study and the Clinical Validation Study;
- Device data from participants during the periods of device use in the Feasibility Study and the Clinical Validation Study;
- Relevant extant datasets shared by the academic and EFPIA partners.

4.1 Data Integration

The flow of IDEA-FAST data is shown in Figure 1. Clinical data generated in the project will be first captured in the clinical database/eCRF system managed by the University of Cambridge (UCAM) in WP2, and then transferred to the DMP via secure APIs. Device data generated in the project will be first stored in device software (or application on a study smartphone), and then transferred to the DMP via secure APIs. The estimated size of data collected during the project is less than 10GB per participant, including clinical data and device data.

The DMP will provide functionality for data integration, sharing, exploration and analysis. Clinical and device data transferred to the DMP will be standardised against the IDEA-FAST clinical data standards and device data standards, which consist of metadata templates, control vocabularies and data dictionaries, and will be designed and delivered by Month 24 of the IDEA-FAST project. The standardised data will then be integrated across different domains (e.g. demographics, diagnosis, laboratory tests, medications), across studies (e.g. Feasibility Study and Clinical Validation Study) and across multiple digital devices. The integrated data will then be saved in the storage component/layer of the DMP for data sharing and analysis.

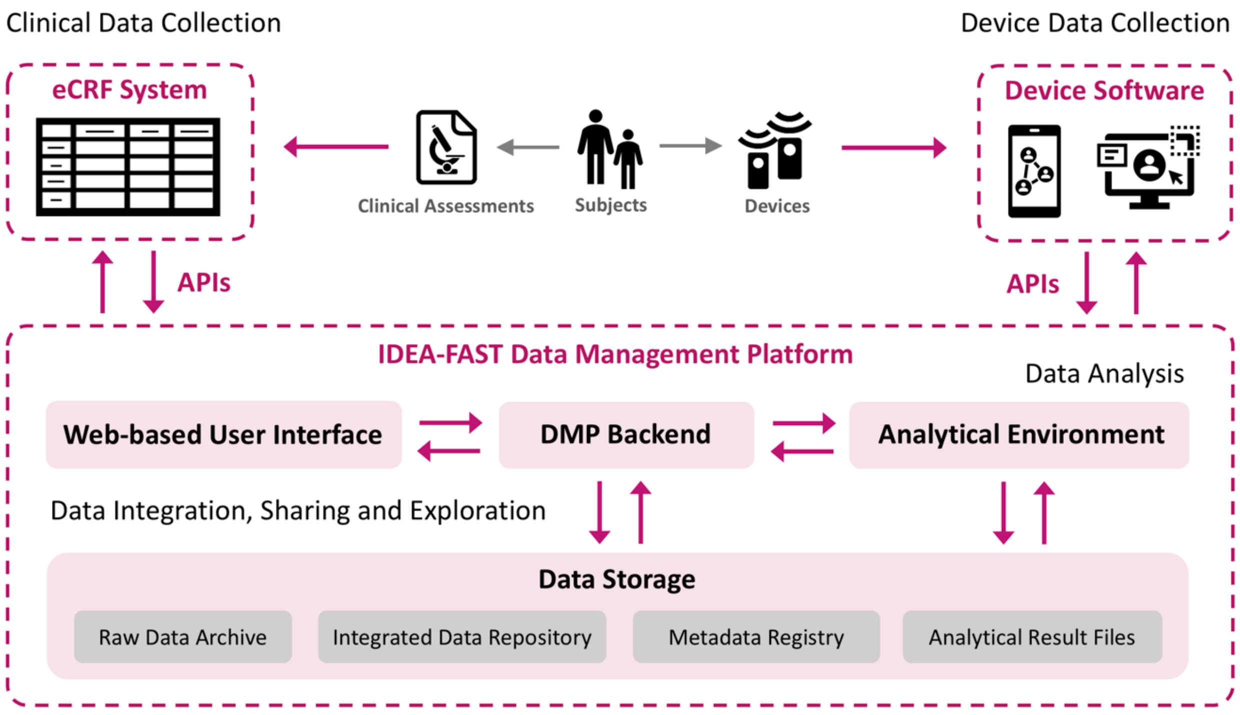


Figure 1: Data flow of the IDEA-FAST datasets and major components of the DMP.

4.2 Data Storage

There are four major components in the DMP (Figure 1): the web-based user interface (UI) which allows users to interact with DMP via a web browser; the DMP backend which receives and processes requests; the storage where data is saved and stored; and the analytical environment which enables users to run their analysis on the DMP. The storage component/layer of the DMP provides reliable, scalable data storage by using NoSQL database MongoDB and S3 compatible high-performance object storage.

The following data forms will be managed in the DMP:

- **Raw data**, which refers to the data captured in the UCAM clinical database and device software/applications without processing;
- **Integrated data**, which refers to the dataset that processed and standardised clinical and device data are integrated into a common model for querying and analysis;
- **Metadata**, which refers to the data that describes the IDEA-FAST clinical data and device data (e.g., properties of a dataset or a variable);
- **Analytical results**, which refers to the data generated from analyses of one or both of the above datasets.

Clear version numbers of the datasets will be provided through the DMP. Change logs that contain lists of notable changes for each version of the datasets will be automatedly generated to track changes.

4.3 Data Sharing and Exploration

The DMP will provide a web-based UI with secure access control that allows users to access the pseudonymised clinical and device data for data exploration and analysis. Keyword-based search functions are provided via the UI to improve data findability and reusability. The implementation of the UI makes use of React, which is a component-based JavaScript library for interactive UI development. Data querying and manipulation are performed via GraphQL.

It is a fundamental premise that no data shall be accessible to any party, internal or external, without full ethical sanction, and that only pseudonymised data will be shared within the Consortium. Identity management and access control mechanisms are in place to enhance security and ensure compliance (Section 5).

The DMP also allows users to upload data files to a study hosted on the DMP via the UI or APIs (Section 6.3 and Appendix A). Users can also perform integrated data modelling and analysis to identify novel digital endpoints via the Analytical Environment of the DMP (Section 7). An automated audit trail functionality tracks all queries/analyses performed against the DMP for provenance and accountability.

5 Authentication, Authorisation and User Management

The datasets generated in the IDEA-FAST project and the extant datasets provided by the academic and EFPIA partners contain sensitive information (e.g. a participant’s clinical records). To enhance data security and increase data privacy protection, identity management and access control mechanisms need to be in place to protect sensitive datasets from breaches and unauthorised access.

The IDEA-FAST DMP is designed to provide efficient and secure mechanisms to cope with sophisticated data privacy legislations and requirements for clinical data management, in particular to comply with the GDPR. Authentication and authorisation mechanisms are being designed and implemented to verify users’ identity and manage access rights and privileges to data resources. A brief explanation and comparison of authentication and authorisation are summarised in Table 2.

Table 2: Summary of authentication and authorisation mechanisms.

	Authentication	**Authorisation**
Purpose	Verifies users' identity.	Validates users' access permissions to resources.
Order of the process	Authentication is performed at the very first step when a user tries to access the DMP, before authorisation.	Authorisation is performed when a user requests to access resources managed by the DMP, after authentication.
Techniques	Examples: • Password-based authentication • Two-factor/multi-factor authentication • Captcha test • Biometric authentication	Examples: • Open Authorisation (OAuth) • Role-based access controls • Token-based authorisation • Access Control Lists

5.1 Authentication Mechanisms

Authentication is the process of validating a user's identity in order to permit access to the DMP. During the process, the user's credentials (such as username/user ID and password) are checked and verified by the DMP. In DMP v0.6.0, a Two-Factor Authentication (2FA) mechanism has been implemented to secure access to project datasets:

- **Password-based authentication:** anyone who would like to access the DMP is required to register his or her credentials (username, password, email address, first/last name and organisation) with the DMP. The username and password need to be provided every time when a registered user wants to access the DMP for authentication.

  Users' account information is stored in the DMP and protected by advanced cryptography techniques. For example, cryptographic salts are used to safeguard passwords in storage. A random string of characters (i.e. a cryptographic salt) is added to each password before the password is hashed, which produces a unique hash to be stored in the DMP. These techniques can protect the DMP against different types attacks, such as rainbow table attacks, and slow down dictionary and brute-force attacks.

- **2FA:** a registered user needs to provide two different authentication factors (password and a 6-digit Time-based One-time Password, TOTP) to verify his or her identify when accessing the DMP. 2FA adds an additional layer of security to the authentication process as it prevents unauthorised users from gaining access to an account with a stolen password.

  The TOTP used during the authentication process is generated from a TOTP algorithm that complies with Internet Engineering Task Force (IETF) standard RFC 6238. Once a user has completed the registration form, a confirmation email will be sent to the user with a Base32 secret key. The user can then use a standard authentication application (e.g., Google Authenticator and Microsoft Authenticator) to register the key for generating TOTPs (also see Section 8.1 for details).

- **API authentication:** the DMP allows registered users to access the platform via a web interface (data.ideafast.eu) and GraphQL-based APIs. The API authentication mechanism allows users to verify their identities in a programmatic manner. In DMP v0.6.0, an HTTP Basic authentication mechanism is used, in which a registered user needs to provide his or her username, password and the TOTP in a login request for authentication (see Section 6.1 for details).

In future versions of the DMP, a public-key authentication mechanism will be designed and implemented for better interaction with other IDEA-FAST related systems and applications. In particular, a public-key management mechanism will be developed to handle public-key credentials of registered users and applications, and a digital signature verification function will be implemented to verify the requester's identity. A web-based key generation service will also be provided by the DMP with which users can generate key pairs on the client-side. The DMP server will not obtain any information from this service, thus preserving users' privacy.

5.2 Authorisation Mechanisms

Authorisation is the process for determining whether the authenticated user has access to the particular resources on the DMP. During this process, the authenticated user's rights and permissions are checked and verified by the DMP. In DMP v0.6.0, Role-based Access Control (RBAC) has been implemented to secure access to project datasets:

- **RBAC:** registered users are granted access to resources on the DMP based on their roles. RBAC allows access control to be managed at a very granular level. Each role is associated with a set of permissions and privileges defined by the DMP. In DMP v0.6.0, roles are implemented both at a system level (e.g. 'system admin' and 'system user' of the DMP) and a resource level (e.g. 'data manager', 'data uploader' and 'data viewer' of a study). The roles and their associated rights are centrally managed by the ICL team. The difference between roles are summarised in Section 5.3.
- **Attribute-based Access Control (ABAC):** ABAC allows access rights to be assigned to a registered user based on a set of policies. A policy is expressed by attributes, which can be characteristics of users, resources, object or environment. In future versions of the DMP, ABAC will be designed and developed according to the project needs. For example, a user who belongs to an organisation may have the permission to delete data files which are uploaded by a member of the same organisation.
- **API Authorisation:** token-based authorisation mechanism such as JSON Web Token (JWT) may be designed and developed in future versions of the DMP according to the project needs. Once a registered user or application has successfully logged in to the DMP via public-key authentication, an access token which is associated with the requester's permissions with an expiration time can be generated by the DMP and issued to the requester. The requester can then use this token to interact with the DMP's APIs until the token expires.

5.3 User Management

User management functions provided by the DMP include: 1) registering new users; 2) modifying existing users; 3) assigning/removing permissions of users; and 4) logging user activities:

- **Registering new users:** a new user needs to create an account on the DMP by completing the user registration form via the DMP's web interface. A confirmation email with a Base32 secret key will sent to the new user's email address after the form is submitted (see Section 5.1 and Appendix A). The new user's role is set to 'system user' by default, and the user has no access to any resource on the DMP at the time of registration. The new user needs to contact a system administrator or a data manager to get access to a dataset hosted on the DMP.

- **Modifying exist users:** a user's account information (e.g. organisation, account expiry date and user type) can be updated by a system administrator of the DMP. Note that some user account information (e.g. username and email address) cannot be changed after registration.

  An account expiration control mechanism is in place to enforce user accounts to be reviewed regularly. All user accounts on the DMP have an expiration date no more than 3 months. The ICL team in WP5 will contact representatives of each IDEA-FAST partners on a quarterly basis to ensure all users on the DMP are active participants in the IDEA-FAST project.

- **Assigning/removing permissions of users:** granular permission control based on RBAC is implemented in the DMP. A system admin can assign system-level roles to users, and a system admin/data manager can assign resource-level roles to users to manage access of a specific dataset. Table 3 summarises different user roles and their rights in the DMP.

- **Logging user activities:** all user activities on the DMP are tracked and recorded in an activity log. Malicious activities (e.g. suspicious logon and logoff attempts) can be identified which can reduce security risks and prevent data breaches. An activity log record contains information including the username, operation type, time and the request status. Note that the user activity log is only visible to the system admins. Distributed Ledger and Smart Contract technologies may be used in future versions of the DMP to provide transparent and tamper-resistance audit logs according to the project needs.

Table 3: Different user roles in the DMP.

Role	**Rights**
System-level User Roles	
System Admin	Log in to the DMP, view the list of users, view and modify account information, deactivate/reactivate a user, delete a user account, create and delete a dataset, access datasets on the DMP, access user activity log
System User	Log in to the DMP, access datasets based on permission
Resource-level User Roles	
Data Manager	View the dataset, assign resource-level roles to users, manage access control for the dataset, upload data to the dataset, download data from the dataset, delete data in the dataset, run analysis on the dataset in the DMP analytical environment
Data Uploader	View the dataset, upload data to the dataset, run analysis on the dataset in the DMP analytical environment
Data Downloader	View the dataset, download data from the dataset, run analysis on the dataset in the DMP analytical environment
Data Viewer	View the dataset, run analysis on the dataset in the DMP analytical environment

6 API Specifications

APIs of the DMP are a set of definitions and protocols which define how other systems and applications (e.g. clinical eCRF and patient-facing application) can interact with and transfer data to/from the DMP. APIs of the DMP allow users to access resources hosted on the platform while maintaining security and control. They also provide flexibility for development and integration of different systems and applications that are being developed in the project.

We build the APIs of the DMP based on GraphQL, an open-source domain-specific language for data query and manipulation. GraphQL provides flexible access to the underlying data through composition, selection and mutation. Rather than using various path based URIs and HTTP methods, it uses a single endpoint with a predefined schema that specifies how to fetch and change data. Compared to other API design architectures such as REST, GraphQL allows clients to make more precise APIs calls to fetch exactly the data they required from the server, therefore preventing over- and under-fetching.

In DMP v0.6.0, the APIs for authentication, uploading and downloading data files have been implemented and released. The API requests can be made and tested using Postman, a widely used tool for API development.

6.1 Authentication via APIs

A client needs to be authenticated by the DMP before it can upload or download any data via APIs. For a registered user, the DMP uses the password-based authentication with the 2FA mechanism to verify the identity of the user, as mentioned in Section 5.1. The registered user needs to send the login data (i.e. username, password and the TOTP) in a GraphQL query (see Appendix C, Code C.1) to the DMP server.

HTTP request:

- Endpoint: https://data.ideafast.eu/graphql
- Method: POST

An example of a GraphQL query for user authentication can be found in Appendix C. Note that the returned data structure in a GraphQL query is flexible and users can precisely specify the required data. Table 4 lists the query parameters:

Table 4: List of parameters in a login request.

Parameters	Value Type	Description
username	string	Username as registered on the DMP.
password	string	User's password.
totp	string	The TOTP generated from the QR code included in the registration confirmation email, see Section 4.1 for details.

After the query has been processed and the client's identity is verified, the server will send the client a response of which the header includes cookies in the following format:

Code 1: An example of the cookie in the server's response header.

```
connect.sid=s%3AL6IUbbRzU456N5MUuIepcrWBHt6UnKdD.BILO2momQ6KFGreU5Gdv3wfwRayDfSFvOV8Km9yOPmI; Path=/; HttpOnly
```

The response cookies need to be attached to any further requests made by the client as a proof of the client's identify. In DMP v0.6.0, the response cookie is set to be valid for a fixed period of 2 hours. A new login request needs to be made for accessing the DMP after 2 hours.

An example of the structure of a response body is shown in Code 2, with properties listed in Table 5.

Code 2: An example of the structure of a response body of a login request.

```
{
    "id": ****,
    "username": ****,
    "type": ****,
    "firstname": ****,
    "lastname": ****,
    "email": ****,
    "organisation": ****,
    "description": "",
    "access": {
        "id": ****,
        "projects": [
            {
                "id": ****,
                "name": ****,
                "studyId": ****
            }
        ],
        "studies": [
            {
                "id": ****,
                "name": ****
            },
        ]
    },
    "createdAt": ****,
    "expiredAt": ****
}
```

Table 5: List of properties that can be returned by a login request.

Property	Value Type	Description
id	string	The requester's user ID.
username	string	The requester's username as registered on the DMP.
type	string	The requester's user type, e.g., standard user or system admin.
firstname	string	The requester's first name as registered on the DMP.
lastname	string	The requester's last name as registered on the DMP.
email	string	The requester's email address as registered on the DMP.
organization	string	The requester's organisation as registered on the DMP. The organisation should be one of the 47 IDEA-FAST project participant organisations.
description	string	The requester's user description as registered on the DMP.
access	object	Information of the requester's accessibility to the studies/projected on the DMP. The granted studies/projects are shown in a list with their IDs and names.
createdAt	number	The time at which the requester created the account.
expiredAt	number	The time at which the requester's account expires.

6.2 Obtaining File List via APIs

If a client/registered user has access to a study hosted on the DMP, the client/registered user can get the list of all data files of that study along with the file properties.

HTTP request:

- Endpoint: https://data.ideafast.eu/graphql
- Method: POST

An example of a GraphQL query for obtaining files associated with a study can be found in Appendix C. In DMP v0.6.0, the parameter required in this query is 'studyID', which is a string representing the ID of the study. Advanced filter functions will be designed and implemented in future API versions to allow users to retrieve a list of files matching some given conditions such all files associated with a specific participant cohort or a specific type of device.

If the query is successful, a response with the information of the files will be returned to the requester. An example of the structure of a response body is shown in Code 3, with properties listed in Table 6.

Code 3: An example of the structure of a response body of a getStudy request.

```
{
    .......
    "files": [
        {
            "id": ****,
            "fileName": ****,
            "studyId": ****,
            "projectId": ****,
            "fileSize": ****,
            "description": ****,
            "uploadTime": ****,
            "uploadedBy": ****
        }
    ],
    ......
}
```

Table 6: List of properties that can be returned by a getStudy request.

Property	Value Type	Description
files	list	The list of files associated with a given study.
file.id	string	The unique ID of the file.
file.fileName	string	The name of the file.
file.studyId	string	The ID of the study that the file belongs to.
file.projectId	string	The ID of the project that the file belongs to.
file.fileSize	number	The size of the file in bytes.
file.description	string	The description of the file. In DMP v0.6.0, the description of a device data file includes the ID of the participant from whom the data was collected, the ID of the device generating the data, the start date of data collection and the end date of data collection.
file.uploadTime	string	The time at which the file was uploaded.
file.uploadedBy	string	The ID of the user who uploaded the file.

6.3 Uploading Files via APIs

A client/registered user can upload data files to a study hosted on the DMP via APIs.

HTTP request:

- Endpoint: https://data.ideafast.eu/graphql
- Method: POST

An example of a GraphQL query for uploading files to a study can be found in Appendix C. The query parameters are listed in Table 7.

Table 7: The list of parameters in an uploadFile request.

Parameters	Value Type	Description
file	null	The file to be uploaded to the DMP.
studyId	string	The ID of the study that the file belongs to.
description	string	The description of the file. In DMP v0.6.0, the description of a device data file includes the ID of the participant from whom the data was collected, the ID of the device generating the data, the start date of data collection and the end date of data collection.

If the query is successful, a response with the uploaded file information will be returned to the requester. An example of the structure of a response body is shown in Code 4, with properties listed in Table 8.

Code 4: An example of the structure of a response body of an uploadFile request.

```
{
    "id": ****,
    "fileName": ****,
    "studyId": ****,
    "projectId": ****,
    "fileSize": ****,
    "description": ****,
    "uploadedBy": ****
}
```

Table 8: List of properties can be returned by an uploadFile request.

Property	Value Type	Description
id	string	The ID assigned to the uploaded file.
fileName	string	The name of the uploaded file.
studyId	string	The ID of the study that the file belongs to.
projectId	string	The ID of the project that the file belongs to.
fileSize	number	The size of the uploaded file in bytes.
description	string	The description of the file. In DMP v0.6.0, the description of a device data file includes the ID of the participant from whom the data was collected, the ID of the device generating the data, the start date of data collection and the end date of data collection.
uploadedBy	string	The ID of the user who uploaded the file.

6.4 Downloading Files via APIs

If a client/registered user has access to a study hosted on the DMP, the client/registered user can download the files under the study via APIs. The ID of the file (which can be obtained by a getStudy request as described in Section 6.2) needs to be specified in the request.

HTTP request:

- Endpoint: https://data.ideafast.eu/file/:fileId
- Method: GET

Example:

- https://data.ideafast.eu/file/108be45e-440b-48d9-9730-4319db4e335b

Note that the cookies obtained during authentication (see Section 6.1) need to be attached as the header of the request as a proof of the requester's identify.

If the query is successful, a response with the requested file and its metadata will be sent back to the requester. Two properties returned by a successful download request are listed in Table 9.

Table 9: List of properties can be returned by a downloadFile request.

Property	Value Type	Description
response.header.Content-Disposition	String	The file name of the downloaded file.
response.content	Byte (In ASCII coding)	The original data of the downloaded file.

Note that the downloaded file will be encoded in ASCII, and the user may need to convert it to a human-readable format. There are multiple ways to do this. For example, the user can use a python script to read the data in the response.content and then write it to a plain file in byte mode (e.g. using the 'wb' writing mode).

6.5 API Versioning

Every time the DMP evolves to a new version, the existing APIs may need to be updated, and new APIs may become available to bring new features. API versioning is an effective practice to ensure such changes do not affect the API users/clients. API versioning permits users/clients to keep on using an existing API and only migrate or update their applications to the latest API version when they are ready.

In DMP v0.6.0, the APIs are maintained in a single version. In future versions of the DMP, API versioning feature will be designed and developed to keep the compatibility of multiple API versions. The following strategies may be used:

- Maintain multiple versions of APIs at the same time in order to provide support to users or systems that do not upgrade old APIs to the latest version in time.
- Keep the consistency of versions or APIs that are in use, i.e. to ensure that each version of APIs in use has both backward and forward compatibility.

7 Analytical Environment

The DMP will provide a secure and user-friendly analytical environment where users can explore and analyse the datasets hosted on the DMP through a web interface. The analytical environment will enable users to develop and validate research hypotheses by investigating clinical and device data. Users of the DMP will be able to use pre-defined data analysis pipelines to run experiments on a selected cohort. The analytical environment will allow users to write and run their own custom scripts to perform analysis based on their needs.

For a data privacy and security perspective, the analyses and computation capacities provided by the analytical environment will allow users to analyse the project datasets without downloading and keeping a copy of the dataset on their local computers, which further prevents data breaches.

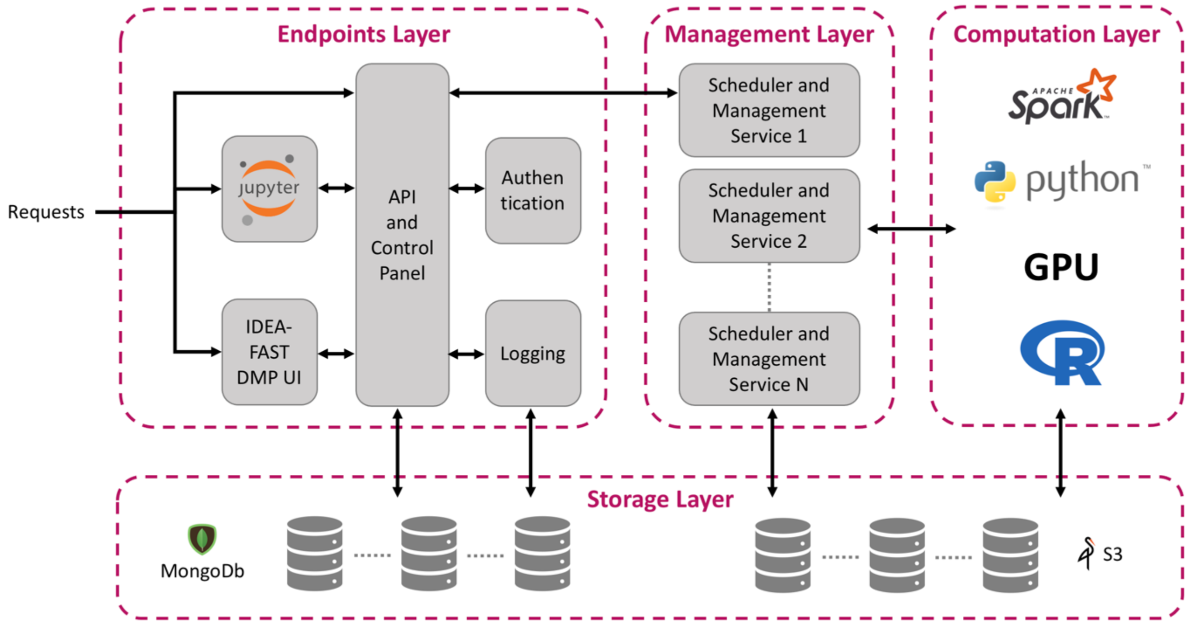


Figure 2: Architecture of the DMP analytical environment. The architecture adapts from the eAE[2].

[2] Oehmichen, A. (2018). *eTRIKS analytical environment: a practical platform for medical big data analysis*.

The design and implementation of the DMP analytical environment will based on the eTRIKS Analytical Environment (eAE)[3], an open source modular high-performance framework for efficient analysis of large-scale biomedical data. The eAE was developed by the IMI eTRIKS project[4], and has been used to support several translational research projects. The IDEA-FAST DMP will reuse the eAE components that were developed by the eTRIKS project, but also develop new features to enhance the functionality of the DMP analytical environment.

The architecture of the DMP analytical environment is shown in Figure 2. It consists of four layers, including: 1) the endpoints layer, which interacts with the users via UIs; 2) the storage layers, where the analytical results are stored; 3) the management layer, which schedules and manages the computational jobs submitted by the users; and 4) the computational layer, where the computations are executed.

- **Endpoints layer:** this layer provides user entry points to the analytical environment. It is composed of interface services (IDEA-FAST DMP UI and Jupyter UI), the API and control panel, authentication and logging services. A user's request will first be verified by the authentication services, and a verified request will be logged by the logging service for audit purposes and traceability. A request also needs to be validated by the interface services. The interface services include: 1) an IDEA-FAST DMP interface that allows users can perform analysis based on predefined analytical pipelines; and 2) a Jupyter Notebook interface that allows user to write and run their own custom scripts. A verified and valid request will be saved in the Storage Layer for the Management Layer to schedule.
- **Storage layer:** this layer manages the project datasets stored in the DMP as well as the DMP-specific data (e.g. user credentials and query parameters). As mentioned in Section 4, the DMP uses the NoSQL database MongoDB to store metadata and the object based S3 storage (MinIO in DMP v0.6.0) to store the project datasets. The storage layer supports all the other layers by providing replicated, distributed, and scalable storage resources.
- **Management layer:** this layer schedules the computation of the analyses onto compute nodes based on their availability and the type of analyses requested. It is composed of a set of scheduling and management services which run independently in parallel. Each service periodically fetches job requests from the storage layer, and for each fetched request, it checks if a compute note is available to execute it. If the check is successfully, the service will then send the job request to the Computation Layer for execution.
- **Computational layer:** this layer provides execution capabilities to the DMP's analytical environment. When receiving a job request, the compute service within the layer will fetch the required data and the analysis codes from the Storage Layer, run the job, generate results, and save the results and the execution logs in the Storage Layer. Different types of computation jobs are supported by the DMP's analytical environment, such as Python 2, Python 3, R, and Spark.

Pre-defined data analysis pipelines will be implemented in the DMP with inputs from IDEA-FAST colleagues involved in data modelling and analysis (e.g., WP4 and WP7). Developed models and algorithms will be transformed and embedded in the DMP's analytical environment, so the models and algorithms can be easily reused by the users of the DMP to investigate and validate results. In addition, meta information of how a machine learning model is trained and tuned, will be properly captured and managed in the DMP's analytical environment, which can facilitate the future model validation and

[3] Oehmichen, A., Guitton, F., Sun, K., Grizet, J., Heinis, T., & Guo, Y. (2017). eTRIKS analytical environment: a modular high-performance framework for medical data analysis. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 353-360).
[4] https://www.imi.europa.eu/projects-results/project-factsheets/etriks

compliance check. Together with the DMP, all analytical pipelines implemented in the analytical environment will be made available to the research community and maintained on GitHub.

8 Security and Regulatory Compliance

8.1 Introduction

8.1.1 Scope and Goals

The purpose of this Section is to provide:

- The general approach to security design of the DMP within the project.
- A limited set of non-functional security related requirements. These requirements are organized into categories according to their specific goal (e.g., access control, cryptography, storage, etc.) and are, as far as possible, expressed in a general form, so as not to impose constraints on the implementation.
- A clear definition of assumption on the system to be "secured".

The role of Pluribus One within the project is to contribute to the Data Management Platform's security by defining and implementing an approach (i.e., both at a high and low level) to manage cyber security and solve concerns. Pluribus One's contribution to this deliverable consists of:

- A definition of all the stakeholders and "actors" that are, albeit at different levels, interested in security-related aspects of the project, and how they are supposed to use this document.
- Providing a list of assumptions and dependencies associated to internal and external elements of the project. This is important in order to evaluate relevant sources of security issues and to reduce unexpected risks.
- Providing a definition of features and non-functional requirements for the data management platform (i.e. and related to its interaction with other systems and "actors" within the IDEA-FAST whole platform).

8.1.2 Target Audience and How-to-Use

The expected readers of this Section are project managers, developers, testers and, in broader terms, all the project stakeholders. We assume all these people to be, albeit at a different level, affected by security related decisions and activities.

This is a list of expected readers:

- Software developers (i.e., developers involved in the implementation of the Data Management Platform and other components that communicate with it).
- Software Engineers that are involved in the design of the Data Management Platform.
- System Administrators and QA operators that take care of the deployment configuration, administration of the environment that runs the Data Management Platform.
- Software developers that will be involved in future activities associated with the maintenance and upgrade of the software.
- Security operators that are involved in the cyber security activities that target the Data Management Platform.
- Project Managers and other consortium partners.

Software developers should read this Section focusing on the list of non-functional requirements. The requirement's list is obviously extremely important for developers of the Data Management Platform but some parts of it are of interest also for developers of other software components that are expected to communicate with the DMP.

System Administrators of the Data Management Platform underlying system should read this Section focusing on the requirements associated with their activity. Several security related requirements focus on the management of the application server, on security policies (e.g. access control, root access, internal server communication, secrets management, log policies, etc.) and not only on the code itself.

Security operators should read this Section looking for guidelines on what are the important requirements and consider it as a base to define a detailed list of security checks.

Project managers should read this Section looking for insight in security-related issues, to have a guideline in the creation of a list of measurable activities that needs to be performed.

8.1.3 General Scope

The purpose of the security-related activities is to help the developers in the implementation of an information system that is aligned to the standards in terms of security.

This goal will be reached by simultaneously working on two levels:

- Higher level – by providing general guidelines that can be applied on the development process (e.g. by raising developers' awareness of security related issues, by integrating security practises within the software development life cycle, etc.).
- Lower level – by performing traditional security level activities on specific software components (e.g. penetration testing against a software component, static code analysis, etc.).

8.1.4 Definitions and Acronyms

ASVS - OWASP Application Security Verification Standard, a framework for web-application security. For more information about OWASP ASVS please refer to https://owasp.org/www-project-application-security-verification-standard/.

Authenticator - Code that authenticates a password, token, MFA, federated assertion, and so on.

DMP - Data Management Platform.

CDN - Content delivery networks, a provider of externally hosted resources used in a web application.

CIA - Confidentiality, Integrity and Availability.

CSP - Credential Service Provider also called an Identity Provider

CSRF - Cross-Site Request Forgery

DoS - Denial of Service, a type of attack.

LFI - Local File Inclusion, a type of File Inclusion attack against web applications.

MFA - Multi factor authenticator, which includes two or more single factors

OAS - OpenAPI Specification (OAS), a standard, language-agnostic interface to RESTful API. https://swagger.io/specification/

OTP - One-time password

PII - personally identifiable information (PII)

RFD - reflective file download, a type of attack against web applications.

RFI - Remote File Inclusion, a type of File Inclusion attack against web applications.

SFA - Single factor authenticators, such as something you know (memorized secrets, passwords, passphrases, PINs), something you are (biometrics, fingerprint, face scans), or something you have (OTP tokens, a cryptographic device such as a smart card).

SRI - Subresource Integrity, an integrity check used in the validation of the integrity of externally hosted resources of a web application.

XSS - Cross-Site Scripting, one of the most common vulnerabilities in web-applications.

Verifier – "An entity that verifies the claimant's identity by verifying the claimant's possession and control of one or two authenticators using an authentication protocol. To do this, the verifier may also need to validate credentials that link the authenticator(s) to the subscriber's identifier and check their status"

8.2 Proposed Approach to Security for the DMP

As previously mentioned, the goal of the security-related activities within WP5 is to contribute to the development of the data management platform in order to improve its "security". In order to achieve this goal, several activities will be performed but one key, underlying principle will always lead our activities: to strive to be become part of the software development life cycle in order to make security part of the system with the aim of creating a secure-by-design-software.

Cybersecurity activities can be many and different but among the most important ones there is the definition of boundaries of what should be done and under what laws, guidelines and regulations the system is supposed to "live".

GDPR provides us with a strong set of starting rules and limitations but this is not enough. There are many advantages and sound reasons for the adoption of a security framework as a set of guidelines under which to work.

A security framework can be useful especially on these aspects.

- **Security improvements.** Security frameworks are designed to help organizations to define, improve and implement their security setting. Security frameworks will help an organization to better understand, manage, and reduce its cybersecurity risks.
- **Prioritize actions**. Security frameworks will help the management to determine which activities are most important, so they help to prioritize investments and maximize the investments' impact.
- **Measure security effectiveness**. Security frameworks provide a common set of practices to follow, and solid baselines and benchmarks for measuring security effectiveness. This allows the organization to simplify the creation of reports on improvement. Moreover, baselines and benchmarks can be used to establish a level of confidence in the applications' security.
- **Communication**. Security frameworks help communicate inside and outside the organization, because they provide a common language to address cybersecurity risk management.

The final decision about which security framework to follow, and a definitive list of security guidelines, has not been reached yet. There has been an agreement on the importance of the adoption of a security framework and about the definition of a set of guidelines.

The following lists presents some key activities that are parts of the proposed approach:

1. Identification of key data that should be protected, secured and managed with extreme care.
2. Identification of possible threats for the data of interest (e.g. corruption, accidental and deliberate data loss, intelligence gathering, etc.).
3. Identification and definition of security related non-functional requirements (e.g. system access management requirements, data management requirements, etc.).
4. Continuous identification of potential issue regarding the system's compliance with standards and regulations for privacy (i.e. GDPR).

The following list includes some activities that will be performed during the development phase in terms of security:

1. Definition of the boundaries of the system and creation of a list of possible targets that require protection.
2. Threat modelling with the purpose of creating a ranked list of possible threats.
3. Definition of a series of guidelines and requirements for the system.
4. Risk assessment and insight in terms of Risk Management Strategy.

8.2.1 Assumptions and Dependencies

Section 8 of the deliverable has been written under some assumptions. It is important to take them into account:

- The entire IDEA-FAST system will be complex and composed of several "parts" (e.g. custom software, medical devices, etc.).
- Some of the IDEA-FAST system parts will be developed within the project while others will be integrated.
- The WP5 activities focuses on a single part of the entire system (i.e. data management platform) but its security-related requirements are associated to "external" elements such as the sources of inbound data (i.e. medical devices).
- The Data Management Platform is a web-application that communicates with external systems via API and also provides a web-based GUI to human users.
- The Data Management Platform has a stand-alone independent data layer with a data storage used to store all data that is important for business and security purposes.
- The Data Management Platform will store data of several kinds, some of which should be considered sensible (i.e. medical data, personal data).
- The security analysts will be allowed to access the software platform to execute an agreed set of tests regarding the security.
- The security analysts will be allowed to test a software platform in a secure way without any risk of causing loss or damage (e.g. by testing a newly deployed software platform with mock data) but at the same time by providing sound and valid feedback for the developers.

8.3 System Security Features and Requirements

According to the IEEE Standard Glossary of Software Engineering Terminology 5, a "requirement" can be defined as:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

This Section aims at providing a specification of all requirements associated to security and establish the basis for an agreement on how the software should function in terms of security.

A *software requirements specification* aims to be a rigorous assessment of requirements before a more specific design phase actually happens and its goal is to reduce, as far as possible, later redesign.

A *software requirement specification* is considered extremely useful because it provides an estimate of production costs and risks [6] and, if appropriately used, it can help to prevent the failure of a software project [7].

In software engineering requirements are usually divided in two main groups: *functional* (i.e. what the system is supposed to do, the "what") and *non-functional* (i.e. criteria usable to judge the performance of the system, the "how").

In simpler words, we can state that non-functional requirements are other requirements that are not associated with features of the software but with important characteristics of it. *Non-functional* requirements focus on the answers to questions such as: how secure is the system? How usable is it? How good are the system's performances?

Non-functional requirements usually are not associated to a specific line of code but are more "emergent" properties that arise from the entire solution; it is nonetheless possible to pick a certain design, to follow some guidelines and develop solutions that can implement the "guidelines" provided by these requirements.

8.3.1 Remarks on Requirements

The most discussed *non-functional* requirement in software development are:

- Security.
- Performance.
- Availability.
- Extensibility (i.e. how easy it is to upgrade and improve the system).

There are some problems associated with non-functional requirements, starting from how difficult it is to strictly define them (e.g. what makes a system "secure"?), to measure them (e.g. how fast is enough to be happy with the performances?) and to compare them.

Taking into account security related requirements we can be sure that security comes with a "price" (e.g. time, human and financial resources, expertise, hardware, etc.), especially when it comes to write secure software and we can also be sure that nobody would like to use "non-secure" software but, at the same time, measuring how secure software is a non-trivial task.

[5] IEEE Computer Society (1990). "IEEE Standard Glossary of Software Engineering Terminology". IEEE Standard.
[6] Bourque, P.; Fairley, R.E. (2014). "Guide to the Software Engineering Body of Knowledge (SWEBOK)". IEEE Computer Society. Retrieved 17 July 2014.
[7] "Software requirements specification helps to protect IT projects from failure". Retrieved 19 December 2016.

8.3.2 Security Requirements Overview

This paragraph focuses on a single non-functional requirement: Security. We define Security as a quality attribute that heavily interacts with other attributes such as Availability and Robustness. Security is the sum of all the attributes of an information system which contributes toward ensuring that all information is processed, stored and transferred while performing three actions:

- Protecting Confidentiality.
- Enforcing Integrity.
- Checking Authenticity.

We can divide the requirements into some sub-categories according to their main "goal" or area of interest. Each sub-category starts with the code "SR" that stands for Security Requirements.

The following list reports all sub-categories with a brief description:

- **SR1 Authentication Verification Requirements** – requirements that focus on the authentication management within the application.
- **SR2 Session Management Verification Requirements** – requirements that focus on the session management.
- **SR3 Access Control Verification Requirements** – requirements that focus on the access control management within the application and its environment. These requirements are relevant for system administrators.
- **SR4 Validation, Sanitization, Encoding Verification Requirements** – requirements that focus on input validation, sanitization and encoding.
- **SR5 Stored Cryptography Verification Requirements** – requirements that focus on cryptography management. These requirements are relevant for system administrators.
- **SR6 Error Handling and Logging Verification Requirements** – requirements that focus on error management and logging. These requirements are relevant for system administrators due to log management.
- **SR7 Data Protection Verification Requirements** – requirements that focus on data protection related issues including data storage, data backup, etc. These requirements are relevant for system administrators due to backup management, cache handling and hardware configuration (e.g. load balancers).
- **SR8 Communications Verification Requirements** – these requirements focus on encryption of the communication. These requirements are relevant for system administrators.
- **SR9 Malicious Code Verification Requirements** – these requirements focus on security associated with third party libraries, middleware and the possibility of malicious code being inserted in the code base.
- **SR10 File and Resource Verification Requirements** – these requirements focus on security issues associated with file management. These requirements are relevant for system administrators.
- **SR11 API and Web Service Verification Requirements** – These requirements focus on security issues associated with the communication with external components.
- **SR12 Configuration Verification Requirements** – These requirements focus on security issues associated with the configuration of the application and of the application environment (i.e., the application server, runtime environments). These requirements are relevant for system administrators.

For a complete list of all non-functional security related requirements, please refer to Appendix D.

8.3.3 Security Requirements Complete List

This Section presents a complete list of security non-functional requirements divided into categories according to their purpose. The general structure of these requirements is strongly based on OWASP Application Security Verification Standard (ASVS), a framework for web-application security (for more information about OWASP ASVS please refer to https://owasp.org/www-project-application-security-verification-standard/).

As previously stated, all requirements are divided into macro categories and, in certain cases, a second level of sub-categories is used too. For each category of requirements, there is a table with the following columns:

- **#RID** – a unique identifier for the requirements for traceability purpose.
- **Description** – a brief description of the requirement. A more complete and precise description will be provided, where required, during the following phases of the development.
- **OWASP** – a series of identifiers of OWASP ASVS "controls". The numbering is based on ASVS ver. 4.0.
- **NIST** – a series of identifiers of NIST 800-63 standard that are associated to the requirement.

9 Conclusions

This document describes the specifications of the IDEA-FAST DMP. It is a working document and will be updated whenever any changes arise.

Appendix A – User Guide to DMP V0.6.0

This section provides guidelines for a user to: 1) register an account to access the IDEA-FAST DMP v0.6.0; 2) upload/download data to/from the IDEA-FAST feasibility study dataset.

Register a new account

Step 1: Go to https://data.ideafast.eu/. Click "Please register" on the Welcome page:

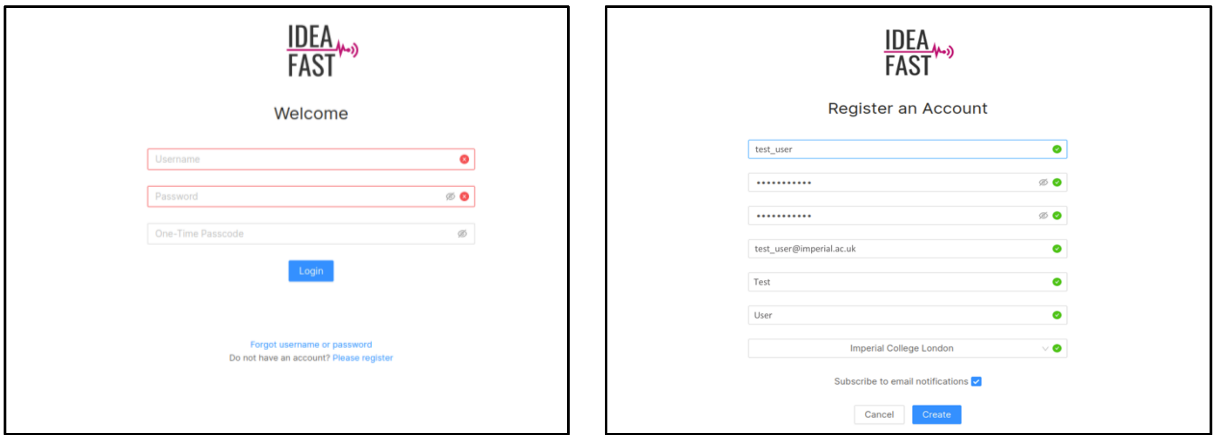
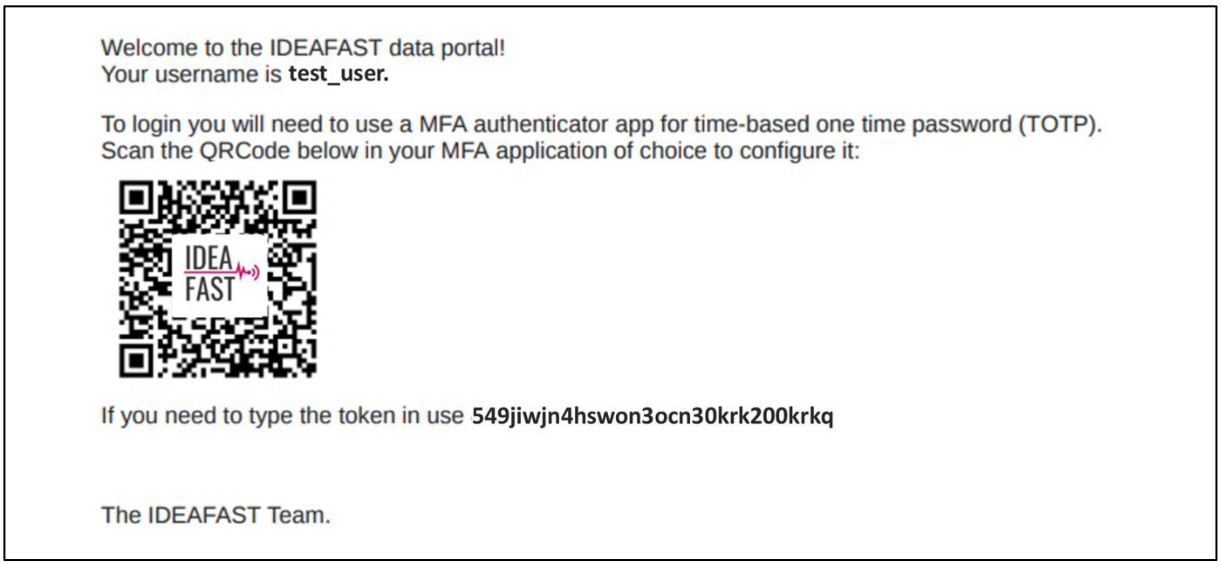


Figure A.1: DMP User Registration.

Step 2: Fill in your registration information: click "Create" and you will see the following page if the registration is successful.

Step 3: Check your email and get your time-based one-time password (TOTP). You will receive an email from <no-reply@ideafast.eu> with a QR code for obtaining the TOTPs.



Welcome to the IDEAFAST data portal!
Your username is **test_user.**

To login you will need to use a MFA authenticator app for time-based one time password (TOTP).
Scan the QRCode below in your MFA application of choice to configure it:

If you need to type the token in use **549jiwjn4hswon3ocn30krk200krkq**

The IDEAFAST Team.

Figure A.2: Example of the account verification email.

Access the platform

Step 1: Go to https://data.ideafast.eu/. Enter your username and password.

Step 2: You will need a time-based one-time password (TOTP) to access the DMP. To get your TOTP, please scan the QR code (or enter the token) provided in the registration email using an Authentication Application (e.g., Google Authentication App or Microsoft Authenticator App) via your smartphone.

Instructions of how to get your TOTP via Google Authentication App:

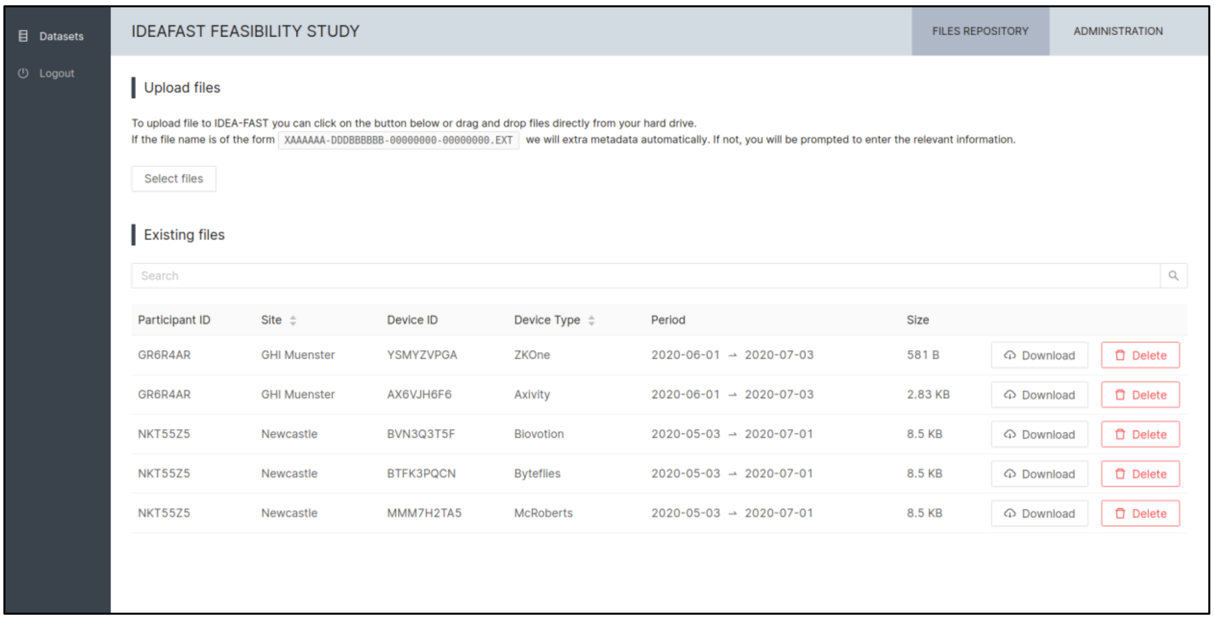
- Step 2.1: Install the App on your smartphone and open the App.
- Step 2.2: Choose **"Scan barcode"** and scan the QR code provided in the email. Or you can choose **"Manual entry"** and enter the token.
- Step 2.3: A 6-digit TOTP will be generated by the App.

Step 3: Enter the 6-digit TOTP in the "One-Time Passcode" field on the Welcome Page and then click "Login".

Access the IDEA-FAST Feasibility Study Dataset

User accounts will be reviewed by the Admin team of the IDEA-FAST DMP. Your permission to access the Feasibility Study Dataset will be assigned by the Admin team.

Step 1: After login you will see the Datasets page. Click the "IDEA-FAST FEASIBILITY STUDY" dataset. You will then see a list of existing files under this dataset. Please contact the WP5 ICL team you have any problems on accessing the dataset.



Upload files

To upload file to IDEA-FAST you can click on the button below or drag and drop files directly from your hard drive.
If the file name is of the form `XAAAAAA-DDDBBBBBB-00000000-00000000.EXT` we will extra metadata automatically. If not, you will be prompted to enter the relevant information.

Existing files

Participant ID	Site	Device ID	Device Type	Period	Size		
GR6R4AR	GHI Muenster	YSMYZVPGA	ZKOne	2020-06-01 → 2020-07-03	581 B	Download	Delete
GR6R4AR	GHI Muenster	AX6VJH6F6	Axivity	2020-06-01 → 2020-07-03	2.83 KB	Download	Delete
NKT55Z5	Newcastle	BVN3Q3T5F	Biovotion	2020-05-03 → 2020-07-01	8.5 KB	Download	Delete
NKT55Z5	Newcastle	BTFK3PQCN	Byteflies	2020-05-03 → 2020-07-01	8.5 KB	Download	Delete
NKT55Z5	Newcastle	MMM7H2TA5	McRoberts	2020-05-03 → 2020-07-01	8.5 KB	Download	Delete

Figure A.3: DMP data file exploration.

Upload Data to the IDEA-FAST Feasibility Study Dataset

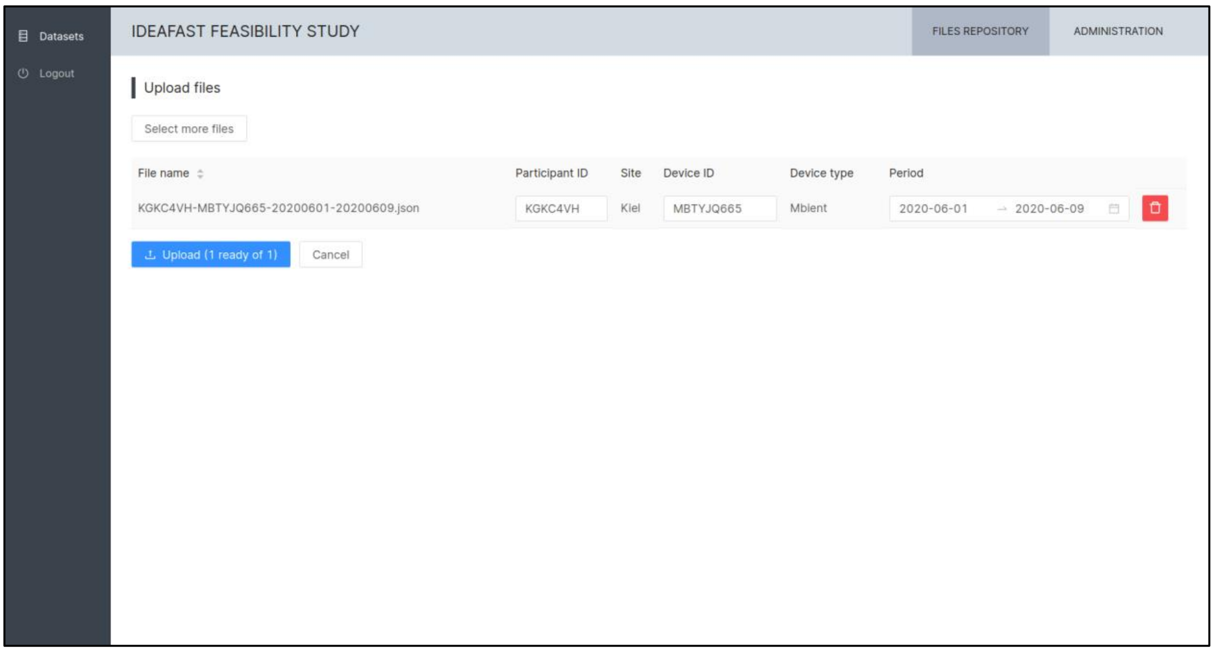
Step 1: On the IDEA-FAST FEASIBILITY STUDY dataset page, click "Select files" under "Upload files" section.

Step 2: Choose the data files from your local machine and click **"OK"**. If the name of a file is in the following form, the platform will extract metadata automatically:

XAAAAAA-DDDBBBBBB-SSSSSSSS-EEEEEEEE.EXT

where "XAAAAAA" is the IDEA-FAST FS participant ID, "DDDBBBBBBBB" is the IDEA-FAST Device ID, "SSSSSSSS" is the record start date, and "EEEEEEEE" is the record end date. For example: IAAAAAA-MMM3PQCN-20200702-20200703.ZIP

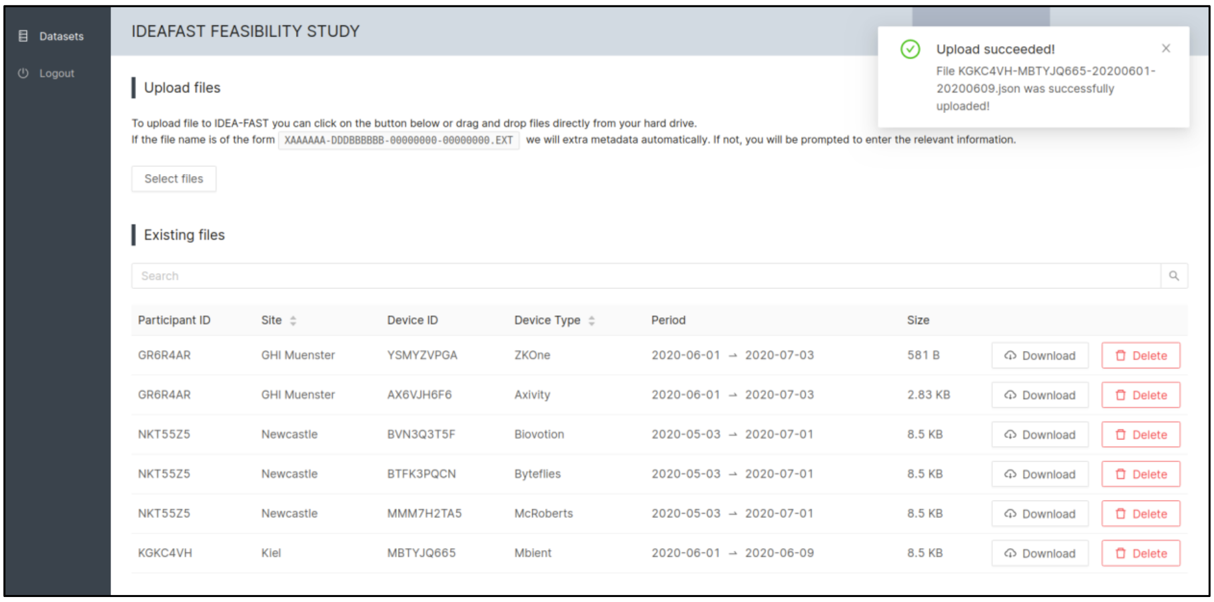
The platform will automatically check the file name and extract the Participant ID, Site, Device ID, Device Type and Recording Period information. If the filename is not in the standard format, you will need to enter the valid Participant ID, Device ID and Recording Period before uploading the data.



IDEAFAST FEASIBILITY STUDY

Upload files

File name	Participant ID	Site	Device ID	Device type	Period
KGKC4VH-MBTYJQ665-20200601-20200609.json	KGKC4VH	Kiel	MBTYJQ665	Mbient	2020-06-01 → 2020-06-09



IDEAFAST FEASIBILITY STUDY

Upload succeeded!
File KGKC4VH-MBTYJQ665-20200601-20200609.json was successfully uploaded!

Upload files

To upload file to IDEA-FAST you can click on the button below or drag and drop files directly from your hard drive.
If the file name is of the form `XAAAAAA-DDDBBBBBB-00000000-00000000.EXT` we will extra metadata automatically. If not, you will be prompted to enter the relevant information.

Existing files

Search

Participant ID	Site	Device ID	Device Type	Period	Size		
GR6R4AR	GHI Muenster	YSMYZVPGA	ZKOne	2020-06-01 → 2020-07-03	581 B	Download	Delete
GR6R4AR	GHI Muenster	AX6VJH6F6	Axivity	2020-06-01 → 2020-07-03	2.83 KB	Download	Delete
NKT55Z5	Newcastle	BVN3Q3T5F	Biovotion	2020-05-03 → 2020-07-01	8.5 KB	Download	Delete
NKT55Z5	Newcastle	BTFK3PQCN	Byteflies	2020-05-03 → 2020-07-01	8.5 KB	Download	Delete
NKT55Z5	Newcastle	MMM7H2TA5	McRoberts	2020-05-03 → 2020-07-01	8.5 KB	Download	Delete
KGKC4VH	Kiel	MBTYJQ665	Mbient	2020-06-01 → 2020-06-09	8.5 KB	Download	Delete

Figure A.4: Upload data file to the DMP.

Step 3: Click the "Upload (N ready of M)" to upload the file to the platform. A pop-up notification of "Upload successful" will appear at the top-right corner if the upload is successful (Figure A.4).

Download Data from the IDEA-FAST Feasibility Study Dataset

Step 1: Go to https://data.ideafast.eu/ and login.

Step 2: Click the "IDEA-FAST FEASIBILITY STUDY" dataset. You will then see a list of existing files under this dataset.

Step 3: Click **"Download"** to start the download process.

Appendix B – DMP Data Schema

This section describes the data schema that is currently supported by the DMP.

Data Value Types

The DMP supports different data types. In the data scheme, the data type needs to be specified for each data field. Currently the following types are supported:

- **'c':** categorical data, which represents characteristics, e.g. gender and ethnicity;
- **'d':** decimal data, which represents fractional numbers, e.g. temperature and weight;
- **'i':** integer data, which represents integer values, e.g. number of siblings;
- **'b':** binary data, which represents data that has only two possible states, normally labelled as True and False;
- **'t':** undefined data, which represents free text in string format.

Data Field/Variable Definition

Data fields and associated attributes need to be defined before any data to be uploaded to the DMP. In DMP v0.6.0, the following attributes are used to describe a data field: 1) fieldID; 2) fieldName; 3) valueType; 4) possibleValues; 5) unit; 6) path; 7) numOfTimePoints; 8) numOfMeasurements; 9) startingTimePoint; 10) startingMeasurement; and 11) notes. Details of these attributes are shown in Table B.1.

Table B.1: DMP data field attribute descriptions.

Attribute	**Description**	**Compulsory/ Optional**	**Example**
fieldID	ID of the variable	Compulsory	42
fieldName	Name of the variable	Compulsory	weight
valueType	Data type of the variable	Compulsory	i
possibleValues	Possible values of the variable	Optional for all value types except categorical	[30, 200]
unit	Unit of the variable	Optional	kg
path	Path of the variable	Compulsory	Physical measures > Anthropometry > Body Measurements
numOfTimePoints	Number of time points	Compulsory	7
numOfMeasurements	Number of measurements	Compulsory	4
startingTimePoint	Starting time point	Compulsory	3
startingMeasurement	Starting index of the measurement	Compulsory	3
notes	Notes or addition comments	Optional	The weight of a subject

Note that the 'possibleValues' attribute is optional for decimal, integer, binary and undefined value types but compulsory for categorical variables. Data field definitions need to be prepared in tab-delimited CSV format, with the list of all data field attributes as the header.

Source Data Files

A source data file contains the data to be uploaded to the DMP. The header of a source data file should contain the list of all data fields/variables, starting from the variable 'Eid', which represents the subject ID. The 'Eid' variable is compulsory for all source data files and cannot be omitted or changed.

Apart from the 'Eid' variable, all other data fields in the source data file should use the following format:

{fieldId}@{timepoint}.{measurement}:{datatype}

The {datatype} part is optional, and when unspecified the DMP will set the variable to 'c' categorical data type by default. For example, "1@0.0" and "2@0.1:c" are both valid data fields in a source data file.

A source data file can either be in CSV format or JSON format. When using CSV format, the file starts with header listing all data fields, and each following line of the CSV file corresponds to the data of one subject. Similarly, when using JSON format, the file starts with an array with all data fields, and each following array in the JSON file corresponds to the data of one subject. The DMP will automatedly check the source data file format and raise any errors or inconsistency in the file during the data upload process.

Tables B.2 – B.4 provide an example of how to convert clinical data records (Table B.2) to a data field definition file (Table B.3) and a source data file (Table B.4) that are in the format supported by the DMP.

Table B.2: A dummy example of clinical data for two subjects. There are two timepoints: baseline visit and longitudinal visit. Data of four variables (age, gender, weight and heart rate) are recorded.

Subject ID	Visit	Variable	Value	Unit
A001	Baseline Visit	Age	40	
A001	Baseline Visit	Gender	Female	
A001	Baseline Visit	Weight	75	kg
A001	Baseline Visit	Heart Rate	93	bpm
A001	Longitudinal Visit	Heart Rate	95	bmp
A002	Baseline Visit	Age	42	
A002	Baseline Visit	Gender	Male	
A002	Baseline Visit	Heart Rate	76	bpm
A002	Longitudinal Visit	Heart Rate	73	bmp

Table B.3: The data field definition for the data shown in Table B.2.

fieldId	fieldName	valueType	Possible Values	unit	path	numOf TimePoints	numOf Measurements	Starting TimePoint	Starting Measurement	notes
1	Age	i			Demographic	1	1	1	1	Age
2	Gender	c	Male, Female		Demographic	1	1	1	1	Gender
3	Weight	d		kg	Body Measurement	1	1	1	1	Weight in Kg
4	HeartRate	i		bpm	Vital Signs	2	1	1	1	Heart Rate, 1:baseline,2:longitudinal

Table B.4: The source data file in CSV (top) and JSON (bottom) formats for the data shown in Table 8.2.

Eid	1@1.1:i	2@1.1:c	3@1.1:d	4@1.1:i	4@2.1:i
A001	40	Female	75	93	95
A002	42	Male		76	73

```
[
        [“Eid”, “1@1.1:i”, “2@1.1:c”, “3@1.1:d”, “4@1.1:i”, “4@2.1:i”],
        [“A001”, “40”, “Female”, “75”, “93”, “95”],
        [“A002”, “42”, “Male”, “”, “76”, “73”]
]
```

Appendix C – GraphQL Requests for User Authentication and Data Transfer

This section provides examples of GraphQL requests for a client/user to:

1. Get authenticated (Code C.1);
2. Obtain the list of files (Code C.2);
3. Upload data to the DMP via the DMP's APIs (Code C.3).

Code C.1: GraphQL query for user authentication.

```
fragment ALL_FOR_USER on User {
    id
    username
    type
    firstname
    lastname
    email
    organisation
    description
    access {
        id
        projects {
            id
            name
            studyId
        }
        studies {
            id
            name
        }
    },
    createdAt,
    expiredAt
}

mutation login($username: String!, $password: String!, $totp: String!) {
  login(username: $username, password: $password, totp: $totp) {
      ...ALL_FOR_USER
  }
}
```

Code C.2: GraphQL query for obtaining a list of files.

```graphql
fragment ALL_FOR_JOB on Job {
        id
        studyId
        projectId
        jobType
        requester
        requestTime
        receivedFiles
        status
        error
        cancelled
        cancelledTime
        data
    }

query getStudy($studyId: String!) {
        getStudy(studyId: $studyId) {
            id
            name
            createdBy
            jobs {
                ...ALL_FOR_JOB
            }
            projects {
                id
                studyId
                name
            }
            roles {
                id
                name
                permissions
                projectId
                studyId
                users {
                    id
                    firstname
                    lastname
                    organisation
                    username
                }
            }
            files {
                id
                fileName
                studyId
                projectId
                fileSize
                description
                uploadTime
                uploadedBy
            }
            numOfSubjects
            currentDataVersion
            dataVersions {
                id
                version
                tag
                uploadDate
                jobId
                extractedFrom
                fileSize
                contentId
                fieldTrees
            }
        }
    }
```

Code C.3: GraphQL query for data upload.

```
mutation uploadFile($studyId: String!, $file: Upload!, $description: String!, $fileLength: Int) {
  uploadFile(studyId: $studyId, description: $description, file: $file, fileLength: $fileLength) {
    id
    fileName
    studyId
    projectId
    fileSize
    description
    uploadedBy
  }
```

Appendix D – Security Requirement Descriptions

SR1 Authentication Verification Requirements

SR1.1 Password Security Requirements

Table D.1: Authentication Requirements – Password Security (SR1.1).

#RID	Description	OWASP	NIST
SR1.1	Verify that user set passwords are compliant to a set of known and agreed regulations (e.g. specific regulations can include a minimum length in characters, a set of allowed characters, etc.).	2.1.1, 2.1.2, 2.1.3, 2.1.4	5.1.1.2
SR1.2	Verify that password change functionality exists.	2.1.5, 2.1.6, 2.1.7,	5.1.1.2
SR1.3	The application password creation and change procedures should include some key features such as the availability of the "paste functionality" and the possibility to use external password managers.	2.1.8, 2.1.9, 2.1.10, 2.1.11, 2.1.12	5.1.1.2

SR1.2 General Authenticator Requirements & SR1.3 Authenticator Lifecycle Requirements

Table D.2: Authentication Requirements – General Authenticator (SR1.2), Authenticator Lifecycle (SR1.3).

#RID	Description	OWASP	NIST
SR1.4	The system must have an anti-automatic control in order to mitigate the risk of breached credential testing, brute force lockout attacks and other automatic threats. There are several possible solutions such as CAPTCHA, rate limiting, etc.	2.2.1	5.2.2, 5.1.1.2, 5.1.4.2, 5.1.5.2
SR1.5	Within the application, the use of weak authenticators (such as SMS and email) should be as limited as possible to operations that are not critical. Verify that as far as technically feasible stronger methods are offered before weak methods, that users are aware of risks and that measures are in place to limit the risks of account compromise.	2.2.2	5.2.10
SR1.6	The application must send notifications to users in a secure form after important updates to authentication details (e.g. credential resets, email or address changes, etc.).	2.2.3	-
SR1.7	Within the application, user received notification should be based on push notification technology or if SMS/email based notifications are used then no sensitive information should be disclosed within the notification.	2.2.3	-
SR1.8	Within the application, the initial system generated passwords and activations codes SHOULD be securely randomly generated and expire after a short period of time.	2.3.1	5.1.1.2 / A3

SR1.4 Credential Storage Requirements & SR1.5 Credential Recovery Requirements

Table D.3: Authentication Requirements – Credential Storage (SR1.4), Credential Recovery (SR1.5).

#RID	Description	OWASP	NIST
SR1.9	The password storage must be resistant to offline attacks. Passwords SHALL be salted and hashed solely by using approved one-way key password hashing functions.	2.4.1	5.1.1.2
SR1.10	Within the application, the salt used as input for the password hashing functions should be at least 32 bits in length and arbitrarily chosen to minimize salt collision.	2.4.2	5.1.1.2
SR1.11	Within the application, system generated initial activation or recovery secret must not be sent in clear text to the user.	2.5.1	5.1.1.2
SR1.12	During the password credential recovery process the current password must not be revealed in any way.	2.5.3	5.1.1.2
SR1.13	Password hints (i.e. "secret questions) should not be present.	2.5.2	5.1.1.2
SR1.14	Shared or default accounts (e.g. "admin", "root", etc..) should not be present.	2.5.4	5.1.1.2 /A.3

SR1.6 Service Authentication Requirements

Table D.4: Authentication Requirements – Service Authentication (SR1.6).

#RID	Description	OWASP	NIST
SR1.15	Integration related secrets must not be based on fixed passwords such as API keys or shared privileged accounts.	2.10.1	5.1.1.1
SR1.16	In case passwords are required then the credential must not be a default account.	2.10.2	5.1.1.1
SR1.17	Passwords should be stored with sufficient protection to prevent offline recovery attacks, including local system access.	2.10.3	5.1.1.1
SR1.18	Passwords, integrations with databases, integration with third-party systems, API keys and internal secrets must not be included in source code and never be stored within the source code repositories. The used storage should resist offline attacks.	2.10.4	-

SR2 Session Management Verification Requirements

Any web-based application or stateful API contains a core component that handles session management in order to maintain the state for a user or device that is interacting with the system itself.

There are two high level requirements that are fundamental for the system:

- Sessions must be unique to each individual user or device and cannot be shared or "guessed".
- Sessions must be invalidated when no longer required and timed out after periods of inactivity.

It should be noted that the most common frameworks for the web-application development already propose several software solutions for a safe session management.

SR2.1 Fundamental Session Management Requirements

Table D.5: Session Management Verification Requirements - Fundamental (SR2.1).

#RID	Description	OWASP	NIST
SR2.1	The application must never reveal session's tokens in URL parameters or error messages.	3.1.1	-
SR2.2	The application must generate a new session token on user/device authentication.	3.1.2	7.1
SR2.3	The application must store session's tokens in the web browser only using secure methods (i.e. using secured cookies or by leveraging HTML 5 session storage).	3.2.3	7.1
SR2.4	The application must leverage approved cryptographic algorithms in order to generate session's tokens.	3.2.4	7.1
SR2.5	The application must invalidate session's tokens at logout and after expiration (i.e. using the back button does not resume an authenticated session).	3.3.1	7.1
SR2.6	If cookie-based session is used then the application should have all the cookie's required associated attributes set (e.g. 'Secure', 'HttpOnly').	3.4.1 / 3.4.5	7.1.1
SR2.7	The application must be protected against Session Management Exploits by ensuring a valid login session or by requiring re-authentication before allowing any sensitive transaction or account modification.	3.7.1	-

SR3 Access Control Verification Requirements

Allowing access to resources only to those permitted to use them is the core tenet of authorization.

There are some high-level requirements that are fundamental for the system:

- All persons accessing resources hold valid credentials to do so.
- There is a well-defined association between Users and Roles/Privileges.
- Role and permission related metadata is protected from tampering.

SR3.1 General Access Control Design & SR3.2 Operation Level Access Control & SR3.3 Other Access Control Considerations.

Table D.6: Access Control Verification Requirements - General Access Control Design (SR3.1), Operation Level Access Control (SR3.2), Other Access Control Considerations (SR3.3).

#RID	Description	OWASP	NIST
SR3.1	The application should enforce access control rules on a trusted service layer. This is especially important if client-side access control is present and could be bypassed.	4.1.1	-
SR3.2	The application must verify that, unless specifically authorized, end users cannot manipulate information used by the access control. This pieces of information include: user attributes, data attributes, policy information.	4.1.2	-
SR3.3	The application should enforce the “principle of least privilege”: users should only be able to access functions, data files, services and, in general, resources for which they possess specific authorization. Note: this implies protection against elevation of privilege and spoofing.	4.1.3	-
SR3.4	The application should enforcement the “principle of deny by default”: new users or roles start with minimal or no permission and do not receive access to new features until access is explicitly assigned.	4.1.4	-
SR3.5	The application’s access controls must fail securely when an exception occurs.	4.1.5	-
SR3.6	Within the application, sensitive data and API must be protected against direct object attacks that target creation, reading, updating and deletion of records. These types of attack include deletion of all records or tampering of existing data.	4.2.1	-
SR3.7	The application should enforce a strong anti-CSRF and anti-automation mechanism in order to protect authentication.	4.2.2	-
SR3.8	Within the application, directory browsing must be disabled unless deliberately desired. The application should not allow discovery or disclosure of files or directory metadata (e.g. Thumbs.db, .DS_Store, .git or .svn folders).	4.3.2	-

SR4 Validation, Sanitization, Encoding Verification Requirements

In web-application security the most common weakness is the failure to properly validate input to the system itself. It does not matter if the input is coming from the client or from the environment, if there is not a prior encoding then there is a risk and this leads to almost all of the significant vulnerabilities in web-applications: Cross-Site Scripting (XSS), SQL Injection, interpreter injection, etc.

There are some high-level requirements that are fundamental for the system:

- Input validation and output encoding architecture have an agreed pipeline to prevent injection attacks.
- Input data is strongly typed, validated (e.g., checked for range and length) or at least sanitized and filtered.
- Output data is encoded or escaped as close to the interpreter as possible.

Output encoding is more important than ever and, since providing robust input validation is often difficult, the use of safe API (e.g., parameterized queries, auto-escaping templates, etc) is critical to the security of the application.

SR4.1 Input Validation Requirements

Implementing a proper input validation control can eliminate more than 90% of all injection attacks. This can be achieved in several ways: strong data typing, positive whitelisting, length and range checks, etc.

A secure input validation is built during application architecture design, code development, and testing development.

Table D.7: Validation, Sanitization, Encoding Verification Requirements - Input Validation (SR4.1).

#RID	Description	OWASP	NIST
SR4.1	The application should be protected against HTTP parameter pollution attacks.	5.1.1	-
SR4.2	The application, or the framework on which it is based, should include a protection against mass parameter assignment or unsafe parameter assignment (e.g. marking a field as private or similar).	5.1.2	-
SR4.3	The application should verify that all inputs (e.g., REST requests, HTML form fields, URL parameters, HTTP Headers, etc.) are validated by using positive validation (i.e. whitelisting).	5.1.3	-
SR4.4	Within the application is mandatory that structured data is strongly typed and validated against a defined schema. Perform stronger checks if possible (e.g. allowed characters, length, patterns, etc.).	5.1.4	-
SR4.5	The application features of URL redirection and URL forwarding should use a whitelist of allowed destinations. As a viable alternative, a warning can be shown to users in case of redirection to potentially untrusted content.	5.1.5	-

SR4.2 Sanitization and Sandboxing Requirements

Table D.8: Validation, Sanitization, Encoding Verification Requirements - Sanitization and Sandboxing (SR4.2).

#RID	Description	OWASP	NIST
SR4.6	The application must sanitize received inputs. Great care should be applied to input such as HTML, scriptable content, expression template content (e.g. markdown, css, BBCode, html, etc.).	5.2.1 / 5.2.3, 5.2.5 / 5.2.8	-
SR4.7	The application must not use any eval() or other dynamic code execution features. In case there are no alternatives, any user input included needs to be sanitized or sandboxed before being executed.	5.2.4	-

SR4.3 Output encoding and Injection Prevention Requirements

When generating the output is important to encode it as close as possible to the expected interpreter. This is a critical issue and failing it to do it will result in an insecure, injectable and unsafe application.

Please take into account that using parameterized queries and escaping SQL is rarely sufficient because several elements of a query cannot be escaped (e.g., table and column name).

Please also take into account that the SVG format should be handle with extreme care since ECMA scripts can be included inside it. SVG can be a vector for XSS attacks and as such SVG upload should be disabled unless required. In case SVG upload is required some solutions must be implemented to manage it safely.

Table D.9: Validation, Sanitization, Encoding Verification Requirements - Output encoding and Injection Prevention (SR4.3).

#RID	Description	OWASP	NIST
SR4.8	The application's output encoding should be relevant for the interpreter and a context should be provided (e.g. use specific encoders for HTML values, HTML attributes, URL parameters, etc.).	5.3.1	-
SR4.9	The application's output encoding should preserve user's chosen character set and locale.	5.3.2	-
SR4.10	The application must make us of data selection and database queries that are based on parameterized queries, ORM, entity frameworks in order to prevent database injection attacks.	5.3.4	-
SR4.11	The application must be protected versus attacks such as XSS (i.e. reflected, stored, DOM based), SQL Injection, JS and JSON Injection (remove JS include, CSP bypass, eval attacks), LDAP injection, OS command injection, Local File Inclusion (LFI) and Remote File Inclusion (RFI), XPath injection and XML injection.	5.3.3, 5.3.5 / 5.3.10	-

SR4.5 Deserialization Prevention Requirements

Table D.10: Validation, Sanitization, Encoding Verification Requirements - Deserialization Prevention (SR4.4).

#RID	Description	OWASP	NIST
SR4.12	The application should correctly restrict XML parsers and force them to use a restrictive configuration in order to ensure that unsafe features (e.g. external entities resolution) are disabled.	5.5.2	-
SR4.13	Both the application and used third party libraries should avoid the deserialization of untrusted data or do it in a secure and "protected" way (e.g. JSON, XML and YAML parsers).	5.5.3	-
SR4.14	Verify that the application parses JSON using JSON.parse and do not uses eval(). This must be checked both for browser parsing and JavaScript based backends.	5.5.4	-

SR5 Stored Cryptography Verification Requirements

There are some high-level requirements that are fundamental for the system:

- All software modules that manage cryptography must fail in a secure manner and errors must be handled correctly.
- Random number generation must be managed with a suitable generator.
- Key access must be managed in a secure way.

SR5.1 Data Classification & SR5.2 Algorithms.

Table D.11: Stored Cryptography Verification Requirements – Data Classification (SR5.1), Algorithms (SR5.2).

#RID	Description	OWASP	NIST
SR5.1	Regulated private data must be stored in an encrypted format while at rest. This include personally identifiable information (PII), sensitive personal information, data subject to EU's GDPR.	6.1.1	-
SR5.2	Regulated health data must be stored encrypted while at rest. This include medical records, medical device details and de-anonymized research records.	6.1.2	-
SR5.3	All cryptographic modules should fail securely and should be chosen and configured according to industry/government proven guidelines. This include the use of approved cryptographic algorithms, random number generation, modes and libraries over custom made solutions.	6.2.1, 6.2.2, 6.2.3, 6.3.1	-
SR5.4	All configuration related to "cryptography" (e.g., hashing and encryption algorithms, cyphers, etc.) should be easily and quickly reconfigured and upgraded at any time to protect against cryptographic breaks.	6.2.4	-
SR5.5	Verify that a secrets management solution (e.g. a key vault) is used to securely manage secrets (i.e. create, store, control access, destroy). This include the use of a similar approach for key material that should not be exposed to the application.	6.4.1, 6.4.2	

SR6 Error Handling and Logging Verification Requirements

Error handling and logging is of primary importance when it comes to provide useful information for the users, administrators, quality assurance (QA) and incident response teams. High quality logs, low in "noise" but high in "signal", will often contains sensitive data and as a consequence they need to be protected as per local privacy laws or directives.

There are some high-level requirements that are fundamental for the system:

- Do not collect or log sensitive information unless specifically required.
- Ensure that all logged information is handled in a secure way and protected as per its data classification.
- Ensure that all logs are not store forever and clearly define a log lifetime that needs to be as short as possible.
- Do not disclose unnecessary information when the application has an internal error.

SR6.1 Log Content Requirements & SR6.2 Log Protection Requirements

Logging sensitive information can be a source of problems, a danger and increase the complexity of the system. Every log containing sensitive information should be considered sensitive itself and, as such, it needs to be encrypted and become subject to retention policies and must be disclosed in security audits. It is important to ensure that only the necessary information is kept in logs and sensitive or personally identifiable information are never stored.

Logs that can be trivially modified or deleted are useless for investigations and prosecutions and disclosure of logs can expose inner details about application and stored data. Logs should be handled and managed with care.

Table D.12: Error Handling and Logging Verification Requirements – Log Content Requirements (SR6.1), Log Protection Requirements (SR6.2).

#RID	Description	OWASP	NIST
SR6.1	The application must not log credentials and session's tokens must not be stored unless in a hashed form.	7.1.1	-
SR6.2	The application must not log sensitive data as defined under relevant privacy laws and security policies.	7.1.2	-
SR6.3	The application must log security relevant events (i.e. successful and failed authentication, access control failures, deserialization failures, input validation failures).	7.1.3	-
SR6.4	The application's logs must include the necessary information that would allow for a detailed investigation of the timeline when an event happens.	7.1.4	-
SR6.5	The application's logs must be protected from unauthorized access and modification.	7.3.3	-
SR6.6	The log's time source should be synchronized to the correct time and time zone. It is strongly suggested to log only in UTC if systems are global in order to assist in post-incident forensic analysis.	7.3.4	-

SR6.3 Error Handling

Table D.13: Error Handling and Logging Verification Requirements – Error Handling (SR6.3).

#RID	Description	OWASP	NIST
SR6.7	The application must return/show a generic message when an unexpected or security sensitive error occurs. If required, a unique ID can be provided in order to support during the investigation.	7.4.1	-
SR6.8	The application should use exception handling (i.e. or a functionally equivalent software solution) in all the codebase to account for expected and unexpected error conditions.	7.4.2	-

SR7 Data Protection Verification Requirements

Data protection focuses on three key elements: Confidentiality, Integrity and Availability (CIA).

There are some high-level requirements that are fundamental for the system:

- Confidentiality – data should be protected from unauthorized observation or disclosure both in transit and when stored.
- Integrity – data should be protected from being maliciously created, altered or deleted by unauthorized parties.
- Availability – data should be available to authorized users as required.

SR7.1 General Data Protection

Table D.14: Data Protection Verification Requirements – General Data Protection (SR7.1).

#RID	Description	OWASP	NIST
SR7.1	The application must protect sensitive data from being cached in server components (e.g. load balancers, application caches, etc.) and all cached sensitive data must be protected from unauthorized access or being purged/invalidated after being accessed by an authorized party.	8.1.1, 8.1.2	-
SR7.2	Regular backups of important data should be performed and stored securely to prevent data theft and data corruption.	8.1.5, 8.1.6	-

SR7.2 Client-side Data Protection

Table D.15: Data Protection Verification Requirements – Client-side Data Protection (SR7.2).

#RID	Description	OWASP	NIST
SR7.3	The application must set sufficient anti-caching headers to prevent browsers from caching sensitive data.	8.2.1	
SR7.4	The application should not store in client-side storage (e.g. HTML5 local storage) any piece of sensitive data or PII	8.2.2	
SR7.5	The application must clear authenticated data from client storage (e.g. the browser's DOM) after the session is terminated.	8.2.3	

SR7.3 Sensitive Private Data

This Section focuses on the protection of sensitive data from being created, read, update or deleted in discreet quantities or in bulk quantities.

These requirements are strictly linked to those regarding Access Control and to privacy and GDPR compliance. Privacy regulations and laws (e.g. GDPR) deeply affect in a direct way how software applications must approach the problem of sensitive and personal information storage, use and transmission.

Table D.16: Data Protection Verification Requirements – Sensitive Private Data (SR7.3).

#RID	Description	OWASP	NIST
SR7.6	The application must handle communication of sensitive data in the HTTP message body or header while query string parameters from any HTTP verbs must not contain sensitive data.	8.3.1	-
SR7.7	The application must provide users with a clear explanation regarding data collection and use, provide an opt-in consent for the use of any data and allow users to remove or export their data on demand.	8.3.2, 8.3.3	-
SR7.8	All sensitive data created and processed by the application must have been identified and there must be a policy in place regarding how to deal with sensitive data.	8.3.4	-

SR8 Communications Verification Requirements

There are some high-level requirements that are fundamental for the system:

- Strong encryption (e.g. TLS) is always used, regardless of the sensitivity of the data being transmitted.
- The most recent or leading configuration advice is used in regards of preferred algorithms.
- Deprecated and insecure algorithms are disabled, weak algorithms are used only as last resort.

SR8.1 Communication Security Requirements & SR8.2 Server Communications Security Requirements

Table D.17: Communications Verification Requirements – Communication Security (SR8.1), Server Communications Security (SR8.2)

#RID	Description	OWASP	NIST
SR8.1	The application must use secure TLS for all the communication with the client and never fall back to insecure or unencrypted protocols.	9.1.1	-
SR8.2	The application must disable all old versions of SSL and TLS such as SSLv2, SSLv3, TLS 1.0 and TLS 1.1	9.1.3	
SR8.3	The application must use trusted TLS certificates for the communication to and from the server. If internally generated or self-signed certificates are used, the application server must be configured to only trust specific internal CAs and self-signed certificates.	9.2.1	
SR8.4	The application must use secure TLS for all the inbound and outbound connections (e.g. ports managements, monitoring, authentication, API, database connection, etc.).	9.2.2	

SR9 Malicious Code Verification Requirements

Every complex software usually leverages several third-party libraries, and this can introduce weaknesses into the system.

Great care should be taken while selecting third party libraries and software components. Once an application is deployed, malicious code can still often be inserted, and this practically can turn every third-party library into a piece of malicious code. The application needs to be protected from common attacks: unsigned code execution from untrusted sources, sub-domain takeover, etc.

SR9.1 Deployed Application Integrity Controls

Table D.18: Malicious Code Verification Requirements – Deployed Application Integrity Controls (SR9.1).

#RID	Description	OWASP	NIST
SR9.1	The application's dependencies and runtime environment must be updated only with software obtained over secure channels and digitally signed. The update code must validate the digital signature before installing or executing the update.	10.3.1	-
SR9.2	The application must include an integrity protection (e.g. code signing) and must not load or execute code from untrusted sources (e.g. loading includes, modules, plugins, etc).	10.3.2	-

SR10 File and Resource Verification Requirements

There are some high-level requirements that are fundamental for the system:

- Untrusted file data must be handled in a secure manner.
- Untrusted file data obtained from untrusted sources must be stored outside the web root and with limited permissions.

SR10.1 File Upload Requirements & SR10.2 File execution Requirements & SR10.3 File Storage Requirements & SR10.4 File Download Requirements & SR10.5 SSRF Protection Requirements

Table D.19: File and Resource Verification Requirements – File Upload (SR10.1), File execution (SR10.2), File Storage (SR10.3), File Download (SR10.4), SSRF Protection (SR10.5).

#RID	Description	OWASP	NIST
SR10.1	The application must not accept large files that could fill up the storage space and/or cause DoS attacks.	12.1.1	-
SR10.2	The application must validate or ignore user-submitted filename metadata to prevent and block several threats: path traversal, local and remote file inclusion (LFI, RFI), reflective file download (RFD).	12.3.1 / 12.3.4	-
SR10.3	The application must store files obtained from untrusted sources outside of the web root, with limited permissions and, preferably, with a strong validation.	12.4.1	-
SR10.4	The application must serve only files with specific file extensions to prevent unintentional information and source code leakage (e.g. no backup files .bak, or compressed files should be served).	12.5.1	-
SR10.5	The application web server's configuration must include a whitelist of resources or systems to which the server can send requests or load data/files from.	12.6.1	-

SR11 API and Web Service Verification Requirements

The application developed within WP5 is expected to communicate with other external components developed (i.e., developed within the IDEA-FAST project and/or already existent). It is assumed that all verified applications that can communicate with the DMP will do it solely by using a trusted service layer APIs (e.g. JSON, XML, GraphQL).

There are some high-level requirements that are fundamental for the system in regards to this communication:

- There must be an adequate authentication, session management and authorization of all web services that are allowed to communicate with the DMP.
- There must be an input validation for all parameters that transit from a lower to a higher trust level.
- Effective security controls for all API types (i.e., cloud data transfer and Serverless API also require security controls).

It should be noted that the requirements included in this Section should be considered combined with other previously written requirements.

Some additional notes are useful, and important, in regards of JSON schema validation. There are no current universally accepted standards regarding JSON schema validation albeit some projects, such as OpenAPI Specification (OAS, please refer to https://swagger.io/specification/), are seeing a constantly growing recognition.

A natural consequence of a lack of universally accepted standard is a lack of mature software solutions to enforce schema validation. When considering using JSON schema validation, which is best practice for SOAP web services, consider these additional strategies in combination with standard JSON schema validation:

- Parse validation – check the JSON object for missing or extra elements.
- Standard Input validation – check the JSON object using standard approaches as data type check, data format check, length, etc.
- Formal Schema validation – check the JSON object against an expected schema.

The approach used for schema validation should be adapted to the software architecture and is expected to be different for validation of external inbound data (i.e., incoming JSON that are submitted from external sources) and for internal data transfer (i.e., data that is generated due to the GraphQL layer).

SR11.1 Generic Web Service Security Verification Requirements & SR11.2 RESTful Web Service Verification Requirements & SR11.3 GraphQL and other Web Service Data Layer Security Requirements

Table D.20: API and Web Service Verification Requirements – Generic Web Service Security Verification (SR11.1), RESTful Web Service Verification (SR11.2), GraphQL and other Web Service Data Layer Security (SR11.3).

#RID	Description	OWASP	NIST
SR11.1	All the application components must use the same encoding and parsers in order to avoid parsing based attacks (i.e. typically associated with SSRF and RFI attacks)	13.1.1	-
SR11.2	The application must limit the access to administration and management functions to authorized administrators.	13.1.2	-
SR11.3	The application must not expose sensitive information (e.g. API key, session tokens, etc.) in the URLs.	13.1.3	-
SR11.4	The application must enable for each user only the RESTful HTTP methods that are allowed according to the user permissions (e.g. prevent all users from using DELETE or PUT on protected API or resources).	13.2.1	-
SR11.5	The application must enforce JSON schema validation before accepting input.	13.2.2	-
SR11.6	The application must enforce protection against Cross-Site Request Forgery (CSRF) by using a standard software solution (i.e. framework solution or well know library).	13.2.3	-
SR11.7	The application should enforce query whitelisting or a combination of depth/amount limiting in order to prevent GraphQL expression denial of service (DoS) as a result of expensive, nested queries.	13.4.1	-
SR11.8	The application should implement authorization logic at the business logic layer instead of GraphQL layer.	13.4.2	-

SR12 Configuration Verification Requirements

There are some high-level requirements that are fundamental for the system:

- The application has a secure, repeatable, automatable build environment.
- Hardened third party library, dependency configuration management in order to guarantee that out of date and insecure components are not included in the application.
- A secure-by-default configuration.

Failure to keep up to date insecure or outdated dependencies is the root cause of the largest and most expensive attacks to date and as a result we can safely say that dependency management is critical for the safety of the application.

SR12.1 Dependency & SR12.2 Unintended Security Disclosure Requirements

Table D.21: Configuration Verification Requirements – Dependency (SR12.1), Unintended Security Disclosure (SR12.2).

#RID	Description	OWASP	NIST
SR12.1	The application must leverage up to date libraries and components.	14.2.1	-
SR12.2	The application source code must not include any unrequired third party library files (e.g. unneeded features, documentation, samples, configuration, etc.).	14.2.2	-
SR12.3	The application must include a Subresource Integrity (SRI) check to validate the integrity of any asset hosted on external source such as content delivery networks (CDN), external providers. This applies for example to JavaScript libraries, CSS stylesheets, web fonts.	14.2.3	-
SR12.4	The application should use only components from pre-defined, trusted and continually maintained repositories.	14.2.4	-
SR12.5	The application documentation must include a catalogue of all third party libraries in use within the application.	14.2.5	-
SR12.6	The application server and framework error messages must not include unintended security disclosures. Only user actionable, customized responses should be delivered.	14.3.1	-
SR12.7	The application server must run with the “debug mode” disabled in production in order to prevent the disclosure of security relevant information such as: debug features, developer console, etc.	14.3.2	-