

D4.3 Final TRUMPET platform and dashboards report

Lead Author: [MD Nurujjaman Nur/TVS]

With contributions from: [Khaled MD Imran/TVS, Ashadul Hoque/TVS]

Reviewer: [Zeev Pritzker/ARTEEVO, Naoufal Elazzouzi/GRAD, Alberto Pedrouzo Ulloa/UVIGO]

Deliverable nature	Report (R)
Dissemination level	Public (PU)
Delivery date	30-10-2025
Version	0.8
Total number of pages	112
Keywords	Federated Learning, Privacy-Enhancing Technologies, Differential Privacy, Homomorphic Encryption, Secure Multi-Party Computation, Armored Federated Learning, HL7 FHIR



TRUStworthy Multi-site Privacy Enhancing Technologies



TRUStworthy Multi-site Privacy Enhancing Technologies

EXECUTIVE SUMMARY

TRUMPET delivers a privacy-first federated AI platform that lets researchers learn from siloed, multi-site clinical data without moving raw patient data across borders. The platform combines a Cloud node and deployable Data Owner nodes, each exposing researcher and data-owner dashboards, a Centralised Access Control, and core services for study setup, privacy control, and federated orchestration. The final architecture uses layered components and microservices with a monorepo codebase and well-defined standard models, designed for modularity, swap-in PETs, and maintainability.

Our technical approach "armours" standard Federated Learning with privacy-enhancing technologies—Homomorphic Encryption, Secure Multi-Party Computation, and Differential Privacy—applied to model updates and integrated with aggregation pipelines. This is coupled with a novel privacy-measurement concept and controls that bound disclosure over the life of a study. Together, these features target realistic threat models, including honest-but-curious aggregation and man-in-the-middle threats, while keeping the researcher's workflow familiar and productive.

The platform's two-sided design serves both demand and supply: Researchers can discover published datasets, define cohorts, select AI models, and launch FL training from an integrated model library; Data Owners can publish dataset metadata, set privacy policies, and operate local nodes connected to their HAPI-FHIR stores. The dashboards and process flows have been iteratively built and validated with the partners, emphasising usability and explainability.

Piloting of the platform has focused on oncology scenarios and radiotherapy workflows (NSCLC, HNC, SBRT). We used the partners' real-world datasets under hospital governance, ran small internal validations first, and then prepared joint pilot plans and readiness checks. This staged approach—already foreseen in the work plan—helped to expose usability and data-quality issues early, guided minor model updates, and improved result matrices without exposing sensitive details.

Security was addressed end-to-end. We integrated privacy hooks aligned with the metric work, performed independent privacy-focused penetration testing as planned, and resolved all identified issues with patches and hardening of identity, network, and data-protection controls. The project explicitly budgeted for external, independent pentesting to validate resilience to re-identification attacks; those tests inform our remediation playbook and risk register.

Governance and compliance were embedded from the start. The design aligns with GDPR and current EU cybersecurity priorities, aiming to provide measurable guarantees for data



TRUstworthy Multi-site Privacy Enhancing Technologies

owners and a path to eventual certification of FL implementations through a privacy metric and measurement tooling. The platform abstracts cross-border data heterogeneity while keeping the process local, supporting EU data-space ambitions.

DOCUMENT INFORMATION

Grant agreement No.	101070038	Acronym	TRUMPET
Full title	TRUstworthy Multi-site Privacy Enhancing Technologies		
Call	HORIZON-CL3-2021-CS-01-04		
Project URL	https://cordis.europa.eu/project/id/101070038		
EU project officer	Ioannis Askoxylakis		

Deliverable	Number	D4.3	Title	Final TRUMPET platform and dashboards report
Work package	Number	WP4	Title	TRUMPET platform development, pilot and validation
Task	Number	T4.2	Title	Design and Develop TRUMPET Platform and Dashboards

Date of delivery	Contractual	30/09/2025	Actual	30/10/2025
Status	version 0.8	<input checked="" type="checkbox"/> Final version		
Nature	<input checked="" type="checkbox"/> R <input type="checkbox"/> DEM <input type="checkbox"/> DMP <input type="checkbox"/> DEC <input type="checkbox"/> ETHICS <input type="checkbox"/> OTHER			
Dissemination level	<input checked="" type="checkbox"/> Public <input type="checkbox"/> Sensitive			

Authors (partners)	TVS
Responsible author	Nurujjaman Nur nur@technovativesolutions.co.uk

Summary (for dissemination)	<i>This document outlines the TRUMPET platform and its dashboards. It describes the architecture, APIs, FHIR-based data flows, PET-hardened federated learning, enhanced security measures, containerised deployments, study management, cohorting, training, and results review.</i>
Keywords	<i>Federated Learning, Privacy-Enhancing Technologies, Differential Privacy, Homomorphic Encryption, Secure Multi-Party Computation, Armored Federated Learning, HL7 FHIR</i>

VERSION LOG			
Issue Date	Rev. No.	Author	Change
10-10-2025	0.1	MD Nurujjaman Nur (TVS)	Initial draft
11-10-2025	0.2	Khaled MD Imran (TVS)	Updated high and low level system architecture
12-10-2025	0.3	Ashadul Hoque (TVS)	Modified several sections and sub-sections
15-10-2025	0.4	Zeev Pritzker (ARTEEVO)	Internal review
20-10-2025	0.5	Naoufal Elazzouzi (GRAD)	Internal review
20-10-2025	0.6	Alberto Pedrouzo Ulloa (UVIGO)	SAB review and feedback
23-10-2025	0.7	Khaled MD Imran (TVS)	Refactor according to the review and feedback
29-10-2025	0.8	MD Nurujjaman Nur (TVS)	Prepare the final version corresponding internal and SAB feedback

TABLE OF CONTENTS

1	Objectives.....	16
1.1	WP4 in TRUMPET	16
1.2	Deliverables & Milestones Map.....	17
2	Requirements Traceability	19
2.1	Stakeholders & Roles.....	19
2.2	Functional Requirements (FR).....	19
2.3	Non-Functional Requirements (NFR)	20
2.4	Compliance & Standards	20
3	System Architecture.....	21
3.1	Cloud Architecture.....	22
3.1.1	Cloud Workflows at a Glance.....	24
3.1.2	Low-Level Design (Modules, APIs, Data Models)	25
3.1.3	Folder Structure and Responsibilities.....	33
3.2	Data-Owner Node Architecture.....	37
3.2.1	Data-Owner Node Workflows at a Glance.....	40
3.2.2	Low-Level Design (Modules, APIs, Data Models)	41
3.3	Cloud-Node Interaction	57
3.4	SDLC & Co-Creation Cadence	58
4	Data Acquisition & Integration	60
4.1	Ingestion Pipelines.....	60
4.2	Adapters & Connectors.....	61
4.3	Operationalising FHIR.....	62
5	Platform & Dashboards	63
5.1	Researcher Dashboard.....	63
5.2	Data Owner Control Panel.....	65
5.2.1	Dashboard on Data Owner Node	65
5.2.2	Dashboard on Cloud.....	68
5.3	End-to-End User Flows.....	70
5.3.1	Researcher flow (Cloud Dashboard)	70
5.3.2	Data Owner flow (Local Node + Cloud Admin)	77
5.3.3	Governance/Admin flow (Cloud).....	84
5.3.4	Federated training round-trip (system view).....	89
5.4	API Design & Versioning.....	91
5.4.1	API styles and layering.....	91
5.4.2	Resource model and response rules	92
5.4.3	Security & Access	92
5.4.4	Example version map.....	92
5.5	Database Design & Multi-Site Considerations	93
5.5.1	Logical split: Cloud vs Data Owner	93
5.5.2	Cloud relational model	93
5.5.3	Data Owner persistence	93
5.5.4	How we store and organise datasets.....	93
6	Privacy Measurement & Validation Hooks	95
6.1	FLCore Integration and Interfacing.....	95
6.2	Test Data & Attack Simulations	97
6.3	Result Capture & Reporting.....	97
7	Deployment & Environments.....	99
7.1	Topologies (Cloud / On-Premises).....	99



TRUStworthy Multi-site Privacy Enhancing Technologies

7.1.1	Cloud (TRUMPET Cloud)	99
7.1.2	Data Owner Node (On-Premises)	99
7.1.3	End-to-End Flow	100
7.2	Packaging, Configuration, Secrets — with Deployment Guides	100
7.2.1	Data Owner Node (On-Prem)	100
7.2.2	TRUMPET Cloud (Hosted)	102
7.3	Access & Onboarding	104
7.3.1	Cloud Access	104
7.3.2	Data Owner Onboarding	104
7.3.3	Researcher Onboarding	105
8	Pilot Preparation & Execution	106
8.1	Case Studies (HNC / NSCLC / SBRT)	106
8.2	Pilot Preparation & Execution	107
9	Security, Pen-testing & Deployment Hardening	109
9.1	Pen-testing Summary	109
9.2	Vulnerability Management in SDLC	110
10	Conclusions	111

The table of contents is generated automatically if the pre-defined styles are used. Do not try to write it yourself but use the option "Update field" in the contextual menu when clicking the right button of the mouse.

DE

LIST OF FIGURES

Figure 1: Deliverables & Milestones Map	17
Figure 2: TRUMPET Platform Components	21
Figure 3: Cloud Architecture.....	22
Figure 4: Cloud workflows at a glance	25
Figure 5: Low Level Dependency Diagram.....	26
Figure 6: Relational Database Schema Diagram of Cloud	31
Figure 7: Data Owner Node Architecture	38
Figure 8: Data Owner Node UI flows	41
Figure 9: Data Owner Node Storage System with Relations	43
Figure 10: Data owner Node Dataset relation to QuestionnaireResponse Records	43
Figure 11: Cloud–Node Study Orchestration.....	58
Figure 12: Data Flows inside Data Owner Node	60
Figure 13: TRUMPET Cloud Researcher Dashboard.....	63
Figure 14: Researcher Dashboard Menu.....	64
Figure 15: Data Owner Node on premise Admin Dashboard	66
Figure 16: Data Owner Node on premise dashboard menu	67
Figure 17: Data Owner dashboard on TRUMPET Cloud	68
Figure 18: Researcher Flow - Cloud - Browse Available Datasets.....	70
Figure 19: Researcher Flow - Cloud - View Dataset Details	71
Figure 20: Researcher Flow - Cloud - View Dataset Metadata	71
Figure 21: Researcher Flow - Cloud - View Dataset Statistics.....	72
Figure 22: Researcher Flow - Cloud - Create Study	73
Figure 23: Researcher Flow - Cloud - Create Study Agreement	74
Figure 24: Researcher Flow - Cloud - View Real Time Training Notifications.....	75
Figure 25: Researcher Flow - Cloud - View Training Result	76
Figure 26: Researcher Flow - Cloud - Download Training Result	77
Figure 27: Data Owner Flow - Cloud - Read Setup Instruction.....	78
Figure 28: Data Owner Flow - Cloud - Read Dataset Upload Instruction.....	79
Figure 29: Data Owner Flow - on Premise Node - Create Dataset Resource	80
Figure 30: Data Owner Flow - on Premise Node - Generate Dataset Statistics.....	81
Figure 31: Data Owner Flow - on Premise Node - Publish Dataset to Cloud.....	82
Figure 32: Data Owner Flow - Cloud - View Pending Training Request.....	82
Figure 33: Data Owner Flow - Cloud - Approve or Reject Training Request.....	83
Figure 34: Governance Admin Flow - Cloud - View Organisation List.....	84
Figure 35: Governance Admin Flow - Cloud - Invite Organisation	85
Figure 36: Governance Admin Flow - Email Client - Email Invitation is Sent to the Organisation	86
Figure 37: Governance Admin Flow - Cloud - Organisation Admin Fills up Registration Form	87
Figure 38: Governance Admin Flow - Cloud - View Pending Organisation Registrations.....	88
Figure 39: Governance Admin Flow - Cloud - View Pending Organisation Details	88
Figure 40: Governance Admin Flow - Cloud - View Pending Organisation Docs and Approve or Reject	89
Figure 41: FL System Flow.....	91
Figure 42: Adapters to implement partners' services	96
Figure 43: Deployment Architecture - Cloud and On-Premises Topologies	100

The list of figures is generated automatically if the pre-defined styles are used. Do not try to write it yourself but use the option "Update field" in the contextual menu when clicking the right button of the mouse.

LIST OF TABLES

Table 1: Cloud Dashboard User Roles.....	23
Table 2: Cloud Dashboards - Roles and Key Functionalities	23
Table 3: TRUMPET Cloud Core Services.....	28
Table 4: Cloud Entities.....	30
Table 5: Cloud Entity Relations	30
Table 6: Cloud Resource Based Endpoints.....	32
Table 7: Cloud Non-resource Endpoints.....	33
Table 8: Followed REST API response codes and their meanings.....	33
Table 9: Cloud Folder Responsibilities at a Glance.....	37
Table 10: Data Owner Node — Components at a Glance	39
Table 11: Data Owner Node — Dashboard Areas & Key Functionality.....	39
Table 12: Data Owner Node — Dataset & Access Model (At a Glance).....	42
Table 13: APIs of data owner node.....	57
Table 14: API Versioning	92

The list of tables is generated automatically if the pre-defined styles are used. Do not try to write it yourself but use the option “Update field” in the contextual menu when clicking the right button of the mouse.

ABBREVIATIONS AND ACRONYMS

AFL — Armored Federated Learning.
Agg — Aggregator.
AI — Artificial Intelligence.
ALK — Anaplastic Lymphoma Kinase.
API — Application Programming Interface.
B2C — Business to Consumer.
CAC — Centralized Access Control.
CI/CD — Continuous Integration and Continuous Delivery.
DAC — Data Access Committee.
DDD — Domain-Driven Design.
DO — Data Owner.
DP — Differential Privacy.
DPA — Data Protection Agency.
DPIA — Data Protection Impact Assessment.
DPO — Data Protection Officer.
EHR — Electronic Health Records.
EGFR — Epidermal Growth Factor Receptor.
ETL — Extract, Transform, Load.
FAIR — Findable, Accessible, Interoperable and Re-Usable.
FCFM — Fibered Confocal Fluorescence Microscopy.
FHE — Fully Homomorphic Encryption.
FHIR — Fast Healthcare Interoperability Resources.
FL — Federated Learning.
FMA — Federated Model Aggregator.
FMM — Federated Learning Management module.
GDPR — General Data Protection Regulation.
gRPC — Remote Procedure Call framework (gRPC).
HE — Homomorphic Encryption.
HL7 — Health Level Seven.
HNC — Head and Neck Cancer.
HAPI FHIR — HAPI FHIR server (HL7 FHIR implementation).
IoT — Internet of Things.
LWE — Learning With Errors.
MKHE — Multi-Key Homomorphic Encryption.
ML — Machine Learning.
NSCLC — Non-Small-Cell Lung Cancer.
PET / PETs — Privacy Enhancing Technologies.
PPFL — Privacy-Preserving Federated Learning.
PRICOM — Privacy Compliance Levels.



TRUstworthy Multi-site Privacy Enhancing Technologies

REST — Representational State Transfer.
RLWE — Ring Learning With Errors.
ROC — Receiver Operating Characteristic.
RTF — Requirement Task Force.
SBRT — Stereotactic Body Radiation Therapy.
SDLC — Software Development Life Cycle.
SHE — Somewhat Homomorphic Encryption.
SMPC — Secure Multi-Party Computation.
SNOMED CT — Clinical terminology system (SNOMED CT).
SCLC — Small Cell Lung Cancer.
TEE — Trusted Execution Environments.
UI — User Interface.
URL — Uniform Resource Locator.
UX — User Experience.
WP — Work Package(s).
ZKP — Zero-Knowledge Proof.

DEFINITIONS

ABAC (Attribute-Based Access Control) — Authorisation model in the cloud dashboards that scopes access by user/resource attributes (beyond simple roles).

Adapter Pattern — Our standard way to wrap third-party systems (e.g., FLCore, HAPI) behind internal interfaces so we can swap vendors without rewrites.

Advertised Dataset Service — Cloud service catalogue datasets published by Data Owners that expose only metadata/statistics, not raw data.

Aggregator (FL Aggregator / Analytics Aggregator) — Cloud components that (i) combine model updates during FL rounds and (ii) maintain privacy-safe statistics for dataset assessment.

API Gateway (CAC at the edge) — Single cloud entry point for routing, token validation, CORS, observability, and API versioning across services.

API Versioning — Stable REST surfaces (e.g., Cloud Public API, FL Control API) are kept additive and tracked per surface (v1).

CAC (Centralised Access Control) — Cloud edge shell fronting microservices; handles routing, auth, CORS and versioning; brokers calls to backends.

Canonical Model (FHIR-first) — The common representation for clinical data across sites; on-prem repositories are constrained to FHIR Questionnaire/QuestionnaireResponse.

Communication API (Cloud↔DO) — Authenticated, audited control plane for invitations, approvals and training messages between Cloud and Data Owner nodes.

Data Owner (DO) — A hospital/organisation operating an on-prem node that keeps raw clinical data local, publishes only metadata, and runs local training.

Data Owner Dashboard — On-prem UI to upload FHIR data, define/publish datasets, generate stats, manage tokens, and run/approve studies.

Data Owner Node — On-prem stack (HAPI FHIR + proxy, DO services, local FL server, Traefik) inside the hospital network with outbound-only flows.

Dependency Injection (DI) — A Technique we use to wire ports/adapters at runtime and keep integrations isolated and testable.

Differential Privacy (DP) — PET that injects calibrated noise into stats/model updates; enforced pre-/per-round via privacy hooks.

Domain-Driven Design (DDD) — Bounded-context approach used to structure cloud services and keep complex FL workflows predictable.

Federated Learning (FL) — Training across multiple DO nodes without sharing raw data; only privacy-screened updates leave each site.

Federated Model Management — Cloud orchestration endpoints that coordinate studies and aggregate FL updates into results.

FL Control API — REST surface coordinating study orchestration between Cloud and the FL adapter/engine.



TRUStworthy Multi-site Privacy Enhancing Technologies

FLCore (Local FL Platform Server) — Replaceable FL engine integrated via an adapter; internal management calls only, with the communication endpoints exposed and token-protected.

FHIR (HL7® FHIR®) — Canonical healthcare data standard used operationally; HAPI FHIR is our on-prem implementation.

Governance Dashboard — Cloud UI for approvals, role administration, node registration and revocation flows.

HAPI FHIR — Open-source HL7 FHIR server deployed at each DO to store clinical resources; shielded by a HAPI Proxy.

HAPI Proxy — Reverse proxy in front of HAPI FHIR that authenticates requests, isolates the FHIR endpoint, and can use a token cache (e.g., Redis).

Hexagonal Architecture (Ports & Adapters) — All external tech (queues, storage, email, caches) sits behind adapters to avoid lock-in.

HL7 FHIR Profiles/Indexes — Conformance and indexing rules enforced locally before datasets can be advertised or used in studies.

Homomorphic Encryption (HE) — PET allowing computation on encrypted values; part of our PET envelope where appropriate.

Message-Driven Cloud↔DO Interaction — Async patterns (queues/websockets) to distribute/collect jobs and results reliably.

PETs (Privacy-Enhancing Technologies) — Envelope of techniques (DP, HE, SMPC) enforced at study design and per FL round via privacy hooks.

PETService (Privacy Hooks) — Service that validates planned aggregation + PET settings against a privacy budget before/throughout training.

PRICOM (Privacy Compliance Levels) — DO-side control to set/track dataset privacy posture; changes are audited alongside validations/approvals.

Questionnaire (FHIR) — The canonical survey/specification that defines variables for a dataset; each QuestionnaireResponse references one Questionnaire URL.

QuestionnaireResponse (FHIR) — Per-patient/record responses stored on-prem; today these form the core of all TRUMPET datasets.

RBAC (Role-Based Access Control) — Role-scoped authorisation applied across cloud dashboards/APIs and reflected in on-prem packages.

Researcher Dashboard — Cloud UI where researchers discover datasets, configure studies/FL jobs, monitor results—without raw data access.

Reverse Proxy (Traefik at DO Edge) — TLS termination and tight port exposure (“API hiding”) for DO services, with automated certs.

SDLC (Software Development Life Cycle) — Our quality baseline: coding standards, reviews, CI/CD and tested integration points across services.

Secure Multi-Party Computation (SMPC) — PET for combining updates/metrics without revealing inputs; part of the enforced PET envelope.

SSH Port-Forwarding Access Pattern (DO Ops) — Default admin access to on-prem services through local tunnels; inbound ports closed by design.



TRUstworthy Multi-site Privacy Enhancing Technologies

Study Agreement — The governance artifact defining which datasets, models and PET settings are authorised for a training run.

Token-Based Authentication — Short-lived tokens validated at the cloud edge and DO proxies; tokens can be rotated or revoked centrally.

TRUMPET Cloud — Public-facing layer for governance, researcher workflows, dataset discovery, and FL orchestration; stores metadata only.

Validation & Quality Gates (Edge) — DO-side checks for schema conformance, terminology, integrity, completeness and timeliness before publish/use.

1 Objectives

TRUMPET exists to break the deadlock between the need for large, multi-site datasets in AI research and the very real legal, ethical, and technical barriers to sharing sensitive data. Federated Learning (FL) helps, but on its own, it still leaks information through gradients and is vulnerable to “curious” aggregators and man-in-the-middle attacks. Our answer is ***Armoured Federated Learning (AFL)***—FL hardened with PETs, delivered as a usable platform for data owners and researchers.

The project builds the AFL platform and opens its reference implementation. It researches and integrates PET combinations tailored to FL. It creates a privacy metric and measurement tool that underpins GDPR-oriented certification. It validates everything in three hospital pilots (**Lung Cancer Research, Radiotherapy, and Head and Neck Cancer**).

The plan is iterative: requirements and PET/metric R&D feed a two-round build–pilot–refine cycle. Milestones pace this cadence—from initial requirements to the first platform prototype (M20), mid-course updates (M28), and the final validated release (M33–M39).

1.1 WP4 in TRUMPET

WP4—TRUMPET platform development, pilot, and validation—turns the science into a working product and proves it under real conditions. TVS leads WP4; hospitals (IRST, CHU) anchor data integration and pilots; GRAD leads performance validation and privacy measurement; AVO steers documentation and training; and UVIGO/INRIA contribute to platform integration and validation.

Objectives

1. Setup FHIR-compliant local repositories at hospital sites.
2. Design and build Data Owner and Researcher dashboards that make PET-hardened FL usable.
3. Execute three pilots and measure scalability, accuracy, usability, privacy, and security.
4. Produce user-facing documentation and training.

What we ship

- **D4.1 Data acquisition & integration (M18)**—strategy, tooling, and populated private repositories using HL7 FHIR/HAPI at sites.
- **D4.2a Initial platform report (M20)**—first complete cut of cloud services, dashboards, APIs, and deployment procedures; **D4.2b Final platform & dashboards (M36)**.

How the platform works— The Initial Platform (D4.2) defines end-to-end user journeys and wireframes—registration, dataset assessment, cohort definition, study creation, model development with data-owner validation, training, and result retrieval—plus governance/admin views. It also documents the low-level architecture (DDD), relational schema (PostgreSQL), REST APIs (with response standards and Swagger), and integration planning for the FL server.

Security & privacy assurance in WP4

- The dashboards and study flow apply PETs transparently and enforce data-owner agreements and privacy budgets.
- Validation includes external privacy pentesting by a subcontracted expert who will attempt ethical re-identification via the platform and issue a severity-graded report, which will feed back into fixes before the final release.

1.2 Deliverables & Milestones Map

How the pieces land over time. WP4 deliverables align with project-wide milestones to pace **build** → **pilot** → **refine**. Dates below are project months (Mx).

LEADER		NAME OF WP / TASK	From	To	1	19	20	21	22	23	29	Mar-25	31	32	Jun-25	34	35	36	Oct-25	38	Dec-25
TVS	WP4	TRUMPET platform development, pilot and validation	1	39	D4.5																
	T4.1	Data acquisition and integration	1	18																	
	T4.2	Design and develop TRUMPET platform and dashboards	6	36											D4.3				D4.3		
	T4.3	Piloting readiness and implementation	20	39																	D4.4
	T4.4	TRUMPET performance validation and privacy measurements	20	39																	D4.6

Figure 1: Deliverables & Milestones Map

Core WP4 deliverables

- **D4.1 – Data acquisition & integration (M18):** strategy, tooling, pipelines; private repositories populated and reachable by the platform.
- **D4.2a – Initial platform report (M20):** first complete cut—dashboards, APIs, deployment, user journeys.
- **D4.4a – Initial performance validation & privacy measurement (M23):** indicators after pilot round 1.
- **D4.2b – Final platform & dashboards (M39):** refined platform after PET/tool updates and requirement refresh.
- **D4.4b – Final performance validation & privacy measurement (M39):** indicators after pilot round 2; final numbers.
- **D4.3 – User manual (M39):** step-by-step guidance for all features.



TRUstworthy Multi-site Privacy Enhancing Technologies

Dependency notes— WP4 pulls inputs from WP2 (*PET specs/benchmarks*) and WP3 (privacy tool) and feeds pilot evidence into WP1 for the M28 requirements refresh. The cadence is intentional: ***build*** → ***pilot*** → ***measure*** → ***refine*** → ***pilot*** → ***finalise***.

2 Requirements Traceability

This section combines what WP1 gathered and refined across the project: what the platform must do (**Functional Requirements, FR**), how it must behave (**Non-Functional Requirements, NFR**), and which regulations and standards it must follow.

2.1 Stakeholders & Roles

TRUMPET is a two-sided platform. The primary users and stakeholders and their interests are:

- **Researchers / Solution Developers (demand side):** design studies, choose cohorts, develop/train models, and review results—always within agreed privacy budgets and without accessing raw data.
They work through the Researcher Dashboard (**TRUMPET Cloud, TC**) and a web-based study experience.
- **Data Owners (supply side, e.g., Data Owners):** keep data on-premise, publish dataset descriptors, approve/deny study execution, and run the local Learning Node. They operate the Data Owner Nodes (DON).
- **Healthcare Professionals (indirect users and stakeholders in pilots):** benefit from validated models and insights generated under strict privacy constraints.
- **Platform Operator / Governance Team:** designs and maintains the cloud servers, APIs, CI/CD, and integration to the FL Core.

2.2 Functional Requirements (FR)

FR-1. FHIR-based data (on-premise): Implement local repositories using HL7 FHIR as the canonical model (e.g., via HAPI FHIR), with adapters to hospital sources; no raw data leaves the site.

FR-2. Dataset publication: Allow Data Owners to publish discoverable dataset descriptors and PET choices.

FR-3. Researchers study the workspace: Provide a web interface to define cohorts, pick models, configure FL jobs, and monitor results—without direct data access.

FR-4. PET-armoured FL: Apply PETs (**HE, SMPC, DP**) to model updates and/or aggregation with combinations selected for the best privacy–utility–cost trade-off.

FR-5. Dashboards & APIs: Offer Cloud and Data Owner dashboards and documented REST APIs for roles, users, datasets, studies, and models; include standardised responses and Swagger documentation.

FR-6. Piloting & evidence: Run three staged pilots (lung cancer; radiotherapy, head and neck cancer) to validate usability, scalability, and privacy performance with real clinical partners.

2.3 Non-Functional Requirements (NFR)

NFR-1. Security & Privacy by design: End-to-end hardening, encrypted channels, role-based access, and “*curious backend*” threat model assumptions in FL Core.

NFR-2. GDPR compliance support: Features that facilitate lawful basis, minimisation, purpose limitation, fairness, transparency, integrity, confidentiality, and accountability; **DPIA** support for pilots.

NFR-3. Interoperability: Strict use of **HL7 FHIR** resources, indexes, and profiles for healthcare data.

NFR-4. Scalability & Performance: Cloud/node design supports multi-site FL training, with measured trade-offs for different PET combinations.

NFR-5. Reliability & Maintainability: Modular services, adapter pattern for the external FL Core server, dependency injection, and tested integration points.

NFR-6. Quality & SDLC: Coding standards, code review, and CI/CD.

NFR-7. Usability: Intuitive dashboards and guided flows for dataset publishing, study creation, and model training.

2.4 Compliance & Standards

- **GDPR:** The platform limits processing to on-premise computation, applies PETs to updates, exposes a privacy budget and a metric, and supports pilot DPIAs and informed consent processes via *WP5*.
- **HL7 FHIR:** FHIR is the canonical format and indexing basis for local repositories and adapters.
- **PETs (HE, SMPC, DP):** PET combinations are first-class design elements in TRUMPET to mitigate residual FL risks, validated against realistic attacker models beyond “**honest-but-curious aggregator.**”

3 System Architecture

TRUMPET is a privacy-armoured federated-AI platform that sits between researchers who want to run models and data owners who must never leak data. Researchers work in the cloud dashboards; data owners operate their local nodes; and the platform enforces privacy via PETs (CDC DP, ThHE). The value chain and governance—spanning the researcher dashboard and data-owner control panel — are core architecture drivers, as evidenced in the following sections.

At a high level, the cloud layer revolves around the Researcher, Data Owner, and Governance dashboards, supported by *RESTful APIs* and a *PostgreSQL-backed RDB*; model development and training are coordinated with the FL core aggregator module integrated with the cloud layer.

The node runs approved queries and training locally on the data-owner side and returns only privacy-preserving outputs to the cloud. The data owners upload and publish their datasets on their data-owner node. Once the data owner registers on the cloud, the installation steps of the data owner application and API integration guidance are available through the cloud dashboard.

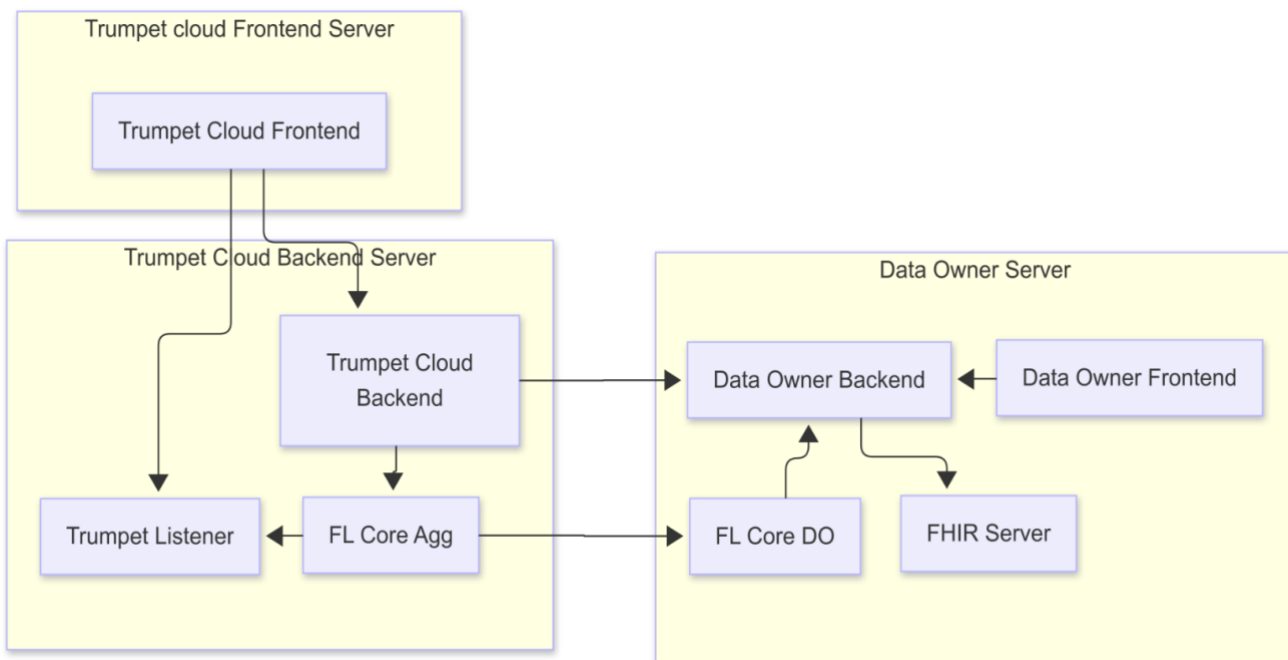


Figure 2: TRUMPET Platform Components

3.1 Cloud Architecture

The Cloud Node manages only non-sensitive metadata, user access, and coordination across sites. It has a clear split: the Frontend (Researcher, Data Owner, and Governance dashboards) talks over HTTPS to Backend REST APIs. The Backend includes a resource store, an FL Core Aggregator that coordinates training with Data Owner FL Cores, and a Webhook Listener that receives status, metrics, and errors from on-prem nodes and feeds real-time updates to the dashboards. With this setup, researchers can discover datasets, define cohorts, submit and track training jobs, and review aggregated results—without any raw hospital data ever leaving the sites.

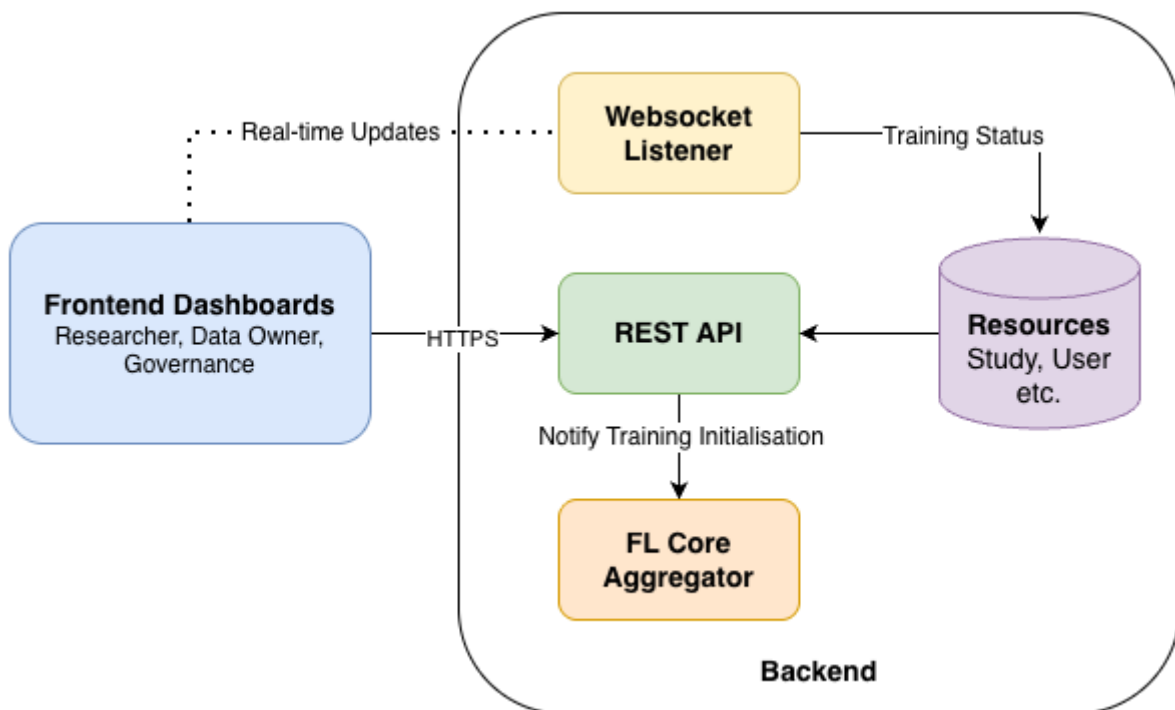


Figure 3: Cloud Architecture

The Cloud provides four dashboards—Governance, Researcher, Researcher Admin, and Data Owner—designed to keep roles clear and workflows simple. Each dashboard focuses on a distinct responsibility area while sharing a common sign-in, permissions model, and stable REST APIs. All actions operate on non-sensitive metadata; raw hospital data stays on-prem at the Data Owner Node.

Role	Access
Governance Admin	Full access to governance resources
Researcher Admin	Full access to researcher cloud dashboard resources

Data Owner Admin	Full access to data owner cloud dashboard resources
Researcher	Full control over study module

Table 1: Cloud Dashboard User Roles

The Governance Admin dashboard is where organisations and admins are onboarded. Governance admins review submitted documents, verify identity, and approve or decline Researcher Admins and Data Owner Admins. They can assign or revoke roles, enforce access policies, and keep an audit trail for compliance.

Researcher Admins use their dashboard to onboard individual researchers, and manage teams. Researchers then use their dashboard to browse available datasets, explore metadata and cohort summaries, create and submit studies, monitor training progress via real-time updates, and download aggregated results or reports—without touching raw clinical data.

Data Owner Admins review study requests for their site’s datasets and decide what to approve or decline. From the same dashboard, they manage site settings, create secure keys for Cloud-Node communication, and access built-in setup guides. Clear, step-by-step instructions cover installing the Data Owner Node, connecting to the local HAPI FHIR server, uploading data, and handling routine operations.

Role based Dashboard	Key functionalities
Governance Admin	Onboard orgs/admins, verify documents, approve/decline Researcher & Data Owner Admins, manage roles/policies.
Researcher Admin	Onboard researchers, manage teams.
Researcher	Browse datasets, view metadata/stats, create & submit studies, monitor training, download aggregated results.
Data Owner Admin	Review/approve study requests, manage site settings, generate keys for Cloud-Node communication, follow setup guides (install Node, connect HAPI FHIR, upload data).

Table 2: Cloud Dashboards - Roles and Key Functionalities

The cloud platform runs in a secure Azure environment with role-based access. We have a CI/CD pipeline that lets us ship updates to the dashboards, APIs, and the FL cloud aggregator quickly and safely, with minimal downtime.

3.1.1 Cloud Workflows at a Glance

In the Cloud dashboards, each role follows its own path: researchers design studies and track results, Data Owner Admins review requests for their site, and Governance Admins handle approvals and policy. The core journey is simple—create a study, agree on scope, get approvals, and run training across participating sites. Real-time status comes back to the study workspace while raw data stays on-prem. Around that, the basics are covered too: register and sign in, reset passwords, and generate keys for secure Cloud to Node communication.

Onboarding & Sign-In:

Admins and organisations start by registering and verifying their email, then completing a short profile with the details needed for approval. Once governance approves the account, users sign in through the standard login screen, and a self-serve password reset is available. The experience is simple on purpose: one place to register, one place to log in, and a clear status on where you are in the approval process.

Researcher Study Lifecycle:

Researchers explore published datasets, review the available metadata/stats, and then create a study from a clean workspace. Within the study, they choose participating datasets, select PET models, and set use-case parameters under the study agreement. The workspace keeps everything together—description, agreements, notifications, run status, and results—so researchers can submit jobs, track real-time progress, and download aggregated outputs without leaving the page.

Model Development with Data Owner Validation:

Before training begins, Data Owner Admins review the proposed model in their dashboard. Once they approve, the researcher triggers the training request. The job runs across participating Data Owner Nodes, with data always remaining on the data owner nodes. The Cloud collects status and metrics through the webhook listener and posts updates and final results to the study area for review and download.

Smaller, routine flows also support the day-to-day: forgotten-password recovery, creating and revising study agreements, managing API keys, and accessing setup guides and other resources. These are kept lightweight and consistent with the main experience, so users don't need to relearn the interface for simple tasks. The end-to-end flows are described in detail in the following section, with UI screenshots to illustrate each step.

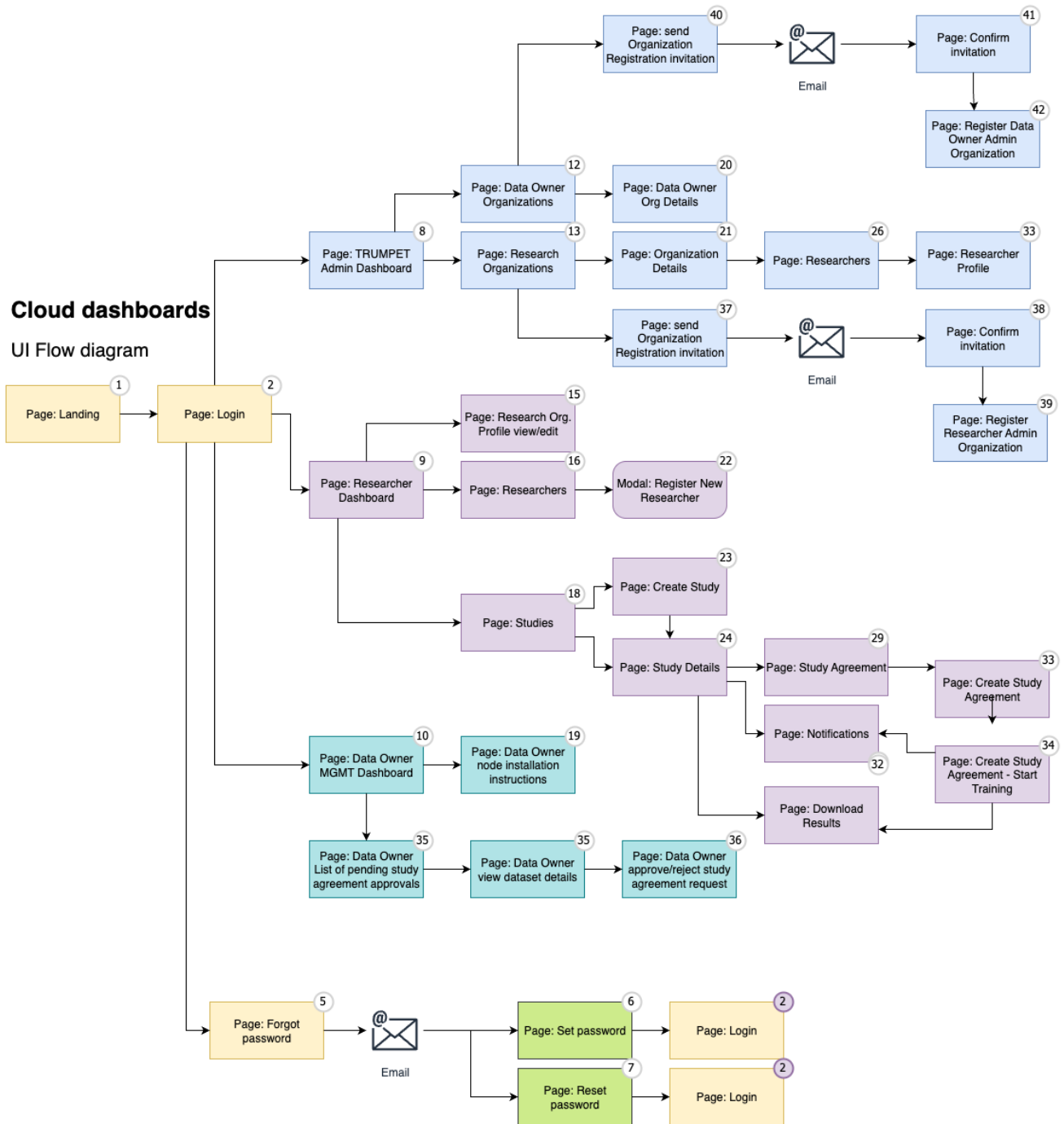


Figure 4: Cloud workflows at a glance

3.1.2 Low-Level Design (Modules, APIs, Data Models)

Services publish resource-first REST APIs on PostgreSQL (via SQLAlchemy), with Alembic handling safe, incremental migrations; the codebase applies Domain-Driven Design and Clean Architecture to keep the domain core cleanly separated from application and infrastructure layers. Core services and entities focus on Organization, User, Study, Dataset, and Agreement. Domain logic lives in plain individual logic files; application

services orchestrate use cases; adapters handle persistence, transport, and external integrations. Boundaries are explicit, and dependencies point inward.

3.1.2.1 Layers and dependency management

We design the cloud so that changes in outer layers (such as third-party libraries) do not force changes in the domain or application layers. Every external dependency is handled through well-defined interfaces in the appropriate layer. Third-party libraries are adapted to our interfaces first, and only then used by the application. This keeps the core logic stable, testable, and easy to maintain—even when vendors or tools change.

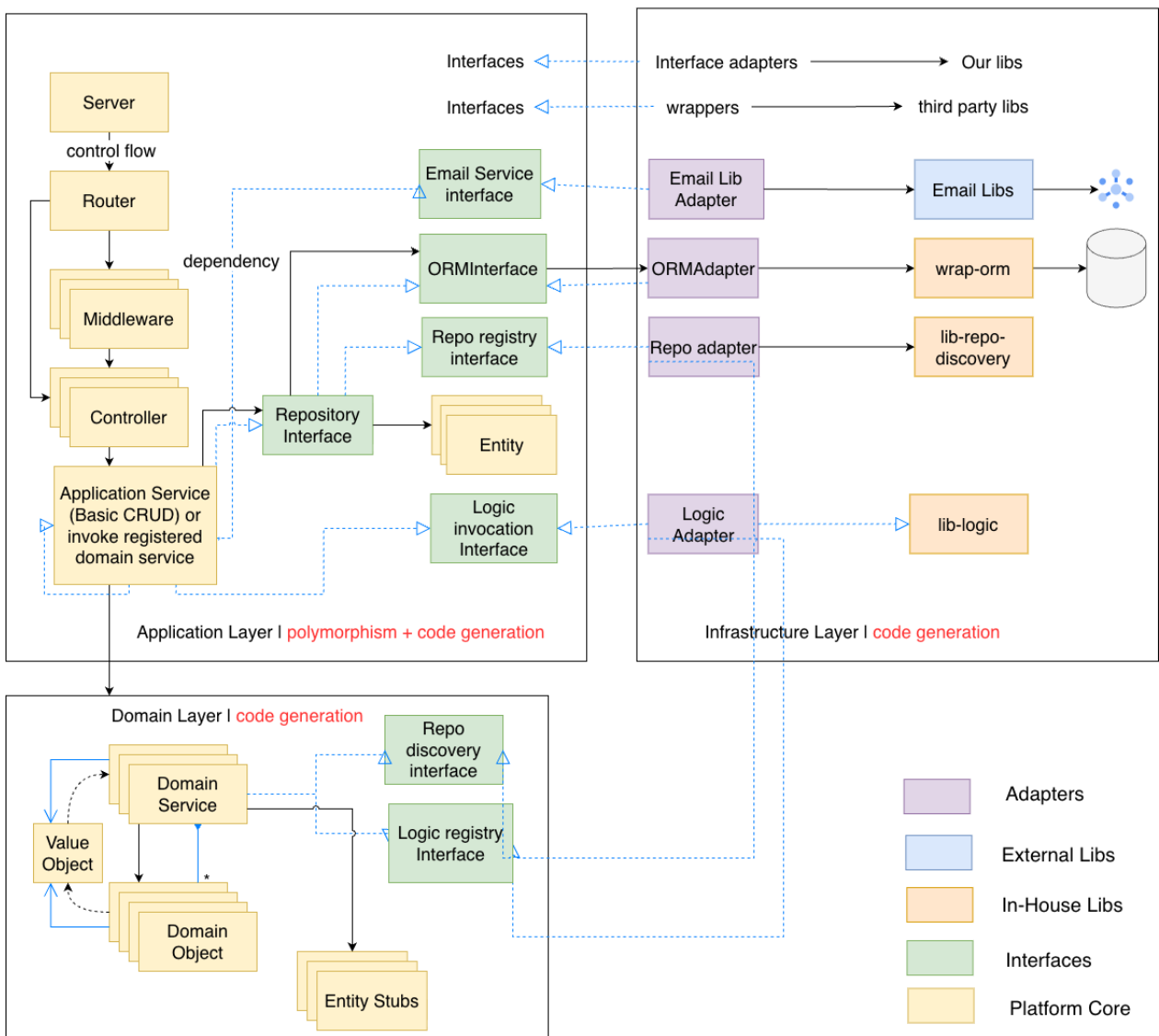


Figure 5: Low Level Dependency Diagram

3.1.2.2 DDD: Aligning Software with the Business

We chose Domain-Driven Design because it tightly aligns the software with our business. By modelling the system around real processes and rules, the codebase reflects how the organisation works, making features easier to reason about and keeping us focused on outcomes rather than frameworks or plumbing.

DDD also improves teamwork. Using a shared, ubiquitous language between developers and domain experts minimises ambiguity and reduces rework. Concepts carry the same meaning in conversations, documentation, and code, so decisions travel cleanly from workshops to implementation.

From an architectural perspective, bounded contexts and aggregates make a modular structure easier to maintain and evolve. Clear boundaries limit ripple effects when requirements change and help us isolate complexity where it belongs. This naturally supports a focus on the core domain: we invest most of our effort where the project creates the most value while keeping supporting areas simpler.

Clear boundaries that make the system easy to maintain also make it easier to scale. Teams can build in parallel within their own contexts, we can decouple deployments where it makes sense, and the platform can grow without sliding into a tangled monolith. In short, DDD keeps us aligned with the business, clear in our language, safe in our changes, and sensible in how we scale.

3.1.2.3 Core Services

The platform's core services start with Identity & Access, which manages accounts, roles, and permissions across all dashboards—Governance, Researcher, Researcher Admin, and Data Owner. Registration, invitations, sign-in, and password recovery are handled through a single flow. At the same time, role-scoped endpoints enforce RBAC at the API layer, so each user only sees the resources they're entitled to use.

Organisation & Governance maintains a registry of participating organisations—research groups and data-owning sites—and provides the administrative views needed for onboarding and compliance. The Data Owner Directory also captures site details for each Data Owner, including node metadata and basic operational statistics. It offers step-by-step guidance to help hospital IT teams set up and maintain their on-prem Data Owner Node.

On the data side, Dataset Management publishes discoverable datasets to researchers using non-sensitive metadata, descriptive statistics, and owner information. Researchers use Study Management to define studies and agreements, add collaborators, choose PET

models and use-case parameters, link one or more datasets, and track training runs and outputs from a single workspace. Researchers can start training, monitor progress, and download aggregated results from here, while Data Owner Admins review and approve study requests tied to their site’s datasets.

For federated execution, the FL Platform Adapter exposes a clean boundary to the external FL server, keeping our services decoupled while enabling privacy-preserving orchestration across sites. Within the Cloud, the FL Core Aggregator coordinates training requests to participating Data Owner FL Cores, and the Webhook Listener receives status, metrics, and error events from those nodes. The listener feeds the backend APIs and metadata store, so dashboards show real-time progress without handling raw hospital data.

Service	Responsibility	Key flows
Identity & Access	Accounts, roles, permissions across all dashboards	Single flow for register/invite/login/reset; RBAC enforced at API layer
Organisation & Governance	Onboard researchers, manage teams.	Onboard/verify orgs; compliance views
Data Owner	Browse datasets, manage study approvals, follow setup guides (install Node, connect HAPI FHIR, upload data).	Setup guidance for on-prem Node; operational tips for hospital IT
Study Management	Review/approve study requests, manage site settings, generate keys for Cloud to Node communication.	Define studies/agreements, add collaborators, select PET models, link datasets, track runs, download aggregated results
FL Core Aggregator	Coordinates training across Data Owner FL Cores	Sends training jobs; aggregates run status/results metadata
Cloud Listener	Ingests status/metrics/errors from sites	Feeds back-end APIs/metadata store; real-time dashboard updates; no raw clinical data

Table 3: TRUMPET Cloud Core Services

Under the hood, services are built with Python 3.11 on FastAPI, using SQLAlchemy and Alembic for persistence and migrations, PyTest for testing, Pydantic-Settings and Dependency-Injector for configuration and wiring, and Poetry for packaging. Everything

runs locally and is containerised for deployment on Docker, keeping developer workflows smooth and operations predictable.

3.1.2.4 Core Entities and Relations

In the Cloud, access follows a straightforward RBAC model: Users are linked to Roles, and permissions are enforced at both the API and resource levels. Users—researchers, data owners, and admins—belong to one Organisation, which anchor identity, governance, and approvals.

At the centre is the Study. Each Study owns its contractual state through a Study Agreement model: a Study has many Study Agreements, and each Study Agreement belongs to a single Study. Agreements reference many Datasets, and the same dataset can appear in multiple contracts, giving us a many-to-many relationship between agreements and datasets via explicit link tables. This structure lets us track approvals and scope precisely per Study, while reusing datasets across different collaborations.

The platform also manages Files—kept separately as organisation files and researcher files—and Notifications, stored per user to surface approvals, training updates, and results. The relational model names Users, Organisations, Studies, Study Agreements, Datasets, and Roles as first-class tables, with join tables for agreement↔dataset and (where needed) Study ↔dataset mappings. Constraints keep relationships consistent and operations predictable. We use PostgreSQL with SQLAlchemy and Alembic to define models and run migrations for implementation.

Entity	Description
User	Researchers, Data Owner Admins, Governance/Admin users.
Role	RBAC roles applied to users.
Organization	Research groups or data-owner organizations.
Study	Top-level research unit/workspace.
Study Agreement	Contractual/approval state per study; scoped to datasets and orgs.
Dataset	Discoverable dataset (metadata and stats only).

File	Stored documents; either Organisation File or Researcher (User) File.
Notification	Per-user alerts (approvals, training updates, results).

Table 4: Cloud Entities

Relation	Cardinality	Details
User ↔ Role	M:N	RBAC enforced at API/resource level (user_role link).
User ↔ Organisation	M:N	Users can belong to multiple orgs (user_org link).
Organisation ↔ Study	1:M	An org can own/host many studies.
Study → Study Agreement	1:M	A study has many agreements; each agreement belongs to one study.
Study Agreement ↔ Dataset	M:N	Agreements reference one or more datasets; datasets can appear in many agreements (agreement_dataset link).
File → Organisation	M:1	Organisation files.
File → User	M:1	Researcher/user files.
Notification → User	M:1	Each notification targets a single user.

Table 5: Cloud Entity Relations

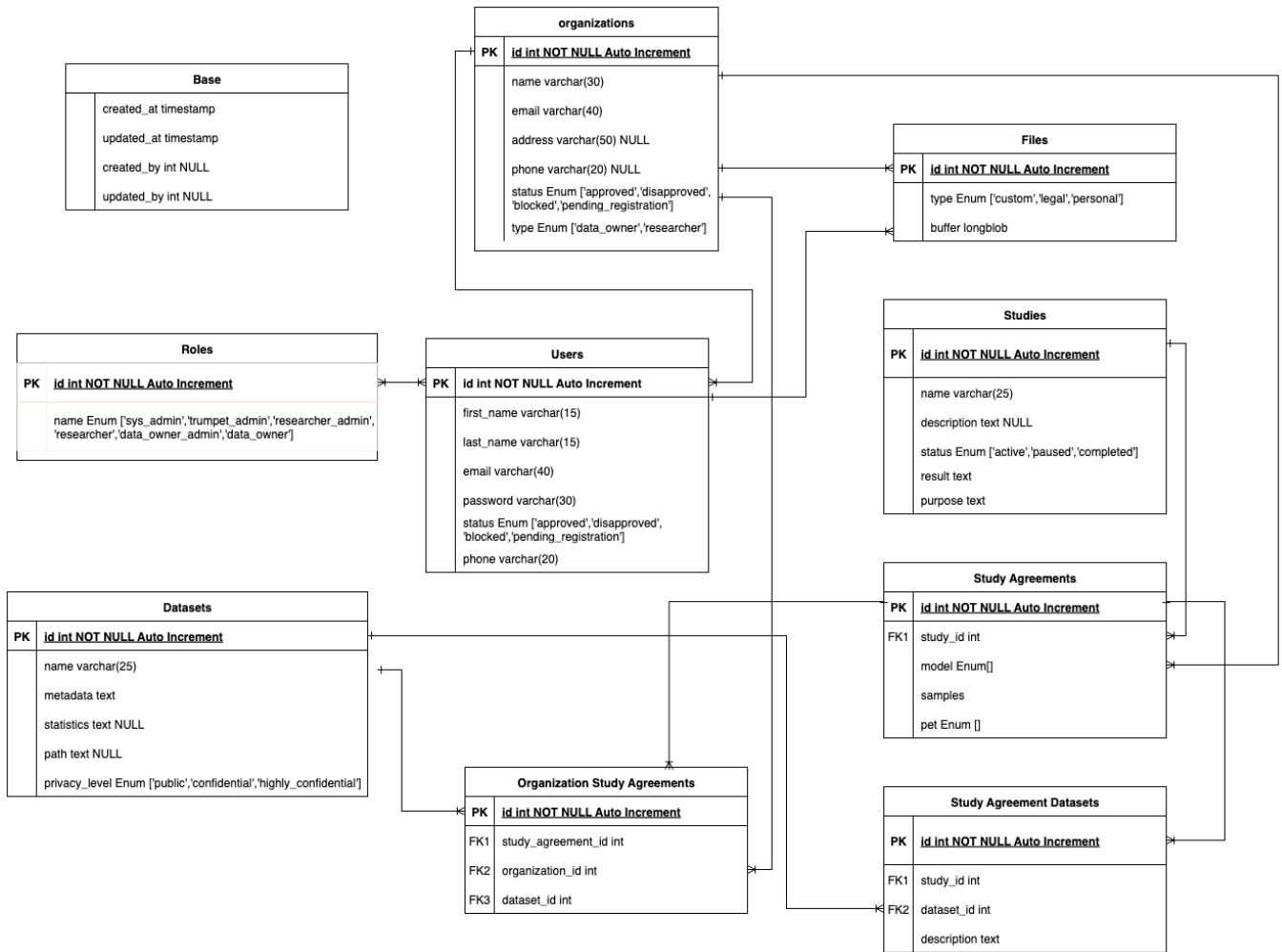


Figure 6: Relational Database Schema Diagram of Cloud

3.1.2.5 API endpoints

The Cloud API is the platform’s backbone. It drives the Cloud frontend, supports the internal webhook listener, and—where appropriate—serves Data Owner Nodes through secured, role-scoped access. Every request is authenticated and audited; raw clinical data never crosses these endpoints.

Resource-based endpoints follow REST, so integrations stay predictable. Paths are versioned and stable, HTTP verbs and status codes are consistent, and errors share a familiar shape. Pagination, filtering/sorting, and ETags keep access efficient and concurrency safe.

Resource	Method	Endpoint
Organizations	POST	/organizations
	GET	/organizations

	GET	/organizations/:id
	PATCH	/organizations/:id
Users	POST	/users
	GET	/users
	GET	/users/:id
	PATCH	/users/:id
	DELETE	/users/:id
OrganizationUsers	GET	/organizations/:id/users
Studies	POST	/studies
	GET	/studies
	GET	/studies/:id
	PATCH	/studies
Datasets	POST	/datasets/:id
	GET	/datasets
	GET	/datasets/:id
StudyAgreements	POST	/studies/:study_id/agreements
	GET	/studies/:study_id/agreements
	GET	/studies/:study_id/agreements/:id
StudyAgreementResults	POST	/study_agreements/:study_agreement_id/results
	GET	/study_agreements/:study_agreement_id/results
	GET	/study_agreements/:study_agreement_id/results/:id
	PATCH	/study_agreements/:study_agreement_id/results/:id
Notifications	GET	/notifications
	PATCH	/notifications/:id

Table 6: Cloud Resource Based Endpoints

Alongside these are action-style, non-resource endpoints. They handle file transfer via pre-signed URLs, token/key generation and rotation, and operational exchanges with Data Owner Nodes and the webhook listener (for example, job start/stop and status ingest). These routes are tightly scoped, require elevated authentication, and use signed requests to protect cross-site communication.

Operation	Method	Endpoint
Login (issue auth tokens)	POST	/login
Send org invitation	POST	/send-organization-invitation
Confirm org email	POST	/confirm-organization-email

Send researcher invitation	POST	/send-researcher-invitation
Confirm researcher email	POST	/confirm-researcher-email
Add new researcher	POST	/add-new-researcher
File upload	POST	/file-upload
File download	POST	/file-download
Send password reset token	POST	/send-password-reset-token
Reset password	POST	/password-reset
Generate API token/key for Data Owner Node	POST	/generate-token
Unpublish dataset	POST	/unpublish-dataset
List study agreements of logged in organization	GET	/get-study-agreements
Approve/decline agreement	POST	/study-agreement-approval
Notification count	GET	/api/notification/count
Start training	POST	/start-training

Table 7: Cloud Non-resource Endpoints

Responses follow the Google JSON Style Guide; standard codes include 200, 201, 400, 401, 404, 422, 500. This uniformity simplifies client code and cross-service integration.

200	Success
201	Created
401	Unauthorised
400	Bad Request
404	Not Found
422	Validation error
500	Server Error

Table 8: Followed REST API response codes and their meanings

Interactive API docs are auto-published at /docs (Swagger) and /redoc (ReDoc) in all environments. For security, these UIs are disabled on the cloud deployment. Instead, we provide a Postman collection with full endpoint coverage for authenticated access.

<https://documenter.getpostman.com/view/26191652/2s9YXh52Zp>

3.1.3 Folder Structure and Responsibilities

The project follows a layered, DDD-inspired structure: application_layer holds use-case entities and interfaces; domain_layer (reserved for core domain logic) isolates business rules; and adapters provide the “edges”—FastAPI controllers, middlewares, response adapters, and the ORM wrapper (wrap_orm_adapters/models) for SQLAlchemy. Shared

base classes live in `lib_archi` (controllers, repositories, entities), with reusable infra helpers in `adapters/lib_archi`. Dependency wiring sits in `logic_injector`, while `main.py` and `app_layer_entrypoint.py` are clean API and app layer entry points. Operational files—`Dockerfile`, `docker-compose` (x86/ARM), `config.json`, and `generate_tables.py`—bundle configuration and setup. The result is a clear separation of concerns: web/API I/O at the edges, orchestration in the application layer, domain rules in their own space, and swappable persistence behind repositories—keeping code testable, replaceable, and easy to evolve.

```

├── adapters
│   ├── entity_adapters
│   │   ├── entity_validation.py
│   │   └── __init__.py
│   ├── __init__.py
│   ├── lib_archirs
│   │   ├── fastapi_controller.py
│   │   ├── __init__.py
│   │   ├── inmemory_repository.py
│   │   └── orm_repository.py
│   ├── middlewares
│   │   ├── cors.py
│   │   ├── __init__.py
│   │   ├── response_middleware.py
│   │   └── validation_middleware.py
│   ├── response_adapters
│   │   ├── __init__.py
│   │   └── response_handler.py
│   └── wrap_orm_adapters
│       ├── __init__.py
│       ├── models
│       │   ├── base.py
│       │   ├── datasets.py
│       │   ├── files.py
│       │   ├── __init__.py
│       │   ├── organizations.py
│       │   ├── organization_users.py
│       │   ├── roles.py
│       │   ├── studies.py
│       │   ├── study_agreement_datasets.py
│       │   ├── study_agreement_queries.py
│       │   ├── study_agreement_results.py
│       │   ├── study_agreements.py
│       │   ├── study_users.py
│       │   ├── user_roles.py
│       └── users.py

```

```

|       └─ orm_adapter.py
├─ app_layer_entrypoint.py
├─ application_layer
|   └─ abstractions
|       └─ controller_interface.py
|       └─ entity_interface.py
|       └─ __init__.py
|       └─ orm_interface.py
|       └─ response_interface.py
├─ entities
|   └─ dataset.py
|   └─ file.py
|   └─ __init__.py
|   └─ organization.py
|   └─ organization_user.py
|   └─ role.py
|   └─ study_agreement_dataset.py
|   └─ study_agreement.py
|   └─ study_agreement_query.py
|   └─ study_agreement_result.py
|   └─ study.py
|   └─ study_user.py
|   └─ user.py
|       └─ user_role.py
├─ __init__.py
├─ config.json
├─ docker-compose.yml
├─ docker-compose.yml.arm
├─ Dockerfile
├─ domain_layer
├─ entrypoint.sh
├─ generate_tables.py
├─ __init__.py
├─ lib_archi
|   └─ base_application_service.py
|   └─ base_controller.py
|   └─ base_entity.py
|   └─ base_repository.py
|       └─ __init__.py
├─ lib_logic
├─ lib_repo_discovery
├─ logic_injector
|   └─ base_logic_injector.py
|       └─ __init__.py

```

```
├─ main.py
├─ poetry.lock
├─ pyproject.toml
└─ README.md
```

code depend on abstractions rather than concrete details. The `domain_layer` is reserved for core business rules and remains free of framework concerns, making it safe to evolve logic without touching I/O. Dependency wiring is handled in `logic_injector`, which assembles implementations behind those interfaces for a given runtime.

At the edges, adapters provide everything needed to talk to the outside world. HTTP concerns are implemented via `lib_archirs/fastapi_controller.py` and middlewares for CORS, validation, and unified responses; outbound shapes are normalised in `response_adapters`. Persistence is isolated behind `wrap_orm_adapters`, which offers a thin repository layer over SQLAlchemy so schema changes don't leak into the application layer. The `models` directory under this adapter maps first-class entities—users, roles, organisations, studies, study agreements and their joins, datasets, files, and notifications—to relational tables. This boundary keeps controllers simple and makes test doubles easy to swap.

Shared building blocks sit in `lib_archi` (base controller, entity, repository, and application service types) and `adapters/lib_archis` (framework glue), while `lib_logic` and `lib_repo_discovery` hold reusable logic and repository discovery helpers used across modules. Entrypoints are minimal by design: `main.py` starts the HTTP surface; `app_layer_entrypoint.py` exposes application services for internal use. Operational pieces—`Dockerfile`, `docker-compose (x86/ARM)`, `entrypoint.sh`, `config.json`, and `generate_tables.py`—bundle configuration, local bootstrap, and schema migration scaffolding. Dependency versions are fixed with the help of Poetry (`pyproject.toml`, `poetry.lock`) to ensure reproducible builds. Setup instructions are in `README.md`. The structure isolates web I/O, persistence, and domain logic, sustaining fast test execution and minimising refactor risk.

Layer / Folder	Responsibility	Key contents / examples
application_layer	Use-case orchestration on clean interfaces; app-facing entities and ports.	entities/, abstractions/ (controller/ORM/response interfaces).
domain_layer	Core business rules, independent of frameworks.	Domain logic (reserved space for pure domain code).

adapters	Edges for I/O: HTTP, middleware, responses, ORM wrapper.	responses, ORM wrapper. lib_archirs/fastapi_controller.py, middlewares/ (CORS, validation), response_adapters/, wrap_orm_adapters/ + models/ (SQLAlchemy).
lib_archi	Shared base types and patterns used across layers.	base_controller.py, base_repository.py, base_entity.py, base_application_service.py.
lib_logic	Reusable logic helpers shared by modules.	Helpers to manage business logics.
lib_repo_discovery	Repository discovery/registration utilities.	Helpers to locate/bind repo implementations.
logic_injector	Dependency wiring for runtime composition.	base_logic_injector.py (binds interfaces → implementations).
adapters/entity_adapters	Validation/adaptation at entity boundaries.	entity_validation.py.
main.py	HTTP endpoint (FastAPI surface).	App startup.
app_layer_entrypoint.py	Programmatic entry to application services.	Internal service access without HTTP.
wrap_orm_adapters/models	Relational mappings for core entities.	users.py, roles.py, organizations.py, studies.py, study_agreements.py, datasets.py, files.py, joins.
Dockerfile / docker-compose*.yml / entrypoint.sh	Containerization and local runtime.	Build/run scripts (x86 & ARM).
pyproject.toml / poetry.lock	Dependency and build management.	Reproducible environments.
config.json / README.md	Configuration and project guidance.	Default settings, setup docs.

Table 9: Cloud Folder Responsibilities at a Glance

3.2 Data-Owner Node Architecture

The Data Owner Node is a self-contained, on-prem setup built for privacy by design. A Django REST-based backend powers the dashboard APIs, while a Next.js frontend gives admins a clear view of studies, approvals, and runtime status. PostgreSQL handles

application data and audit trails. Clinical data is stored as HL7 FHIR resources in a local HAPI FHIR server, which sits behind a HAPI proxy to enforce authentication and limit exposure.

Model training happens locally through the FL Core. It pulls prepared inputs via the backend APIs, trains on site data, and exchanges only model parameters with external FL cores to stay in sync. No raw data is ever shared; only model updates move across the wire. The result is a node that participates fully in federated workflows while keeping patient data inside hospital boundaries.

A Traefik gateway sits before the node to lock down network exposure. Most services bind to private ports; only a small set of endpoints is published, and those are restricted to specific source IPs through Traefik rules. Internal-only endpoints remain unreachable from outside; even local use typically requires explicit port forwarding before accessing an admin workstation. This setup narrows the attack surface while keeping routine operations predictable and controlled.

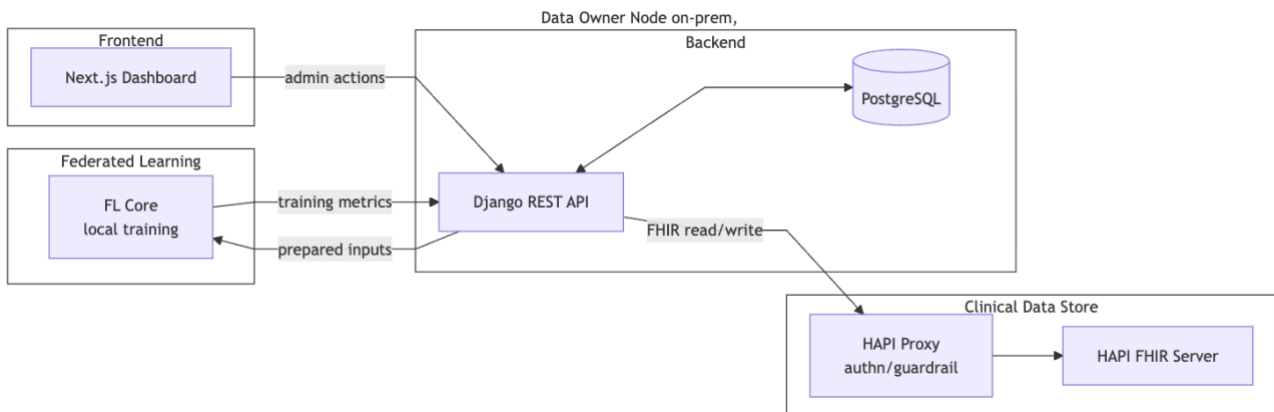


Figure 7: Data Owner Node Architecture

Component	Technology	Purpose	Data handled
Frontend Dashboard	Next.js	Admin views for studies, approvals, runtime status	Non-sensitive UI state/metadata
Backend API	Django REST Framework	Orchestrates node operations, prepares training inputs	Site metadata, job configs (no raw clinical data)
Application DB	PostgreSQL	Stores app data, audit logs, configs	Non-clinical operational data

FHIR Store	HAPI FHIR (behind HAPI Proxy)	Persists HL7 FHIR resources securely	Clinical data (on-site only)
HAPI Proxy	Auth gateway	Authenticates/guards FHIR access, limits exposure	Access control, no data persistence
FL Core	Local FL engine	Trains on local data; exchanges model parameters only	Model weights/updates (no raw data)
Traefik Gateway	Reverse proxy	Restricts ports/IPs; exposes only approved endpoints	Network traffic control

Table 10: Data Owner Node — Components at a Glance

The Data Owner Node dashboard’s primary job is managing site datasets—creating them, computing basic statistics, and publishing or unpublishing them to the Cloud catalogue. Publication changes are pushed to the Cloud securely, so visibility updates propagate without exposing raw data. The node also issues and manages access tokens used for inbound and outbound communication with the remote Cloud.

After registration in the Cloud dashboard, administrators receive guided deployment instructions tailored to the site. Setup is intentionally lightweight: run a short sequence of commands, add a few settings, and bring the node online. From there, the dashboard walks through everyday operations such as dataset updates, stats generation, publishing, and token maintenance.

Area	Key functionality
Dataset management	Create/update datasets, generate or refresh statistics, validate metadata.
Publication controls	Publish/unpublish datasets; sync status to the Cloud catalogue securely.
Connectivity & tokens	Create/rotate access tokens for inbound/outbound Cloud communication.

Table 11: Data Owner Node — Dashboard Areas & Key Functionality

3.2.1 Data-Owner Node Workflows at a Glance

The flow runs from a lightweight setup to day-to-day dataset publishing and privacy-preserving training in the on-prem Data Owner Node. The dashboard and API work together with the local FHIR store and FL Core so studies can run where the data lives. Only metadata and model updates move outward; raw clinical data stays on site.

Node setup & onboarding:

First, the organisation/admin registers in the Cloud and retrieves the installation guide from the dashboard. The node components are then deployed on-prem and connected to the local repository (HAPI FHIR and PostgreSQL). Once healthy, the site will be available in the cloud and ready to receive study requests.

Dataset ingestion & publishing:

Datasets are uploaded into the local FHIR-compliant store. The node prepares dataset metadata and statistics from what was ingested, then publishes the descriptor to the Cloud catalogue so researchers can discover it. Visibility is metadata-only; underlying clinical records never leave the premises.

Federated training execution:

After approvals are in place, the researcher's training request reaches the site, and the FL Core starts a local run. Only model updates and metrics are exchanged with the federated process as training proceeds in rounds until convergence. Integration keeps the local runtime interoperable with the Cloud coordinator while shielding the dataset.

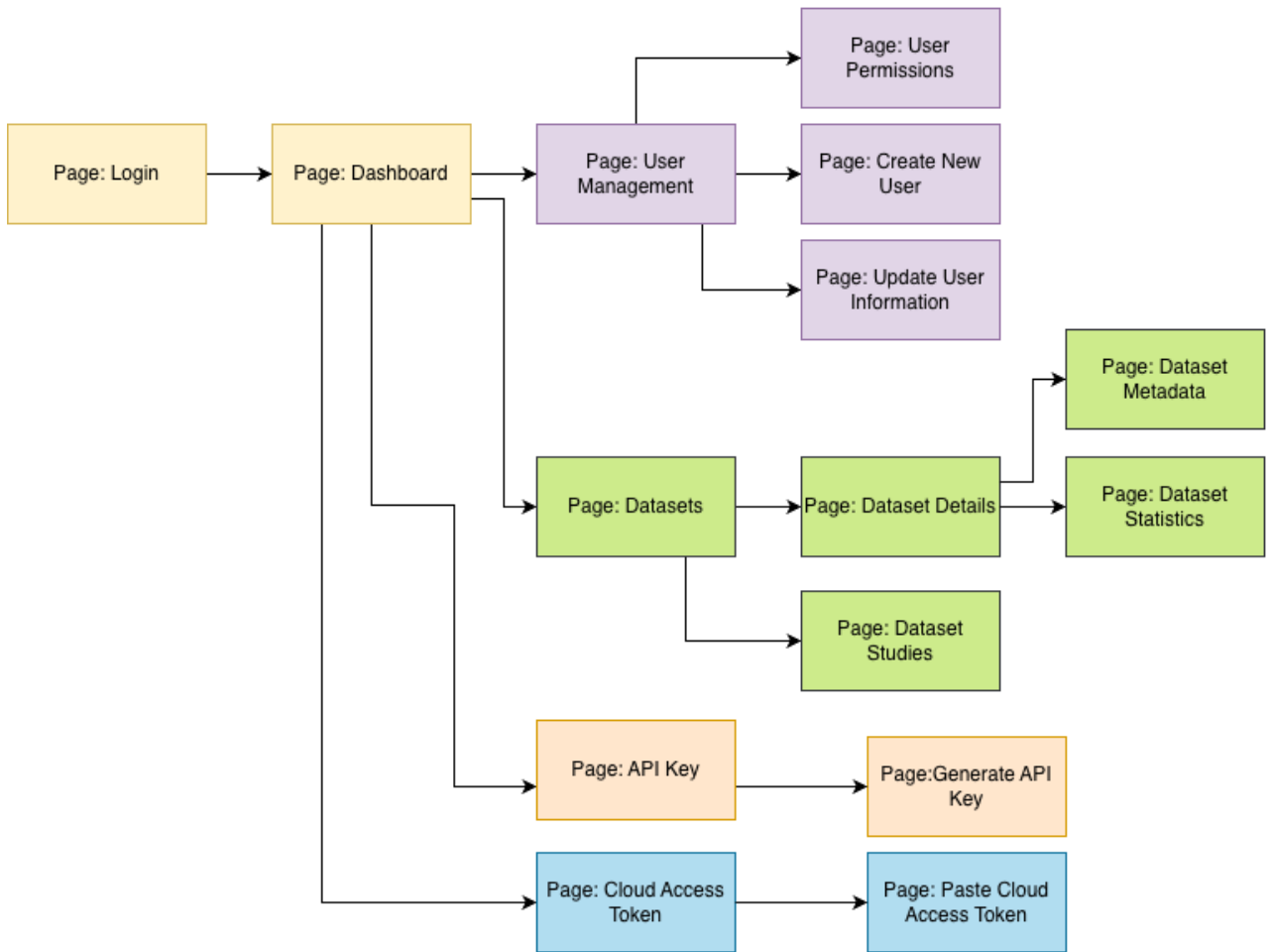


Figure 8: Data Owner Node UI flows

3.2.2 Low-Level Design (Modules, APIs, Data Models)

The backend follows a service–repository pattern to keep business logic separate from persistence. Services coordinate use cases, while repositories encapsulate data access. This split simplifies testing, makes storage choices replaceable, and keeps privacy rules close to the workflows they govern.

Architecture leans on Clean Architecture and Hexagonal (Ports & Adapters) principles. Business rules and use cases sit at the centre, while frameworks and I/O stay at the edges. Third-party systems—message brokers, storage, email, caching—are accessed through adapters, reducing vendor lock-in and allowing swaps or compositions without rewrites. Dependency injection wires ports to adapters at runtime, keeping boundaries explicit and testing lightweight.

Across the Cloud and Node, Domain-Driven Design defines clear bounded contexts so federated learning scenarios remain predictable. Aggregates and context boundaries limit ripple effects, while a shared language across modules keeps model-training flows and

privacy constraints consistent. The result is a modular, testable codebase, and resilient to change.

3.2.2.1 Core Entities and Relations

The core entities in the Data Owner Node are lean and purpose-built. Users manage local operations and own datasets; a single user can create and maintain many Datasets. Each dataset entry acts like a catalogue card, pointing to its records in the local HAPI FHIR store and tracking metadata, counts, and sync status—without duplicating clinical content.

Access and connectivity rely on keys rather than roles: APIKey governs internal calls to the dashboard API and HAPI proxy. CloudAccessToken is the site’s credential for publishing/unpublishing datasets and exchanging run-time signals with the Cloud. This split makes ownership obvious (User → Datasets), anchors each dataset to its HAPI FHIR source of truth, and cleanly separates local access from cross-site communication—leaving a clear, auditable boundary.

Item	Details	Purpose	Scope
Dataset entry	Catalog record pointing to local HAPI FHIR data	Tracks metadata, counts, sync status (no data duplication)	Local node
APIKey	Internal credential	Controlled access to dashboard API and HAPI proxy	Inside the node
CloudAccessToken	Site credential for Cloud	Publish/unpublish datasets; send runtime signals to Cloud	Cross-site (Cloud ↔ Node)
Ownership	User → Datasets	Keeps who owns/maintains datasets unambiguous	Governance within node

Table 12: Data Owner Node — Dataset & Access Model (At a Glance)

The **Dataset table** links each dataset to its canonical **Questionnaire** (by URL). Using this link, the system knows which sets of FHIR records belong to each dataset.

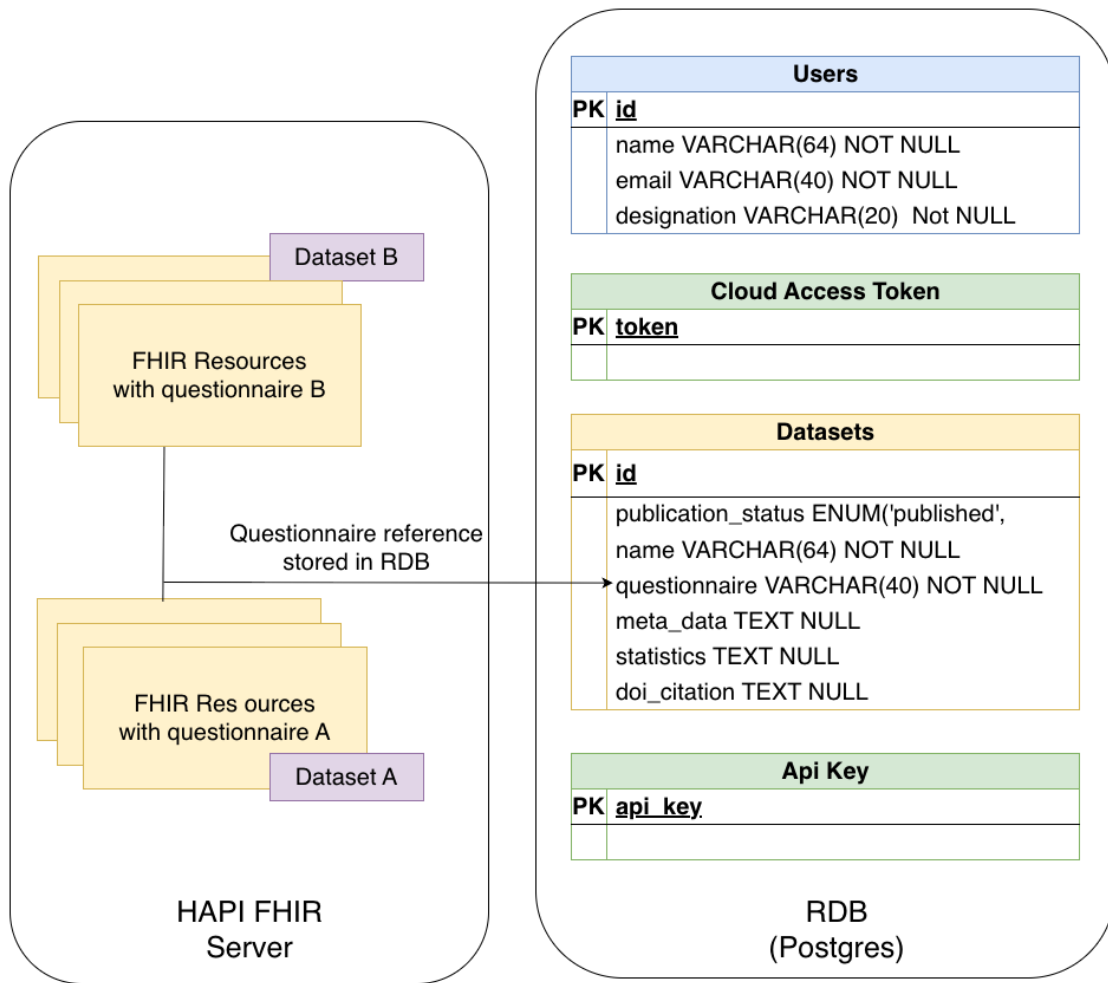


Figure 9: Data Owner Node Storage System with Relations

HAPI FHIR Server stores HL7 FHIR resources (e.g., Questionnaire/QuestionnaireResponse) used for analytics and federated training. Raw clinical data stays here and never leaves the hospital.

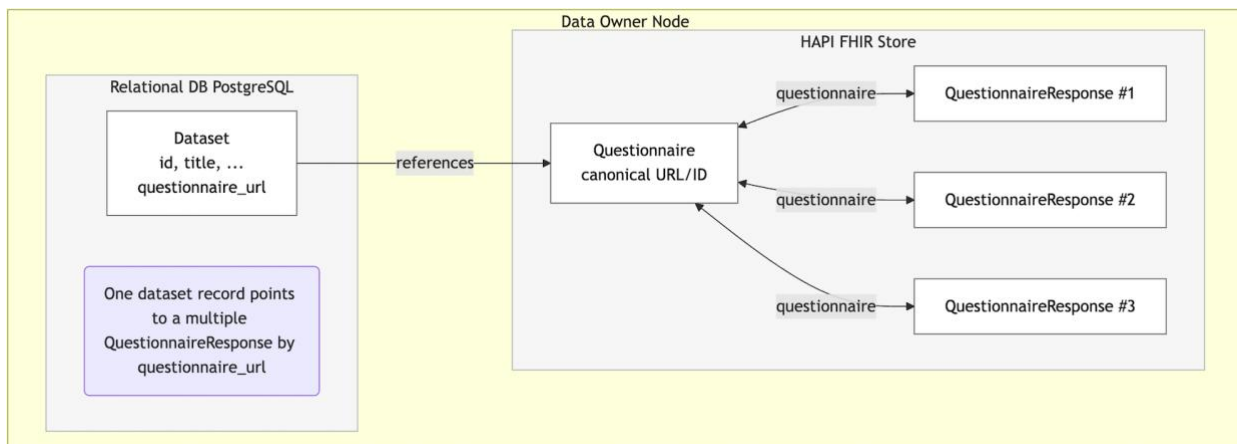


Figure 10: Data owner Node Dataset relation to QuestionnaireResponse Records



TRUStworthy Multi-site Privacy Enhancing Technologies

Custom QuestionnaireResponse template

We defined a custom QuestionnaireResponse template with our hospital partners that fits our data flows and study needs. This template standardises how fields are captured, minimises ambiguity, and makes downstream analytics and federated training consistent.

Below is a custom template used in the platform.

```
{
  "resourceType": "Questionnaire",
  "id": "CancerPatientDataCollection",
  "meta": {
    "profile": [
      "http://hl7.org/fhir/4.0/StructureDefinition/Questionnaire"
    ]
  },
  "url": "http://trumpetplatform.org/questionnaires/NSCLCmodif",
  "version": "5.0.0",
  "name": "TRUMPET NSCLC",
  "title": "TRUMPET NSCLC",
  "status": "draft",
  "date": "2025-09-25",
  "publisher": "IRST",
  "description": "Questionnaire to collect detailed information on cancer patient data, including genetic alterations, biomarkers, treatments, and outcomes.",
  "purpose": "To gather comprehensive information on cancer patients for research and clinical decision-making.",
  "item": [
    {
      "item": [
        {
          "type": "string",
          "linkId": "Hugo Symbol",
          "code": [
            {
              "system": "http://loinc.org",
              "code": "LP7824-8"
            }
          ],
          "text": "Hugo Symbol",
          "required": false
        },
        {
          "type": "choice",
          "extension": [
            {

```



TRUstworthy Multi-site Privacy Enhancing Technologies

```
"url": "http://hl7.org/fhir/StructureDefinition/questionnaire-itemControl",
"valueCodeableConcept": {
  "coding": [
    {
      "system": "http://hl7.org/fhir/questionnaire-item-control",
      "code": "drop-down",
      "display": "Drop down"
    }
  ]
}
},
"linkId": "Variant Classification",
"text": "Variant Classification",
"required": false
},
{
  "type": "string",
  "linkId": "HGVS",
  "text": "HGVS",
  "required": false
},
{
  "type": "decimal",
  "linkId": "Variant Allele Frequency",
  "text": "Variant Allele Frequency",
  "required": false
}
],
"type": "group",
"extension": [
  {
    "url": "http://hl7.org/fhir/StructureDefinition/questionnaire-itemControl",
    "valueCodeableConcept": {
      "coding": [
        {
          "system": "http://hl7.org/fhir/questionnaire-item-control",
          "code": "list",
          "display": "List"
        }
      ]
    }
  }
]
},
],
```



TRUStworthy Multi-site Privacy Enhancing Technologies

```
"linkId": "gene_alteration",
"text": "Gene Alterations",
"required": false,
"repeats": true,
"readOnly": false
},
{
  "item": [
    {
      "type": "decimal",
      "linkId": "PDL1",
      "text": "PDL1",
      "required": false
    },
    {
      "type": "decimal",
      "linkId": "Neutrophils",
      "text": "Neutrophils",
      "required": false
    },
    {
      "type": "decimal",
      "linkId": "Lymphocyte",
      "text": "Lymphocyte",
      "required": false
    },
    {
      "type": "decimal",
      "linkId": "Monocyte",
      "text": "Monocyte",
      "required": false
    },
    {
      "type": "decimal",
      "linkId": "Eosinophils",
      "text": "Eosinophils",
      "required": false
    },
    {
      "type": "decimal",
      "linkId": "Basophils",
      "text": "Basophils",
      "required": false
    }
  ]
}
```

```

],
"type": "group",
"linkId": "biomarker_blood_counts",
"text": "Biomarker and Blood Counts",
"required": false,
"repeats": false
},
{
"item": [
{
"type": "string",
"extension": [
{
"url": "http://hl7.org/fhir/StructureDefinition/questionnaire-itemControl",
"valueCodeableConcept": {
"coding": [
{
"system": "http://hl7.org/fhir/questionnaire-item-control",
"code": "drop-down",
"display": "Drop down"
}
]
}
}
]
},
],
"linkId": "Sex",
"text": "Sex",
"required": true
},
{
"type": "string",
"extension": [
{
"url": "http://hl7.org/fhir/StructureDefinition/questionnaire-itemControl",
"valueCodeableConcept": {
"coding": [
{
"system": "http://hl7.org/fhir/questionnaire-item-control",
"code": "drop-down",
"display": "Drop down"
}
]
}
]
}
}
]
}
}

```

```

    ],
    "linkId": "Smoking status at start of treatment",
    "text": "Smoking status at start of treatment",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Performance Status at start of treatment",
    "text": "Performance Status at start of treatment",
    "required": false
  }
],
"type": "group",
"linkId": "patient_demographics_status",
"text": "Patient Demographics and Status",
"required": true
},
{
  "item": [
    {
      "type": "string",
      "linkId": "Type of therapy",
      "text": "Type of therapy",
      "required": true
    },
    {
      "type": "integer",
      "linkId": "Therapy start in patient's age",
      "text": "Therapy start in patient's age",
      "required": false
    },
    {
      "type": "string",
      "linkId": "Best response to therapy",
      "text": "Best response to therapy",
      "required": false
    },
    {
      "type": "integer",
      "linkId": "Duration of response (in months)",
      "text": "Duration of response (in months)",
      "required": false
    }
  ],
  {

```



TRUStworthy Multi-site Privacy Enhancing Technologies

```
"type": "string",
"linkId": "Presence of disease progression",
"text": "Presence of disease progression",
"required": false
},
{
  "type": "integer",
  "linkId": "Date of disease progression in patient's age",
  "text": "Date of disease progression in patient's age",
  "required": false
}
],
"type": "group",
"linkId": "treatment_details",
"text": "Treatment Details and Outcome",
"required": true
},
{
  "item": [
    {
      "type": "integer",
      "linkId": "Date of examination in patient's age",
      "text": "Date of examination in patient's age",
      "required": false
    },
    {
      "type": "integer",
      "extension": [
        {
          "url": "http://hl7.org/fhir/StructureDefinition/questionnaire-unit",
          "valueCoding": {
            "system": "http://unitsofmeasure.org",
            "code": "mm",
            "display": "millimeter"
          }
        }
      ]
    }
  ],
  "linkId": "Biggest Lung Lesion size",
  "text": "Biggest Lung Lesion size",
  "required": false
},
{
  "type": "integer",
  "linkId": "Lung involvment",
```

```

    "text": "Lung involvment",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Pleural Effusion",
    "text": "Pleural Effusion",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Lesion Necrosis [*]",
    "text": "Lesion Necrosis [*]",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Lesion Cavitation [*]",
    "text": "Lesion Cavitation [*]",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Thoracic Lymphadenopathy",
    "text": "Thoracic Lymphadenopathy",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Abdominal Metastases",
    "text": "Abdominal Metastases",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Biggest size of Abdominal Metastasis",
    "text": "Biggest size of Abdominal Metastasis",
    "required": false
  },
  {
    "type": "integer",
    "linkId": "Brain Metastases",
    "text": "Brain Metastases",
    "required": false
  }

```

```

    },
    {
      "type": "integer",
      "linkId": "Bone Metastases",
      "text": "Bone Metastases",
      "required": false
    }
  ],
  "type": "group",
  "linkId": "metastasis_lesion_details",
  "text": "Metastasis and Lesion Details",
  "required": false
},
{
  "item": [
    {
      "type": "string",
      "extension": [
        {
          "url": "http://hl7.org/fhir/StructureDefinition/questionnaire-itemControl",
          "valueCodeableConcept": {
            "coding": [
              {
                "system": "http://hl7.org/fhir/questionnaire-item-control",
                "code": "drop-down",
                "display": "Drop down"
              }
            ]
          }
        }
      ]
    }
  ],
  "linkId": "Vital Status",
  "text": "Vital Status",
  "required": false
},
{
  "type": "integer",
  "linkId": "Date of death or last follow-up in patient's age",
  "text": "Date of death or last follow-up in patient's age",
  "required": false
}
],
"type": "group",
"linkId": "vital_status_follow_up",

```



TRUStworthy Multi-site Privacy Enhancing Technologies

```
    "text": "Vital Status and Follow-up",
    "required": true
  }
]
}
```

Here is a demo QuestionnaireResponse resource -

```
{
  "resourceType": "QuestionnaireResponse",
  "id": "20",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2025-09-29T12:39:54.149+00:00"
  },
  "questionnaire": "http://trumpetplatform.org/questionnaires/NSCLC_IRST",
  "status": "completed",
  "subject": {
    "type": "Patient",
    "identifier": {
      "system": "urn:ietf:rfc:3986",
      "value": "patient-100"
    }
  },
  "authored": "2025-09-26T11:22:18.029760Z",
  "item": [
    {
      "linkId": "biomarker_blood_counts",
      "text": "Biomarker and Blood Counts",
      "item": [
        {
          "linkId": "PDL1",
          "text": "PDL1",
          "answer": [
            {
              "valueDecimal": 65.0
            }
          ]
        },
        {
          "linkId": "Neutrophils",
          "text": "Neutrophils",
          "answer": [
            {

```



TRUStworthy Multi-site Privacy Enhancing Technologies

```
        "valueDecimal": 85.7
      }
    ]
  },
  {
    "linkId": "Lymphocyte",
    "text": "Lymphocyte",
    "answer": [
      {
        "valueDecimal": 9.5
      }
    ]
  },
  {
    "linkId": "Monocyte",
    "text": "Monocyte",
    "answer": [
      {
        "valueDecimal": 4.5
      }
    ]
  },
  {
    "linkId": "Eosinophils",
    "text": "Eosinophils",
    "answer": [
      {
        "valueDecimal": 0.1
      }
    ]
  },
  {
    "linkId": "Basophils",
    "text": "Basophils",
    "answer": [
      {
        "valueDecimal": 0.2
      }
    ]
  }
]
},
{
  "linkId": "patient_demographics_status",
```

```

"text": "Patient Demographics and Status",
"item": [
  {
    "linkId": "Sex",
    "text": "Sex",
    "answer": [
      {
        "valueString": "female"
      }
    ]
  },
  {
    "linkId": "Smoking status at start of treatment",
    "text": "Smoking status at start of treatment",
    "answer": [
      {
        "valueString": "former"
      }
    ]
  },
  {
    "linkId": "Performance Status at start of treatment",
    "text": "Performance Status at start of treatment",
    "answer": [
      {
        "valueInteger": 2
      }
    ]
  }
],
{
  "linkId": "treatment_details",
  "text": "Treatment Details and Outcome",
  "item": [
    {
      "linkId": "Type of therapy",
      "text": "Type of therapy",
      "answer": [
        {
          "valueString": "monotherapy"
        }
      ]
    }
  ],
},

```

```

{
  "linkId": "Therapy start in patient's age",
  "text": "Therapy start in patient's age",
  "answer": [
    {
      "valueInteger": 50
    }
  ]
},
{
  "linkId": "Best response to therapy",
  "text": "Best response to therapy",
  "answer": [
    {
      "valueString": "partial response"
    }
  ]
},
{
  "linkId": "Duration of response (in months)",
  "text": "Duration of response (in months)",
  "answer": [
    {
      "valueInteger": 7
    }
  ]
},
{
  "linkId": "Presence of disease progression",
  "text": "Presence of disease progression",
  "answer": [
    {
      "valueString": "no"
    }
  ]
}
],
{
  "linkId": "vital_status_follow_up",
  "text": "Vital Status and Follow-up",
  "item": [
    {
      "linkId": "Vital Status",

```

```

    "text": "Vital Status",
    "answer": [
      {
        "valueString": "dead"
      }
    ]
  }
]
}
]
}
]
}
}

```

3.2.2.2 API endpoints

The Data Owner Node API is the on-prem backbone for site operations. It powers the local dashboard, brokers secure calls to the HAPI proxy, and issues credentials for controlled Cloud-Node communication. Every request is authenticated and logged; only metadata and model signals leave the site, never raw clinical data.

Endpoints are resource- or action-oriented but remain predictable and RESTful. Registration, login, logout, and password changes manage local identities; API keys gate internal access to the dashboard API and HAPI proxy; and cloud access tokens enable publishing and federated coordination. The FHIR endpoint accepts QuestionnaireResponse posts to the local store through the proxy, preserving privacy while keeping clinical records on site.

Task	Method	URI
Register	POST	/register
Login	POST	/login
Logout	POST	/logout
Change Password	POST	/change-password
Generate API Key	POST	/api-key
Remove API Key	DELETE	/api-key
Create Cloud Access Token	POST	/cloud-access-token/
Create FHIR QuestionnaireResponse	POST	/fhir/QuestionnaireResponse
List of Datasets	GET	/datasets/
Dataset Details	GET	/datasets/:id/
Create Dataset	POST	/datasets/
Update Dataset	PATCH	/datasets/:id/
Publish a Dataset	POST	/datasets/:id/publish/

Unpublish a Dataset	POST	/datasets/:id/unpublish/
Generate Dataset Statistics	POST	/datasets/:id/generate-statistics/

Table 13: APIs of data owner node

3.3 Cloud–Node Interaction

Study work begins in the Cloud: researchers assemble cohorts, define model settings, and submit a training request. Data Owner Admins review the proposal in their Cloud dashboard—checking scope, datasets, and terms—and approve or decline. Once approved, orchestration signals move to the site. From that point on, training and data processing happen inside the Data Owner Node. The Cloud coordinates rounds and schedules, and a webhook channel streams status, metrics, and errors back to the study workspace so progress is visible without exposing any underlying records.

Only the minimum needed crosses the boundary. Metadata, approvals, model definitions, and privacy-preserving outputs flow between Cloud and site; raw clinical data never leaves hospital infrastructure. The contract is enforced through documented REST endpoints with stable payloads, role-scoped authentication, and auditing. This arrangement keeps responsibilities clear—Cloud for discovery and coordination, Node for execution on local data—while maintaining a clean, verifiable interface that supports real-world operations and compliance.

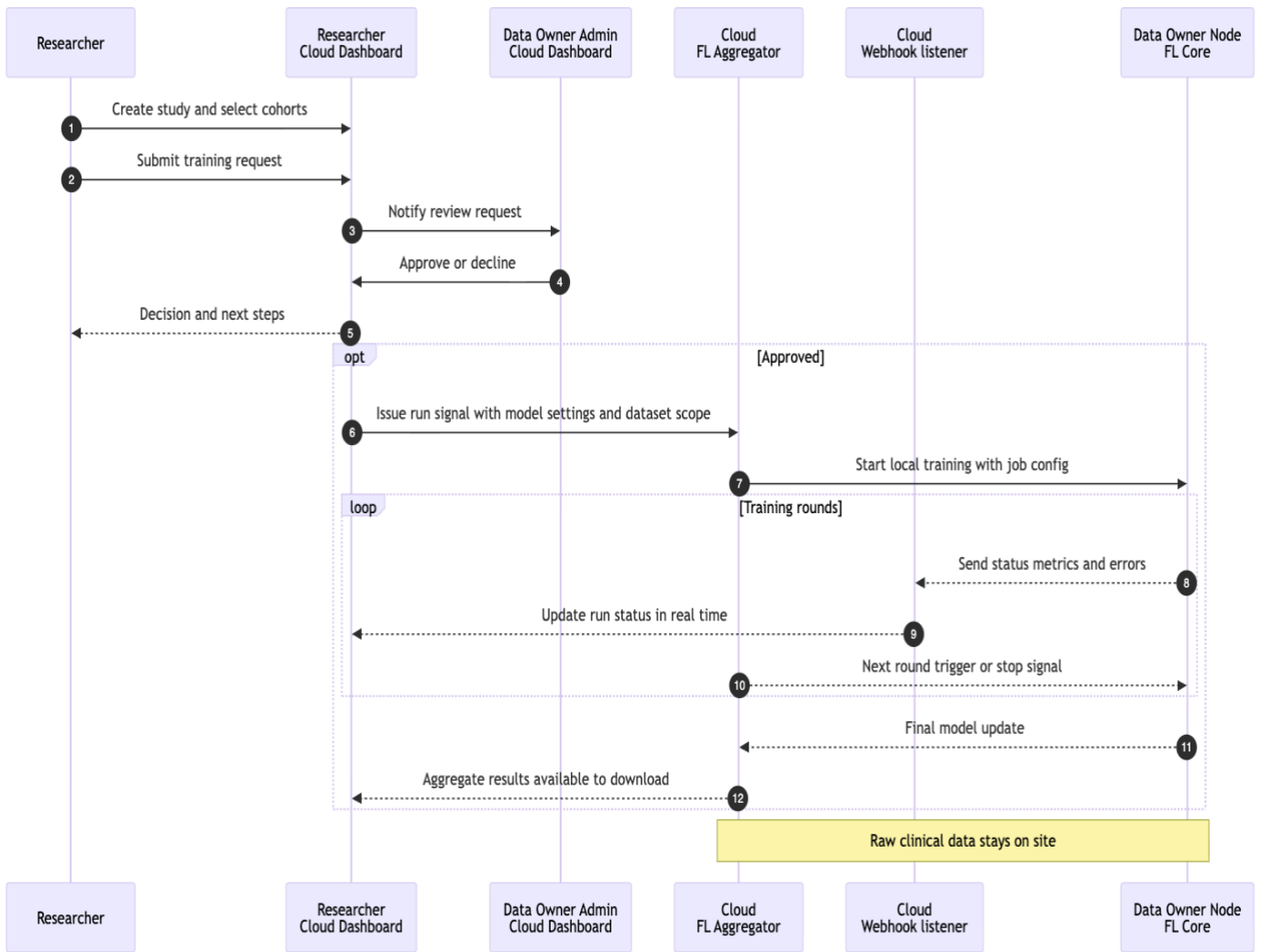


Figure 11: Cloud–Node Study Orchestration

3.4 SDLC & Co-Creation Cadence

TRUMPET is built, tested, released, and refined through an iterative, lightweight process designed for continuous feedback from researchers and data owners. Short sprints produce incremental releases that feed two pilot waves: an initial pass to collect real-world input and a follow-up to harden the platform. The approach prioritises flexibility as requirements evolve, keeping privacy and usability at the centre of each change.

Delivery runs on 2–3 week sprints culminating in live demos covering the Cloud and the Data Owner Node. A monthly retrospective with partners surfaces what worked, what didn't, and where to improve next. Weekly cross-team syncs (TVS, GRAD, INRIA) align development, unblock issues, and plan handoffs. At the same time, targeted hospital check-ins validate node usability, data flows, and deployment steps when deeper input is needed.

Quality and security are treated as continuous activities rather than milestones. Penetration testing occurs in phases, with findings addressed and re-validated before closing. Internal QA executes ongoing functional and regression suites, raising change requests that flow straight into the backlog. Threat modelling and privacy reviews are part of the Definition of Done for features touching FL aggregation, dashboards, and GDPR-focused checks from WP3 are built into pilot runs.

Agile mechanics keep work traceable and testable. Platform epics—such as data ingestion, dashboards, and FL orchestration—are decomposed into sprint stories with clear acceptance criteria. Sprint reviews capture feedback from researchers and data owners, while retrospectives drive process and quality improvements. Backlog items map to WP requirements and WP4 tasks (T4.1–T4.2), maintaining a clean line from requirement to design to test.

Co-creation anchors the product direction. WP1 facilitates workshops and touchpoints with clinicians, researchers, and data owners, while an Advisory Board offers tactical guidance. Design focuses on process flows first—modelling use cases and functional units before APIs or UI—so discussions stay concrete and user-led. Round-1 prototypes emphasise usability, scalability, and privacy signals; Round-2 incorporates lessons learned and extends features.

Roles and handoffs are explicit. WP4 leads platform and dashboard implementation and coordinates on-prem deployment with hospitals. Data owners and researchers provide ongoing usage feedback through demos and pilots, directly shaping backlog priorities. The operating model is complemented by living documentation: versioned materials describe requirements, flows, APIs, and deployment, with open repositories aligned to FAIR/EOSC principles for outputs.

Success is measured by delivery and adoption. Sprint velocity, release predictability, and change lead time track engineering health, while pilot task completion, stakeholder satisfaction, and sprint-by-sprint feedback closure indicate real-world usability. The cadence keeps teams close to users, delivers value in small steps, and embeds privacy and quality into every change—from design to deployment.

4 Data Acquisition & Integration

In this project, Data Acquisition and integration is the on-premise pipeline each hospital runs to load local clinical data into a privacy-preserving format and make it usable for federated studies. Practically, it ingests data from the hospital’s own sources, converts and harmonises it into HL7 FHIR, stores it in a hospital-hosted HAPI FHIR repository, and exposes only safe, privacy-controlled metadata and statistics to the TRUMPET Cloud. Raw data never leaves the site; only what’s necessary to conduct studies and coordinate training is shared.

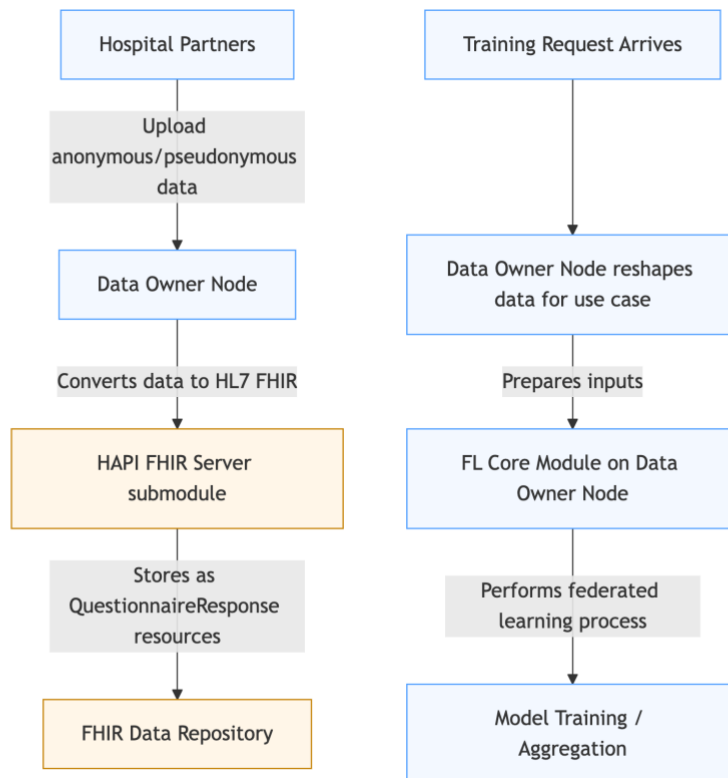


Figure 12: Data Flows inside Data Owner Node

4.1 Ingestion Pipelines

Our data pipeline is hospital-first by design. Each partner connects local sources (databases, warehouses, or exports) to a Data Owner Node that runs on-premise. Incoming records are harmonised to a healthcare-native standard—HL7 FHIR—so downstream analytics and federated training see a consistent view without moving raw patient data off-site. Before any study runs, the node validates structure, semantics, and privacy posture; only privacy-safe metadata and model updates are ever exposed to the TRUMPET Cloud.

Connectivity and canonicalisation are handled “**FHIR-first.**” The node uses HAPI FHIR as its backbone, fronted by a secure proxy. Uploads are authenticated before persistence,

and the rollout includes practical sizing and indexing plans so hospitals know how much to provision and how their local index aligns with central discovery. For pilots, we focus on **Questionnaire** and **QuestionnaireResponse** resources, giving us a clear canonical model while keeping the door open to broader FHIR profiles later.

Pre-processing keeps transformations traceable and repeatable. Sources are extracted via connectors (*SQL/API*), mapped to the appropriate FHIR resources, normalised for codes/units, enriched where available, then written into the HAPI FHIR store and a specialised index. These steps are run via REST APIs and documented schemas to minimise manual work and make replays safe.

Quality is enforced at the edge. The node checks conformance to FHIR profiles and required fields, validates terminology, and verifies integrity across linked resources. It also computes basic statistics and statistical information locally so researchers can assess study-ready information without touching patient-level data. The Researcher Dashboard surfaces these metadata and summaries to ground cohort design and study decisions.

Governance is built in. Every upload, validation, approval, and study run is logged; role-based access controls cover registration, authentication, and authorisation, with token revocation. If a load fails validation, the offending records or batches are rolled back with clear error codes so owners can fix mappings and replay safely. The proxy and caching layer shield the FHIR store and enable controlled retries, which are still entirely on-premise.

Once a dataset passes checks and is published, researchers can discover it, request model development or training, and proceed.

4.2 Adapters & Connectors

This part of the platform lets hospitals plug their own systems into TRUMPET with minimal fuss. We drop in lightweight **adapters** that read from local sources (SQL, APIs, exports), convert records into **HL7 FHIR**, and store them in a hospital-hosted **HAPI FHIR** repository.

Each Data Owner Node runs HAPI FHIR behind a hardened **Traefik** reverse proxy. Traefik is the front door: it terminates TLS, applies rate limits, and only exposes the necessary endpoints. Behind that door, the HAPI Proxy and HAPI FHIR handle the clinical payloads. Adapters talk to this internal FHIR endpoint, and the TRUMPET Cloud only sees what's allowed (metadata, job requests, and privacy-safe outputs).

Why Traefik matters

Think of Traefik as the bouncer and traffic cop in one:

- It **locks down ports** and routes only approved paths to the right service.

- It keeps **HAPI FHIR out of sight**, so only vetted adapters and trusted systems can reach it.

4.3 Operationalising FHIR

FHIR isn't just our file format; it's the way the system runs day to day. Hospitals keep data on-prem in a **FHIR-compliant HAPI FHIR repository**, and researchers work with consistent, queryable resources—without raw patient data ever leaving hospital premises. Every integration, cohort selection, exploratory stat, and training request flows through FHIR resources and APIs, not custom schemas. The platform requires **FHIR-compliant local repositories**, so this isn't optional plumbing—it's the operating fabric.

Why FHIR is the backbone

Using one canonical language (FHIR) gives uniform semantics across sites without forcing hospitals to remodel their databases. It also gives us reliable, secure endpoints for automated analytics and federated learning. Duties stay cleanly separated: **hospitals manage FHIR data and privacy**, while the **cloud orchestrates discovery, jobs, and aggregation**. This design flows through our architecture, APIs, and UI so teams can work fast and safely.

How it works

All clinical records we use in pilots are stored as **FHIR QuestionnaireResponse** resources in HAPI FHIR. Each response points to a **Questionnaire canonical URL** (its "resource URL"). That single link is how we group records into datasets and run queries consistently across sites.

Creating or updating a response in standard FHIR

```
POST {baseUrl}/QuestionnaireResponse
Content-Type: application/fhir+json
```

Querying a dataset by its questionnaire is just as simple

```
GET {baseUrl}/QuestionnaireResponse?questionnaire={canonicalUrl}
# Example:
# GET
https://hapi.example.org/fhir/QuestionnaireResponse?questionnaire=http://trumpet.cloud/fhir/Questionnaire/nsclc-v1
```

Working with multiple datasets

If two **QuestionnaireResponse** records use the same **Questionnaire** canonical URL (*for example, .../Questionnaire/nsclc-v1*), they belong to the same dataset. Hospitals upload responses with the correct questionnaire URL; the Data Owner Dashboard then lets group by that URL, compute local statistics, preview charts, and publish to the cloud—sharing only metadata and privacy-safe summaries, never raw records.

5 Platform & Dashboards

This section shows how TRUMPET lets researchers work with hospital-held data without moving raw records, using HL7 FHIR as the standard model and HAPI FHIR (via a secure proxy) on each Data Owner node. It summarises the two dashboards: the cloud Researcher Dashboard for dataset discovery, cohorting, quick stats, model selection and job tracking; and the on-prem Data Owner Control Panel for dataset curation, approvals and node health. Privacy is enforced end-to-end through centralised access control and PETs, which include differential privacy, homomorphic encryption, and secure MPC. At the same time, an FL Core on each node participates in secure training rounds so only safe, aggregate signals leave the site.

The section then walks through complete user journeys for researchers, data owners and admins; explains the federated training round-trip and the typical artefacts produced; details API design and versioning around FHIR resources; clarifies the database split between cloud metadata and local FHIR persistence, including what is stored where and how PET outputs/federation state are recorded; and closes with observability and operations across cloud and hospital nodes—metrics, logs, traces, health checks, and practical runbooks.

5.1 Researcher Dashboard

The Researcher Dashboard is where data scientists and clinicians discover datasets, set up studies and agreements, build models, and run federated training—without accessing raw patient data.

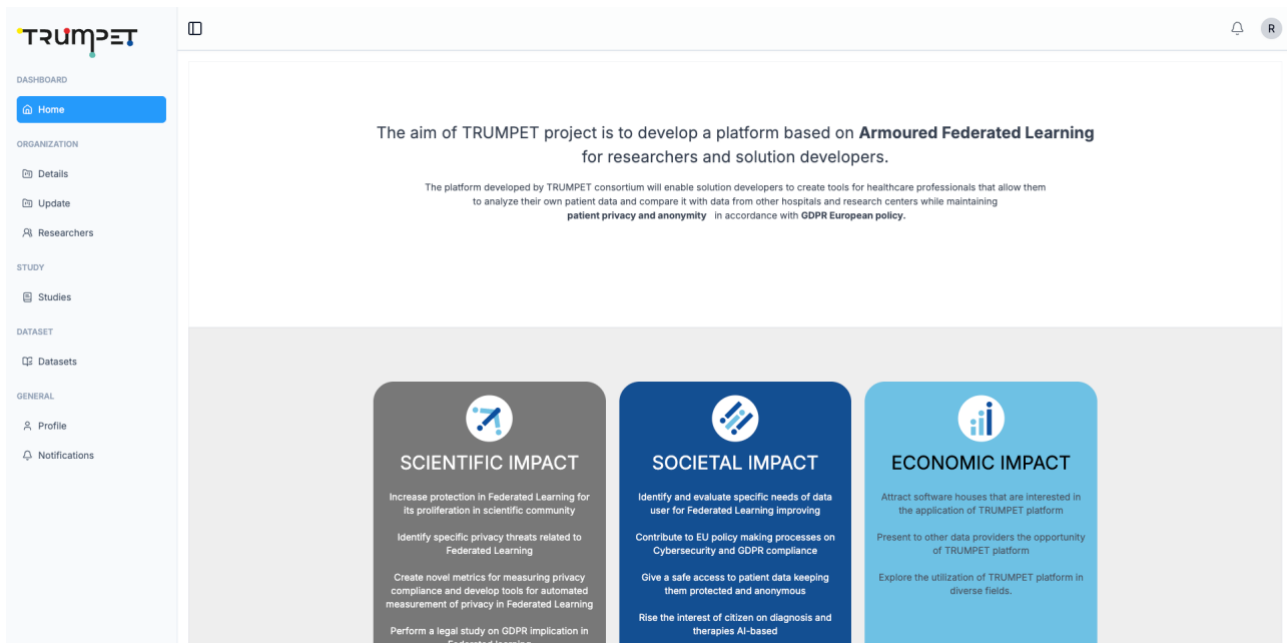


Figure 13: TRUMPET Cloud Researcher Dashboard

It walks users through a simple, step-by-step workflow and enforces PETs, study approvals, and full audit trails. Admin-level researchers can also send invitations to onboard new fellow researchers.

What users can do

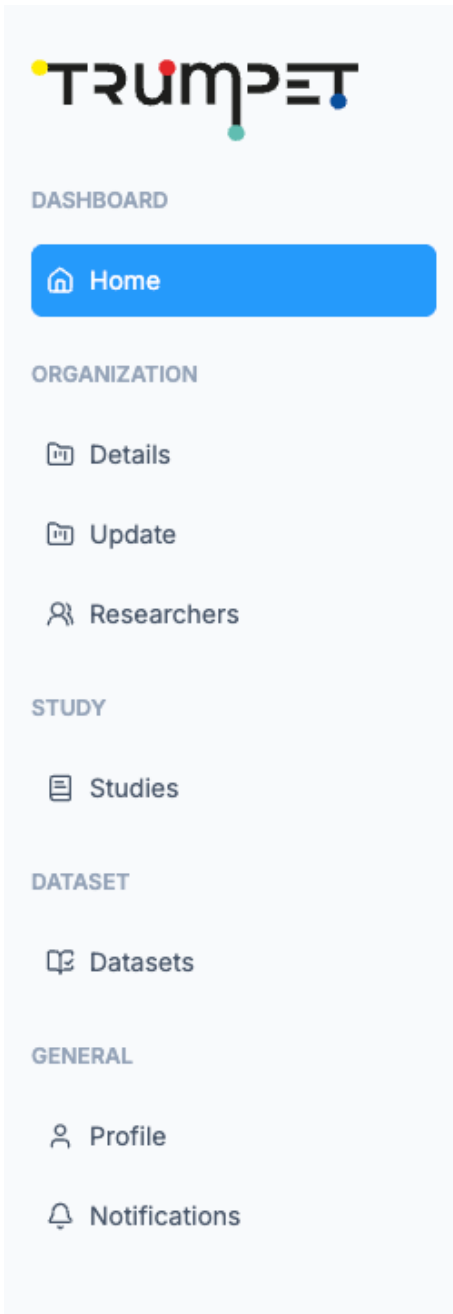


Figure 14: Researcher Dashboard Menu

Find and assess datasets published by Data Owners, review metadata, and review basic statistics. Screenshots include published dataset lists and details pages (metadata, statistical report, owner profile).

Create a study and its agreements: name the study, choose one or more datasets, select the PET methods (including details), and pick the use case and scenarios. The platform then generates the study with its linked agreements. A single study can include multiple agreements, each with its own datasets and settings.

Once dataowners approve, the model is trained across participating sites. Training requests are routed to Data Owner nodes, and results are returned to the study agreement record.

Download results (e.g., metrics, artefacts permitted by policy) and track notifications, agreements, and status from the study details page.

How privacy is controlled

Armoured FL by design: Model updates are protected by PETs (CDC_DP, FedAvg, ThHE) selected and combined for the use case; researchers interact normally while protections are applied under the hood.

Data Owner gatekeeping: Based on PET checks and the dataset's policies, each request can be accepted or rejected at the node—no raw data leaves the hospital.



TRUStworthy Multi-site Privacy Enhancing Technologies

Condensed Workflows & Capabilities

Study management follows a simple flow: browse published datasets and inspect their stats; create a study, choose datasets and define the cohort; add one or more study agreements that capture datasets, PET methods, required labels, sample-size ratios and other parameters; submit the federated training job, after which each Data Owner reviews and approves or rejects from their own dashboard; monitor notifications and download the results.

Onboarding a new researcher is simple: send an email invite, the researcher confirms and registers, and a researcher admin reviews and approves the request.

The dashboard unifies dataset discovery (via the Advertised Dataset service), end-to-end study management (descriptions, agreements, notifications, results), built-in invitations for onboarding, and role-based access with token-based authentication. Its UI mirrors the cloud flows and wireframes—lists, details, study creation, model setup, training submission, and results download—so work remains predictable, auditable, and familiar to ML practitioners.

Why it matters- The dashboard hides the complexity of multi-site, cross-border FL and keeps researchers in a standard ML loop—discover, define, model, train, review—while PETs, and approval workflows run in the background in line with GDPR-first principles.

5.2 Data Owner Control Panel

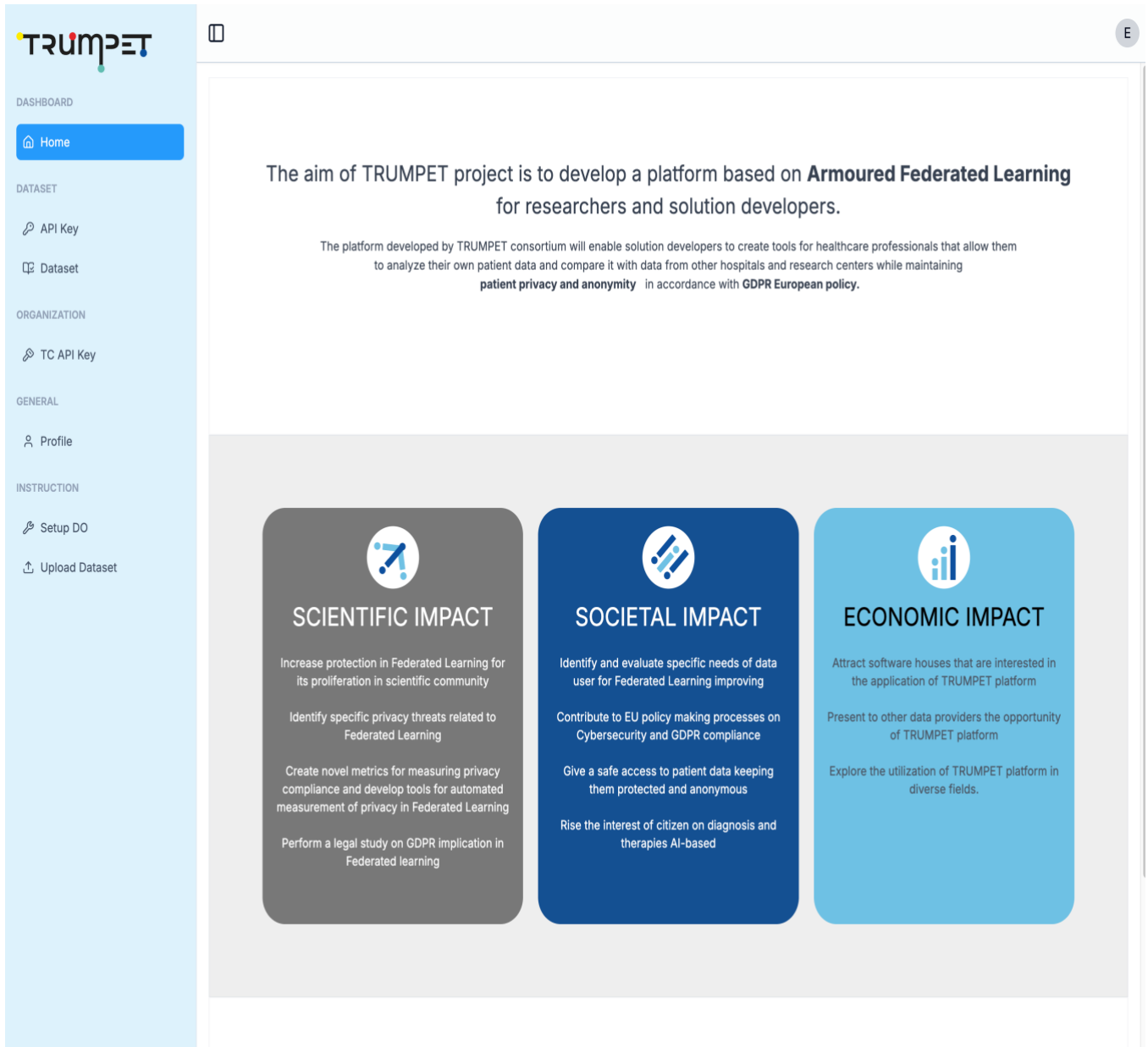
To balance privacy with operational oversight, the Data Owner has two dashboards with clear, complementary roles:

Local (on-prem) dashboard— Deployed inside the hospital network and reachable only via port forwarding, this dashboard is never exposed to the public internet. It's used for day-to-day node administration, dataset upload via the HAPI FHIR stack, metadata management, and monitoring/health checks. No raw patient data leaves the premises; all actions occur within the local environment.

Cloud Data Owner dashboard— The cloud view provides governance, not data handling. Admins review and act on study agreements, track which datasets have been advertised/published to researchers, and see participation status across studies—without exposing underlying hospital data. This separation keeps research workflow coordination in the cloud while reserving all sensitive operations to the local node.

5.2.1 Dashboard on Data Owner Node

The Data Owner Control Panel is the local, on-premise console hospitals use to prepare, govern, and monitor their participation in TRUMPET. It sits inside the hospital network—next to the data and the FHIR server—so no raw patient data leaves the site. From here, authorised personal/IT admins can upload anonymous/pseudonymous data.



TRUMPET

DASHBOARD

- Home

DATASET

- API Key
- Dataset

ORGANIZATION

- TC API Key

GENERAL

- Profile

INSTRUCTION

- Setup DO
- Upload Dataset

The aim of TRUMPET project is to develop a platform based on **Armoured Federated Learning** for researchers and solution developers.

The platform developed by TRUMPET consortium will enable solution developers to create tools for healthcare professionals that allow them to analyze their own patient data and compare it with data from other hospitals and research centers while maintaining **patient privacy and anonymity** in accordance with **GDPR European policy**.

SCIENTIFIC IMPACT

- Increase protection in Federated Learning for its proliferation in scientific community
- Identify specific privacy threats related to Federated Learning
- Create novel metrics for measuring privacy compliance and develop tools for automated measurement of privacy in Federated Learning
- Perform a legal study on GDPR implication in Federated learning

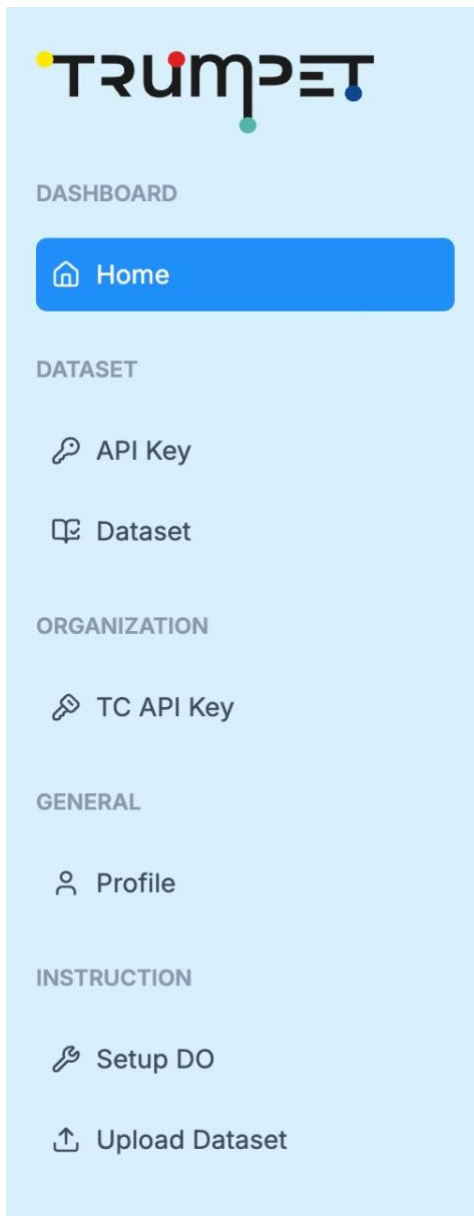
SOCIETAL IMPACT

- Identify and evaluate specific needs of data user for Federated Learning improving
- Contribute to EU policy making processes on Cybersecurity and GDPR compliance
- Give a safe access to patient data keeping them protected and anonymous
- Rise the interest of citizen on diagnosis and therapies AI-based

ECONOMIC IMPACT

- Attract software houses that are interested in the application of TRUMPET platform
- Present to other data providers the opportunity of TRUMPET platform
- Explore the utilization of TRUMPET platform in diverse fields.

Figure 15: Data Owner Node on premise Admin Dashboard



What Users can perform

Upload data (FHIR): Hospitals use HTTP REST endpoints to send data to the local HAPI FHIR server through a hardened HAPI Proxy for authentication. Uploads follow a predefined QuestionnaireResponse template. The proxy authenticates and isolates all traffic.

Create a Dataset record (Dashboard → PostgreSQL): A clean UI lets you define a dataset that points to a group of HAPI FHIR resources sharing the same Questionnaire URL. You can also add key metadata such as title, use case, temporal coverage, geospatial coverage, and DOI citation.

Generate dataset statistics: Backend routines compute summary stats and produce charts so teams can quickly understand data quality and coverage.

Publish a dataset: The dashboard publishes the dataset with one action. Only metadata and statistics are securely pushed to the TRUMPET Cloud, making the dataset discoverable to researchers—no raw data leaves the site.

Store Cloud API key: Save the Cloud API key in the node so the system can securely upload dataset metadata and statistics when needed.

Figure 16: Data Owner Node on premise dashboard menu

5.2.2 Dashboard on Cloud

The Cloud-based Data Owner Dashboard is where hospitals govern and monitor their participation in TRUMPET. Authorised members can review and approve study agreements, manage secure keys for cloud↔node communication, view their own datasets as seen by researchers, and access step-by-step guidance to install the on-prem Data Owner Node and upload FHIR data.

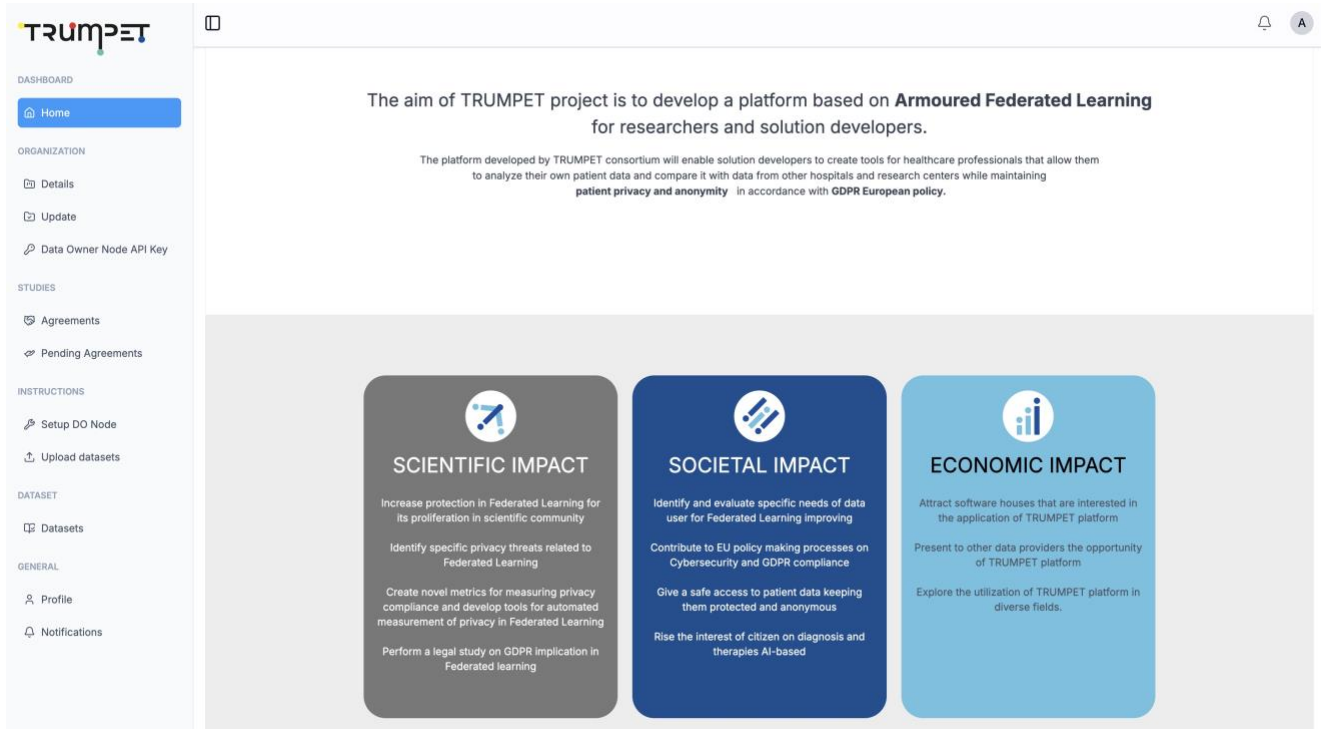
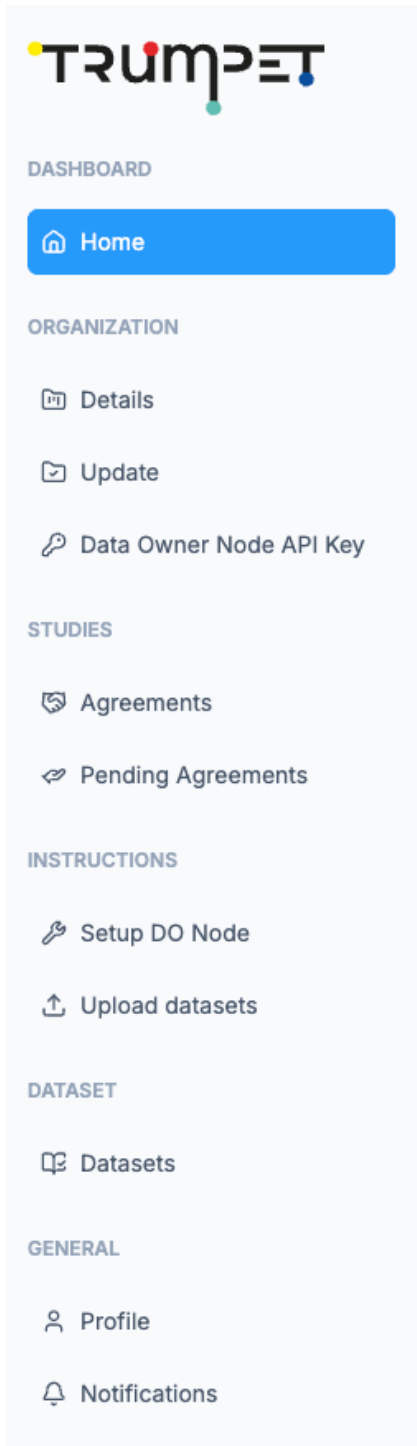


Figure 17: Data Owner dashboard on TRUMPET Cloud



What Users can perform

Monitor and control permissions – Review incoming study agreement requests, see request details and researcher organisation info, and approve or decline with a clear audit trail.

Follow setup guidance – Access simple, step-by-step instructions for installing your on-prem node and uploading HL7 FHIR data.

Manage secret/API keys—Issue, rotate, and revoke keys to keep cloud–node communication secure.

View and verify datasets – Check that your published metadata and statistics (not raw data) are displayed correctly to researchers.

5.3 End-to-End User Flows

This section walks through what happens—step by step—when a researcher runs a study and when a data owner prepares and governs datasets. It ties together the dashboards, privacy controls, and the federated training round-trip.

5.3.1 Researcher flow (Cloud Dashboard)

Goal: Discover datasets, design a study, and train a model without accessing raw patient data.

Sign-in and role check: The researcher signs in. The system validates the token and applies RBAC to unlock researcher features such as studies, models, and analytics.

Browse published datasets: From Published Datasets, the researcher explores dataset cards and details—metadata, descriptive statistics, and data owner information—directly in the cloud dashboard.

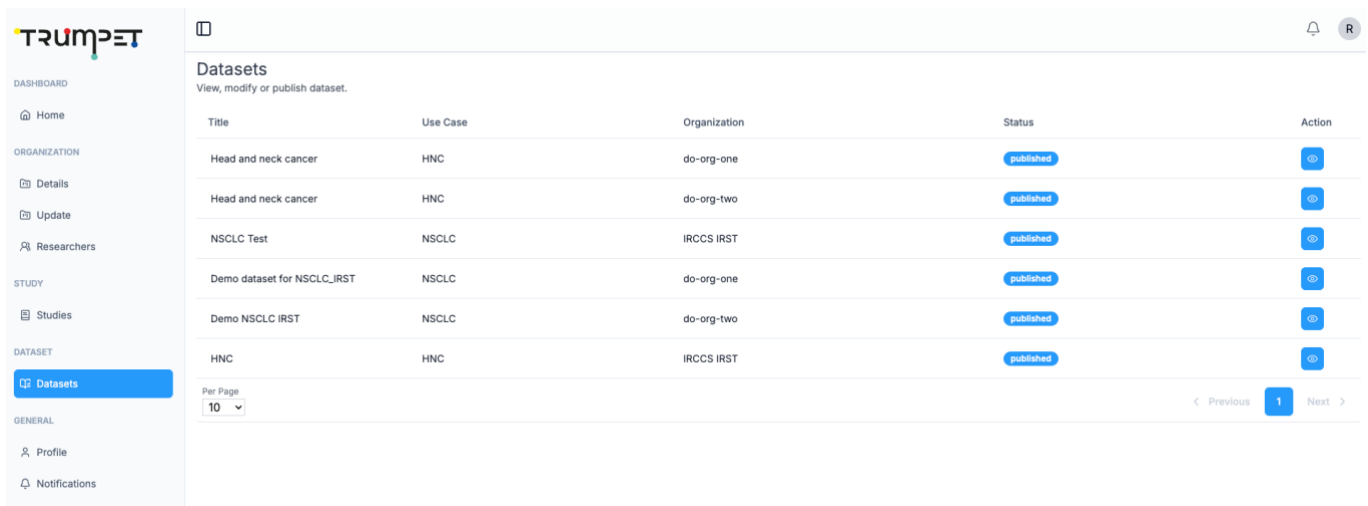
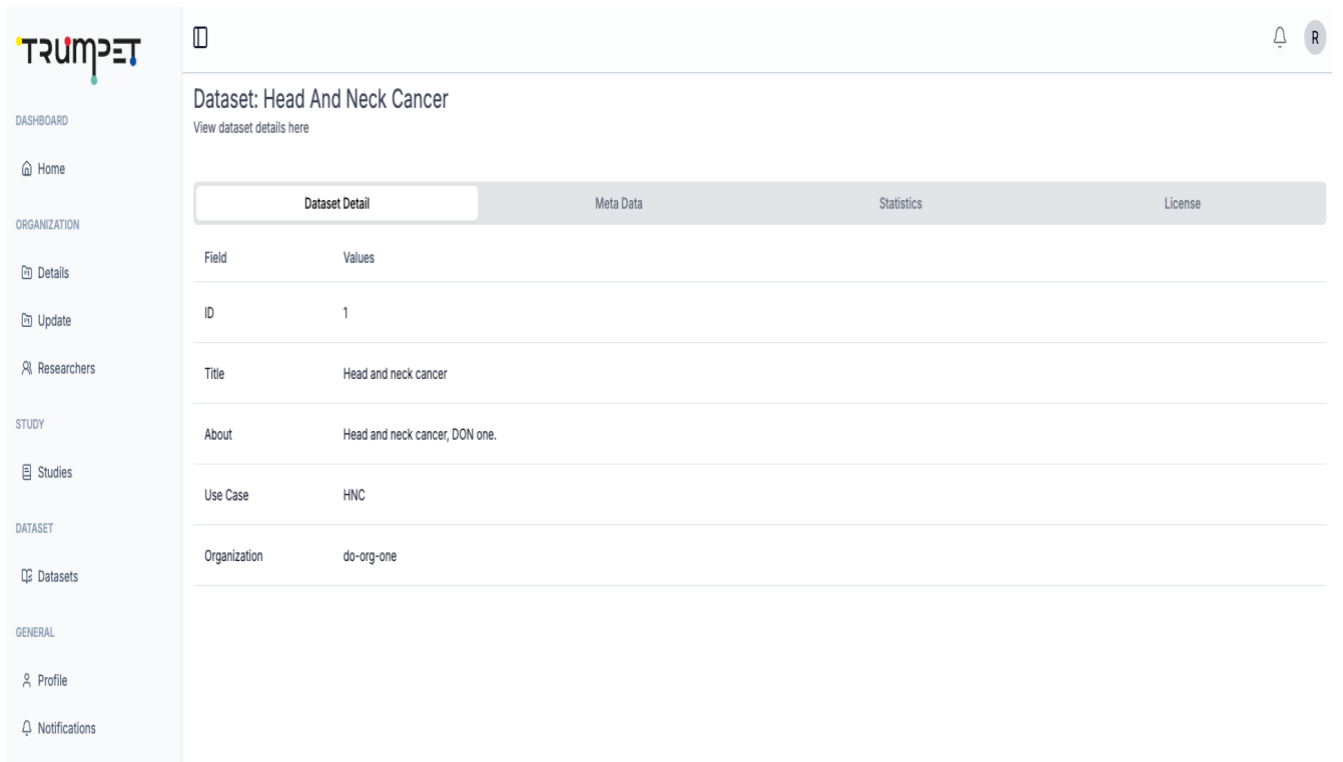


Figure 18: Researcher Flow - Cloud - Browse Available Datasets

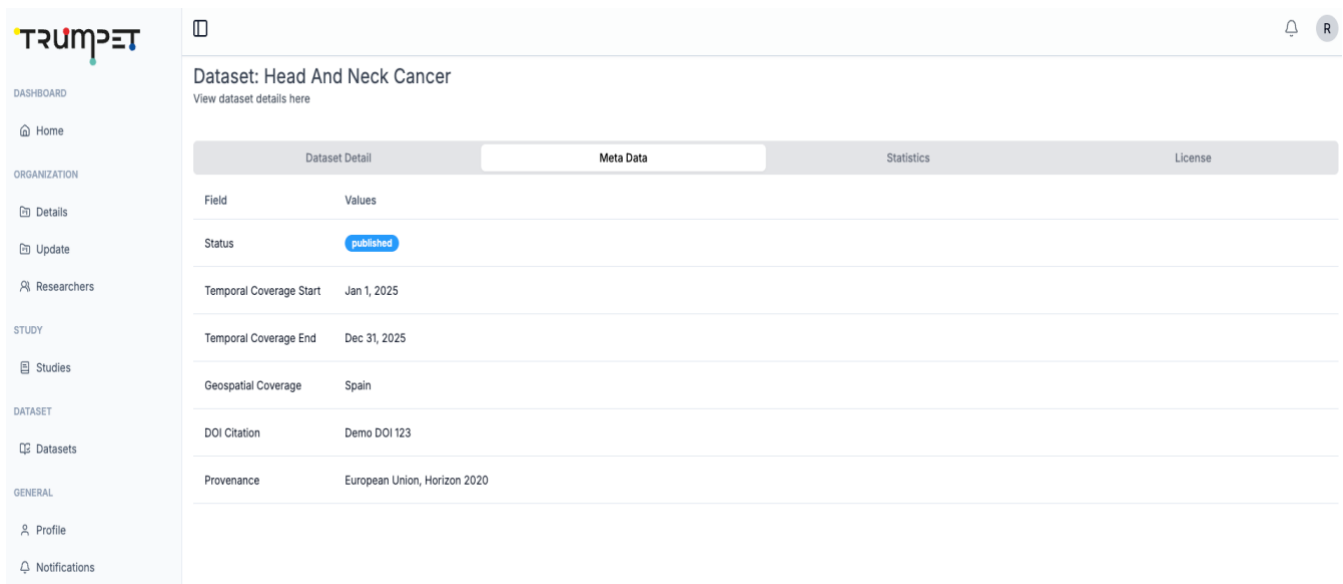


Dataset: Head And Neck Cancer
View dataset details here

Navigation tabs: Dataset Detail (selected), Meta Data, Statistics, License

Field	Values
ID	1
Title	Head and neck cancer
About	Head and neck cancer, DON one.
Use Case	HNC
Organization	do-org-one

Figure 19: Researcher Flow - Cloud - View Dataset Details



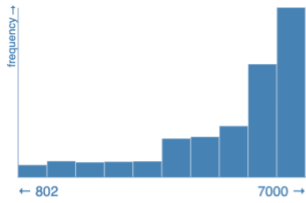
Dataset: Head And Neck Cancer
View dataset details here

Navigation tabs: Dataset Detail, Meta Data (selected), Statistics, License

Field	Values
Status	published
Temporal Coverage Start	Jan 1, 2025
Temporal Coverage End	Dec 31, 2025
Geospatial Coverage	Spain
DOI Citation	Demo DOI 123
Provenance	European Union, Horizon 2020

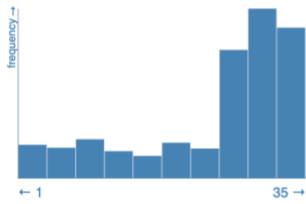
Figure 20: Researcher Flow - Cloud - View Dataset Metadata

OIS_TOTAL_DOSE



Valid	2505	100%
Missing	0	0%
Mean	5450.5345	
Std. Deviation	1524.9518	
Quantiles		
	802	Min
	4711	25%
	6110	50%
	6575	75%
	7000	Max

OIS_NB_FRACTIONS

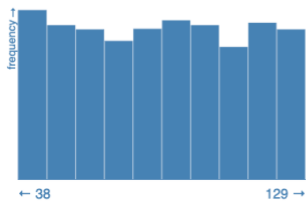


Valid	2505	100%
Missing	0	0%
Mean	24.4303	
Std. Deviation	9.4114	
Quantiles		
	1	Min
	19	25%
	28	50%
	31	75%
	35	Max

ENT_VISIT_DATE

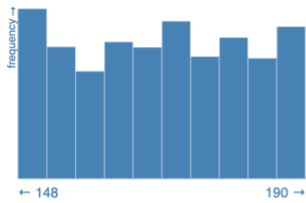
Valid	2505	100%
Missing	0	0%

WEIGHT



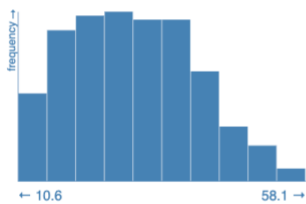
Valid	2505	100%
Missing	0	0%
Mean	82.9812	
Std. Deviation	26.4209	
Quantiles		
	38	Min
	60	25%
	83	50%
	106	75%
	129	Max

HEIGHT



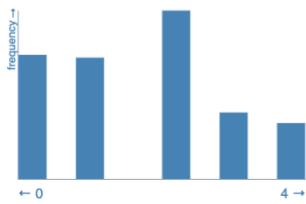
Valid	2505	100%
Missing	0	0%
Mean	168.7613	
Std. Deviation	12.3930	
Quantiles		
	148	Min
	158	25%
	169	50%
	180	75%
	190	Max

BMI



Valid	2505	100%
Missing	0	0%
Mean	29.5626	
Std. Deviation	10.2902	
Quantiles		
	10.6	Min
	21.1	25%
	29	50%
	36.7	75%
	58.1	Max

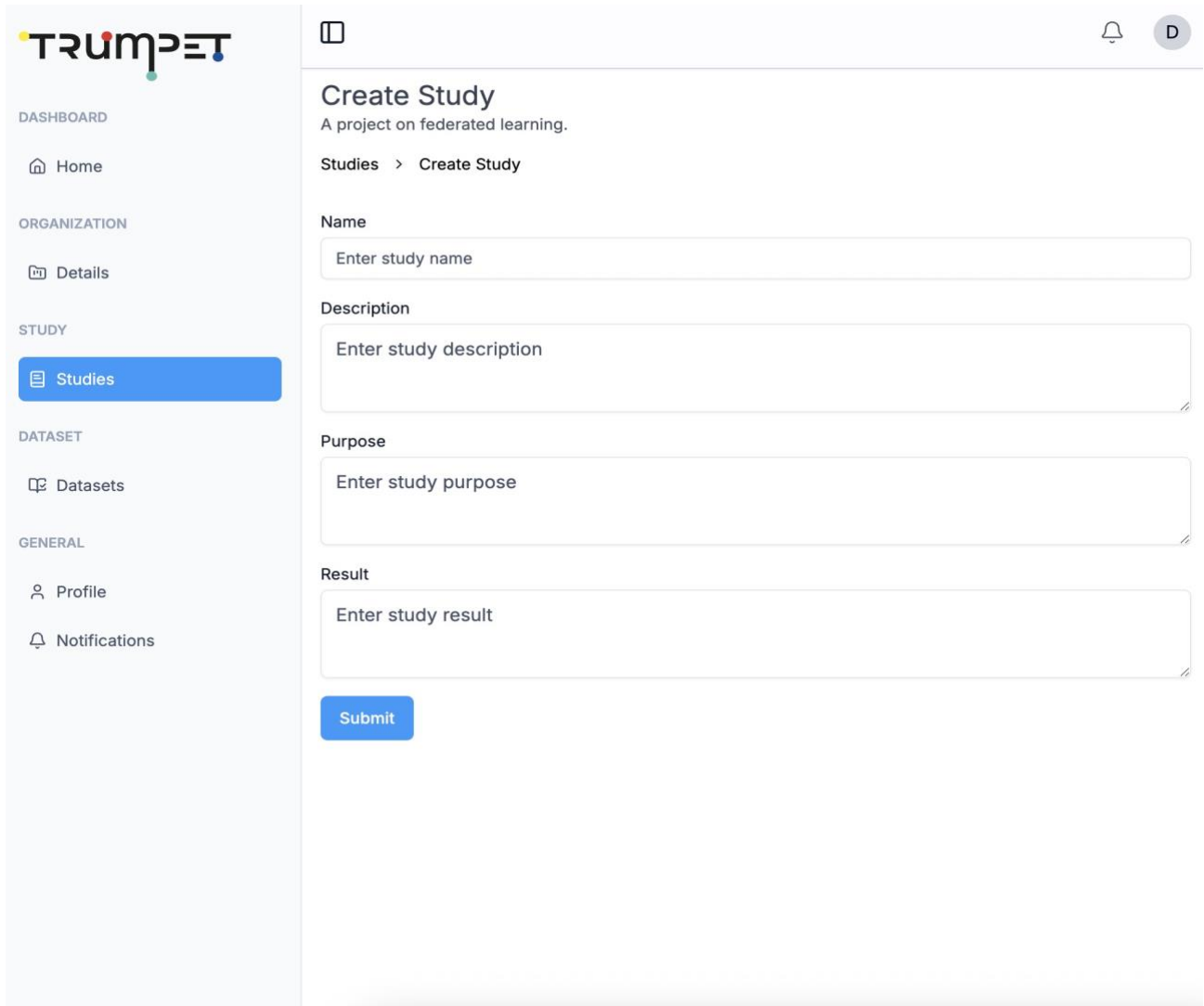
ECOG



Valid	2505	100%
Missing	0	0%
Mean	1.6403	
Std. Deviation	1.2507	
Quantiles		
	0	Min
	1	25%
	2	50%
	2	75%
	4	Max

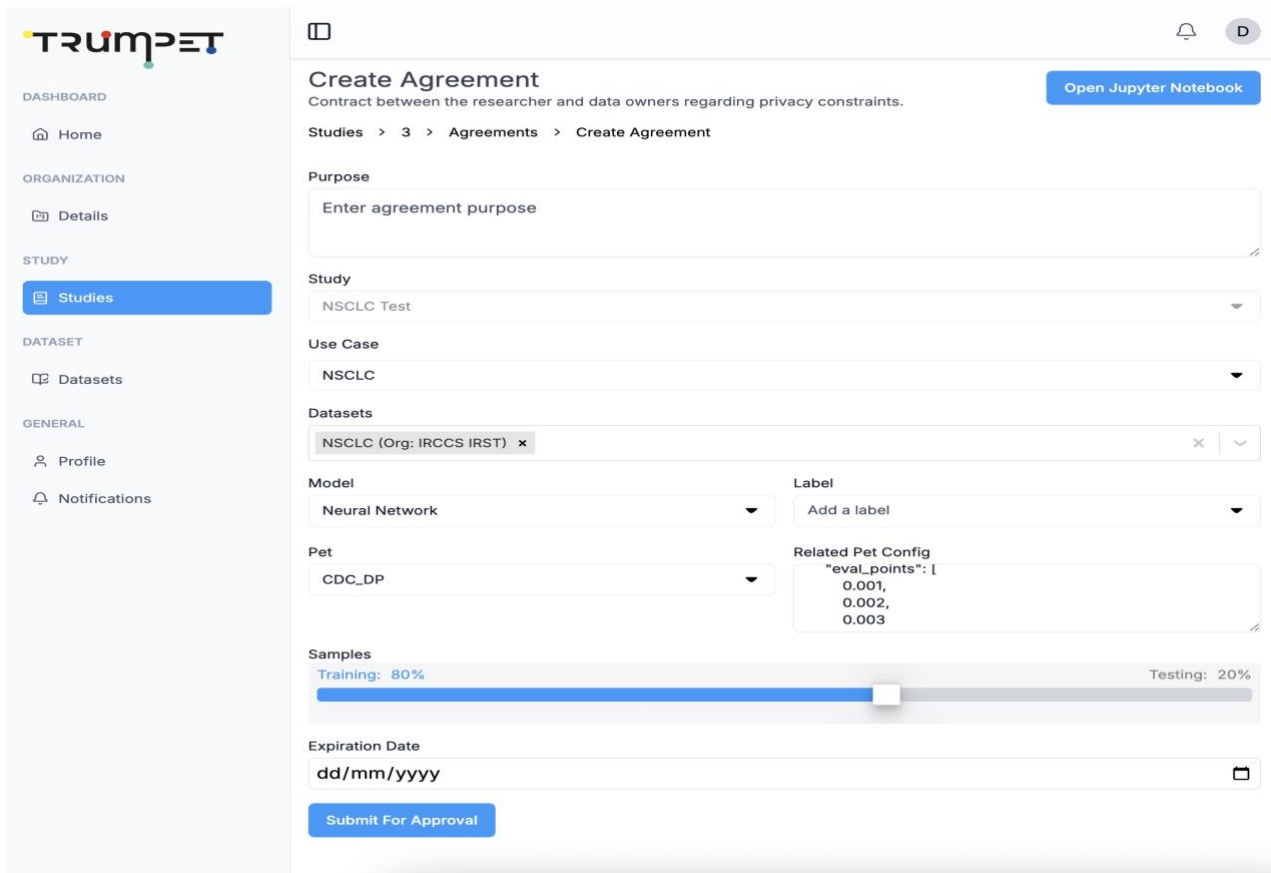
Figure 21: Researcher Flow - Cloud - View Dataset Statistics

Create a study: The researcher creates a study, selects one or more published datasets, and defines the task (features/labels). They can also set PET methods and personalize the PET parameters if necessary , privacy budget, use case, and sample size settings.



The screenshot shows the 'Create Study' form in the TRUMPET interface. The form is titled 'Create Study' and is described as 'A project on federated learning.' The breadcrumb navigation shows 'Studies > Create Study'. The form contains four text input fields: 'Name' (placeholder: 'Enter study name'), 'Description' (placeholder: 'Enter study description'), 'Purpose' (placeholder: 'Enter study purpose'), and 'Result' (placeholder: 'Enter study result'). A blue 'Submit' button is located at the bottom of the form. The left sidebar contains navigation links for 'DASHBOARD', 'ORGANIZATION', 'STUDY', 'DATASET', and 'GENERAL'. The 'STUDY' section is active, with 'Studies' highlighted. The top right corner shows a notification bell and a user profile icon labeled 'D'.

Figure 22: Researcher Flow - Cloud - Create Study



TRUMPET

DASHBOARD

- Home

ORGANIZATION

- Details

STUDY

- Studies**

DATASET

- Datasets

GENERAL

- Profile
- Notifications

Create Agreement

Contract between the researcher and data owners regarding privacy constraints.

Studies > 3 > Agreements > Create Agreement

[Open Jupyter Notebook](#)

Purpose

Enter agreement purpose

Study

NSCLC Test

Use Case

NSCLC

Datasets

NSCLC (Org: IRCCS IRST) x

Model

Neural Network

Label

Add a label

Pet

CDC_DP

Related Pet Config

```
"eval_points": [
  0.001,
  0.002,
  0.003
```

Samples

Training: 80% Testing: 20%

Expiration Date

dd/mm/yyyy

[Submit For Approval](#)

Figure 23: Researcher Flow - Cloud - Create Study Agreement

Await Data Owner approvals: The researcher is notified when Data Owners approve or request changes to the study agreements. Real-time notifications are delivered via WebSocket.

Submit training: The request goes to the FL Aggregator in the cloud when training is submitted. The aggregator notifies the participating Data Owner nodes, which trigger their local FL Core. Each node trains on its own data and returns privacy-preserving updates. The cloud listener records results and stores final outputs in the backend database (PostgreSQL).


Training Logs

View training logs below

training_notification	Oct 6, 2025 4:56:23 PM	{"training_status": "training_started"}
training_notification	Oct 6, 2025 4:56:24 PM	{"training_status": "round_started", "round": 1}
training_notification	Oct 6, 2025 4:56:24 PM	{"training_status": "training_started"}
training_notification	Oct 6, 2025 4:56:24 PM	{"training_status": "round_started", "round": 1}
training_notification	Oct 6, 2025 4:56:29 PM	{"round_completion_time": 5.152911424636841, "round": 1}
training_notification	Oct 6, 2025 4:56:30 PM	{"round_completion_time": 3.3854734897613525, "round": 1}
training_notification	Oct 6, 2025 4:56:35 PM	{"training_status": "round_started", "round": 2}
training_notification	Oct 6, 2025 4:56:35 PM	{"training_status": "round_started", "round": 2}
training_notification	Oct 6, 2025 4:56:38 PM	{"round_completion_time": 2.4429755210876465, "round": 2}
training_notification	Oct 6, 2025 4:56:38 PM	{"round_completion_time": 2.3932642936706543, "round": 2}
training_notification	Oct 6, 2025 4:56:43 PM	{"training_status": "round_started", "round": 3}
training_notification	Oct 6, 2025 4:56:43 PM	{"training_status": "round_started", "round": 3}
training_notification	Oct 6, 2025 4:56:46 PM	{"round_completion_time": 2.3881874084472656, "round": 3}
training_notification	Oct 6, 2025 4:56:46 PM	{"round_completion_time": 2.444448471069336, "round": 3}
training_notification	Oct 6, 2025 4:56:51 PM	{"training_status": "training_finished"}
training_notification	Oct 6, 2025 4:56:51 PM	{"training_status": "training_finished"}
model_metrics_published	Oct 6, 2025 4:56:52 PM	

Figure 24: Researcher Flow - Cloud - View Real Time Training Notifications

Monitor rounds & review results: The dashboard shows live study status, round progression, and final model artefacts and summaries. Raw data never leaves the sites—only privacy-preserving updates and aggregated metrics are shared.


🔔
D

DASHBOARD

Home

ORGANIZATION

Details

STUDY

Studies

DATASET

Datasets

GENERAL

Profile

Notifications

Agreement Result Details

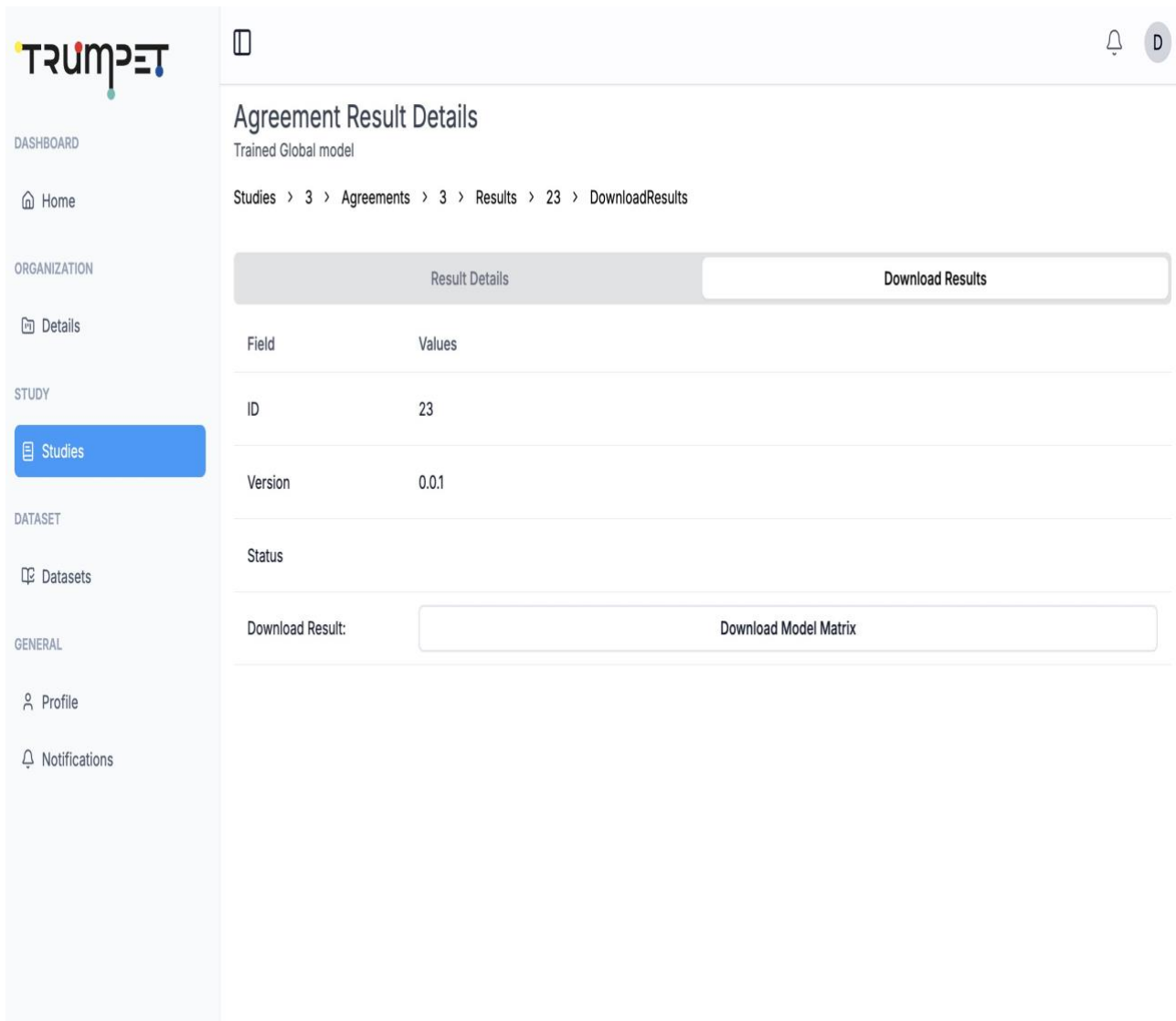
Trained Global model

Studies > 3 > Agreements > 3 > Results > 23 > ResultDetails

Result Details
Download Results

Field	Values	
ID	23	
Specifications	Mse	54.114
	Rmse	7.356
	Mae	5.743
	R2	0.175
	Mape percent	43.234
	Smape percent	35.712
	Explained variance	0.246
Study Agreement ID	3	
Created	Oct 24, 2025 6:03:37 PM	
Updated	Oct 24, 2025 6:03:37 PM	

Figure 25: Researcher Flow - Cloud - View Training Result



Agreement Result Details
Trained Global model

Studies > 3 > Agreements > 3 > Results > 23 > DownloadResults

Result Details | Download Results

Field	Values
ID	23
Version	0.01
Status	

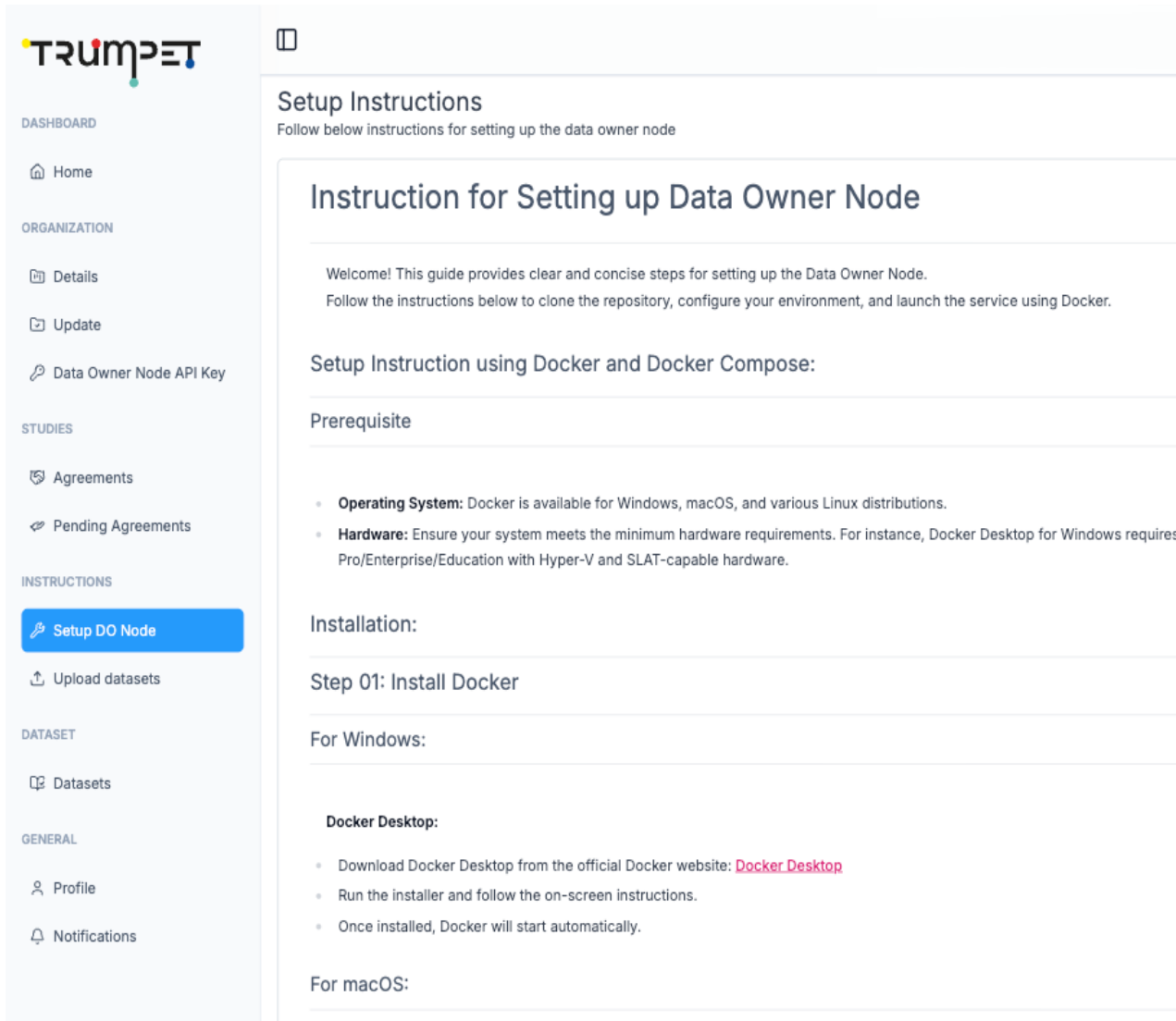
Download Result:

Figure 26: Researcher Flow - Cloud - Download Training Result

5.3.2 Data Owner flow (Local Node + Cloud Admin)

Goal: Publish safe-to-use datasets and control how they are used.

Register the organisation & set up the node: The Data Owner admin registers the organisation in the cloud and follows the Get Data Owner Node guide to install the on-prem node (including the reverse proxy to HAPI FHIR).



TRUMPET

DASHBOARD

- Home

ORGANIZATION

- Details
- Update
- Data Owner Node API Key

STUDIES

- Agreements
- Pending Agreements

INSTRUCTIONS

- Setup DO Node**
- Upload datasets

DATASET

- Datasets

GENERAL

- Profile
- Notifications

Setup Instructions

Follow below instructions for setting up the data owner node

Instruction for Setting up Data Owner Node

Welcome! This guide provides clear and concise steps for setting up the Data Owner Node. Follow the instructions below to clone the repository, configure your environment, and launch the service using Docker.

Setup Instruction using Docker and Docker Compose:

Prerequisite

- Operating System:** Docker is available for Windows, macOS, and various Linux distributions.
- Hardware:** Ensure your system meets the minimum hardware requirements. For instance, Docker Desktop for Windows requires Pro/Enterprise/Education with Hyper-V and SLAT-capable hardware.

Installation:

Step 01: Install Docker

For Windows:

Docker Desktop:

- Download Docker Desktop from the official Docker website: [Docker Desktop](#)
- Run the installer and follow the on-screen instructions.
- Once installed, Docker will start automatically.

For macOS:

Figure 27: Data Owner Flow - Cloud - Read Setup Instruction

Upload datasets (FHIR): The IT manager uploads anonymised or pseudonymized data as FHIR resources. The node generates local statistics and metadata according to the UI and use-case specifications.

Data Owner Upload Dataset Instructions

[View data owner upload dataset instructions](#)

Instruction for Uploading Datasets!

Step 1. Setup the Data Owner Node following the DON Setup Instructions

Step 2. Open the Dashboard UI

To open the dashboard UI, use SSH port forwarding to access the services running on the server. The following command will forward the necessary ports:

```
ssh \
-L 3000:127.0.0.1:3000 \
-L 9000:127.0.0.1:9000 \
-L 9001:127.0.0.1:9001 \
<your-username@your-server-ip>
```

Then open your web browser and navigate to `http://localhost:3000` to access the dashboard UI.

N.B: The URL `http://localhost:9000` will connect you to the backend service and the URL `http://localhost:9001` will connect you to the hapi-fhir service.

Step 3. Create an API Key

Login using your username and password and navigate to `http://localhost:3000/en/apikey` and generate an API key. Copy this key for using in the next steps.

Step 4. Create Questionnaire Response Resources

Create a `QuestionnaireResponse` resource for a `questionnaire` in the hapi-fhir database. Replace `your_api_key` and `questionnaire_response` with appropriate values. Example `curl` command:

```
curl --location 'http://localhost:9001/QuestionnaireResponse' \
--header 'X-API-Token: your_api_key' \
--header 'Content-Type: application/json' \
--data 'questionnaire_response'
```

Similarly, create all the `QuestionnaireResponse` resources.

Step 5. Create a Dataset

Navigate to `http://localhost:3000/en/datasets/create` and fill out the form with appropriate data to create a dataset. Provide the `questionnaire` url used in the previous step to create `QuestionnaireResponse` resources in the `Questionnaire` field.

Step 6. Generate Statistics for the Dataset

Navigate to `http://localhost:3000/en/datasets` and click the eye icon to view your dataset. Go to the `Statistics` tab and click the `Generate Statistics` button.

Step 7. Generate Trumpet Cloud API Key

Once the dataset is created. Navigate to `https://trumpet.technovativesolutions.co.uk`, login to your account and navigate to `https://trumpet.technovativesolutions.co.uk/en/apikey`. Then generate an API Key by setting a suitable expiry. Copy this API Key.

Step 8. Paste Trumpet Cloud API Key in Data Owner Node

Navigate to `http://localhost:3000/en/trumpet-cloud-apikey` and paste the API Key generated in the previous step.

Step 9. Publish Dataset

Navigate to the dataset list page at `http://localhost:3000/en/datasets` and view the dataset you want to publish. If you have generated its statistics previously, you should see a `Publish Dataset` button. Click that button and your dataset will be published to the trumpet cloud.

Figure 28: Data Owner Flow - Cloud - Read Dataset Upload Instruction

TRUMPET

DASHBOARD

- Home

DATASET

- API Key
- Dataset**

ORGANIZATION

- TC API Key

GENERAL

- Profile

INSTRUCTION

- Setup DO
- Upload Dataset

Create Dataset

Create Dataset using the form below

Datasets > Create Dataset

Title *

Enter the dataset title

Questionnaire * ⓘ

Enter the questionnaire url

Use Case *

Select Usecase

About

Enter details about your dataset

Temporal Coverage Start ⓘ

dd/mm/yyyy

Temporal Coverage End ⓘ

dd/mm/yyyy

Geospatial Coverage ⓘ

Enter Geospatial coverage

DOI Citation ⓘ

Figure 29: Data Owner Flow - on Premise Node - Create Dataset Resource

Publish the dataset to the cloud: Publish the dataset once metadata and stats are ready. Only descriptors are advertised in the cloud; the source records stay on-prem.

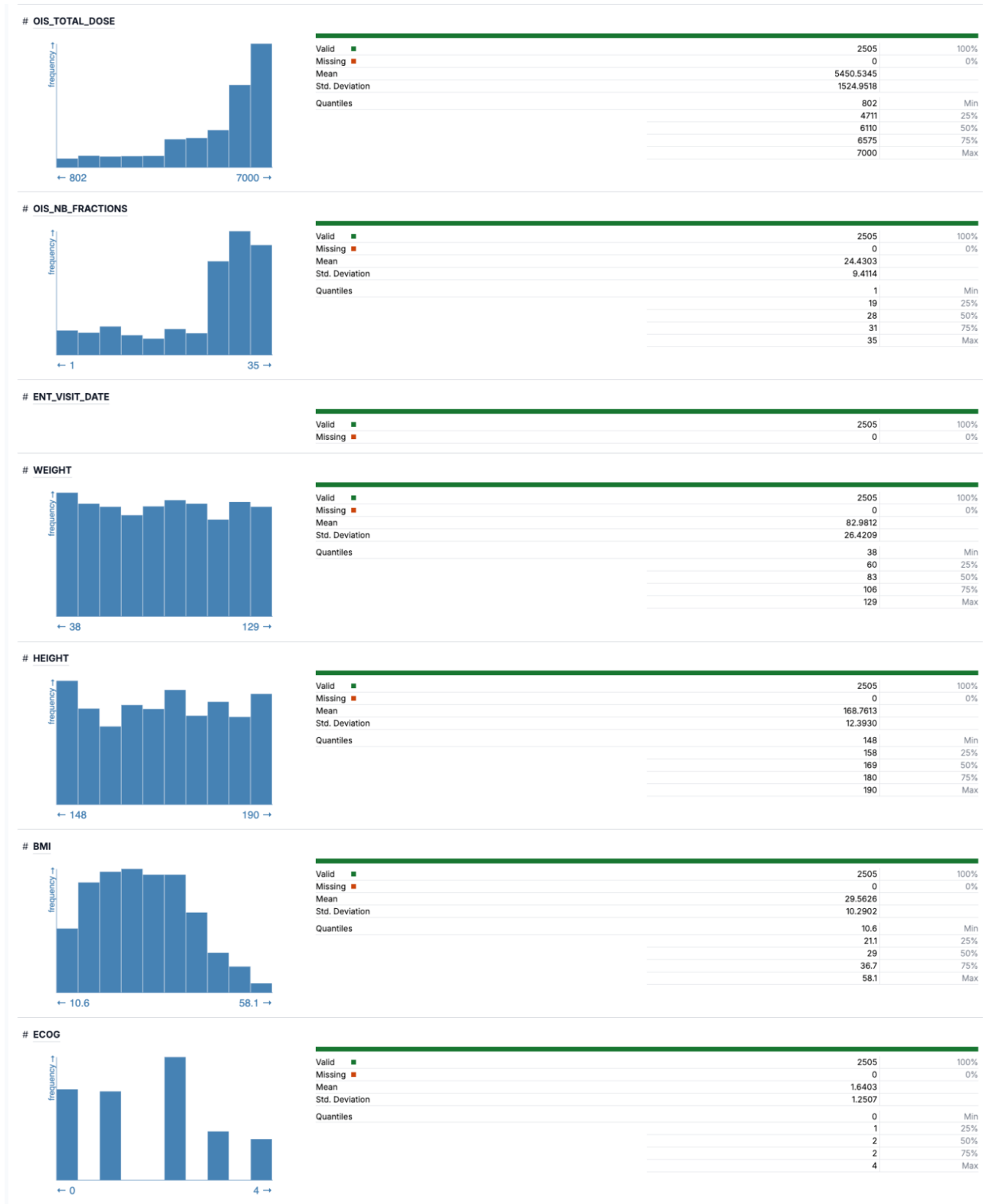


Figure 30: Data Owner Flow - on Premise Node - Generate Dataset Statistics

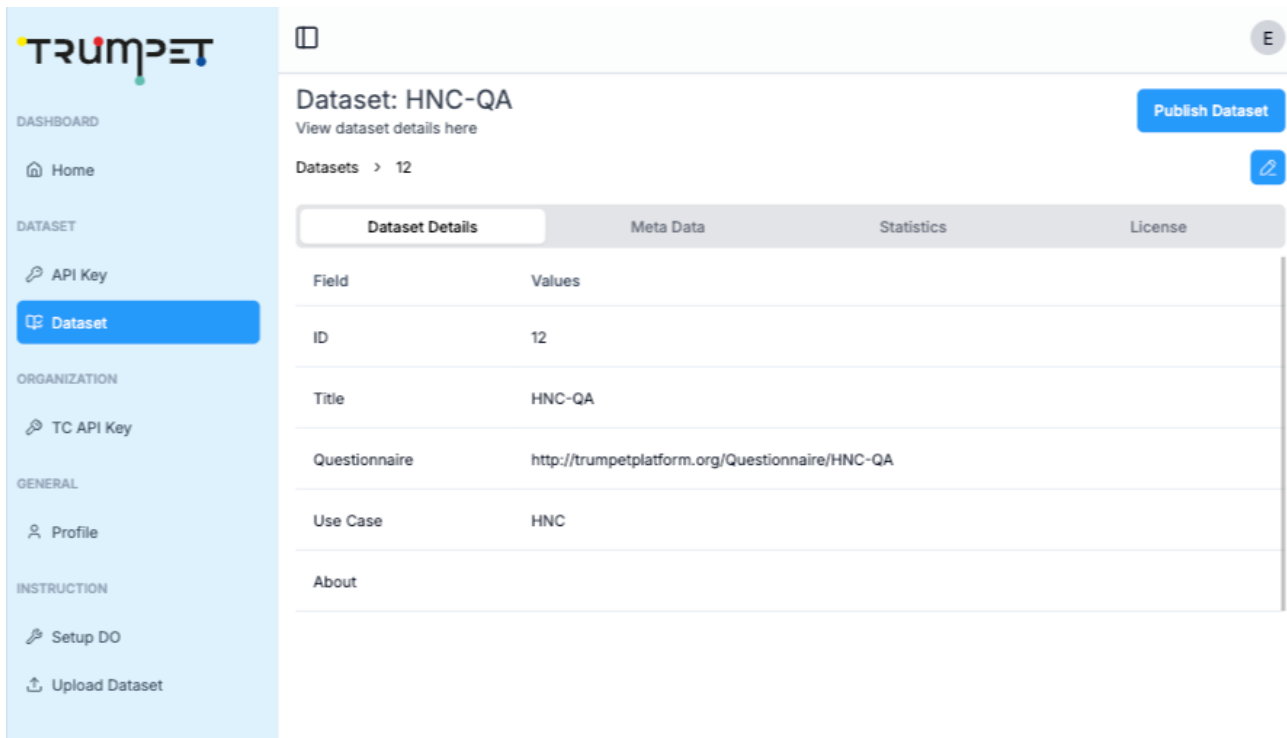


Figure 31: Data Owner Flow - on Premise Node - Publish Dataset to Cloud

Approve or reject study requests: From the Cloud Dashboard, the Data Owner reviews incoming study agreement requests and approves, requests changes, or rejects.

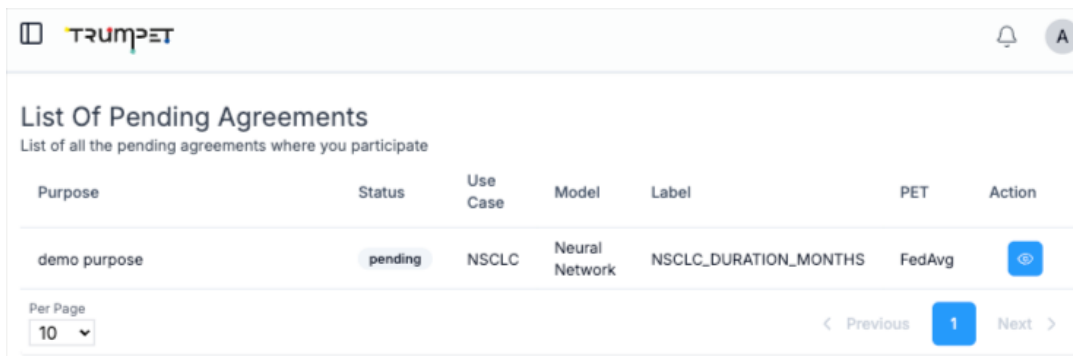


Figure 32: Data Owner Flow - Cloud - View Pending Training Request

Gate incoming studies: When approved, the node enforces PET profiles, use-case scope, and purpose labels. It may still accept or block execution based on policy, and it logs every decision for audit.

TRUMPET
🔔 A

Agreement: Demo Purpose

Contract between the researcher and data owners regarding privacy constraints

Edit Agreement

✓
✗

Study Agreement Details

Notifications

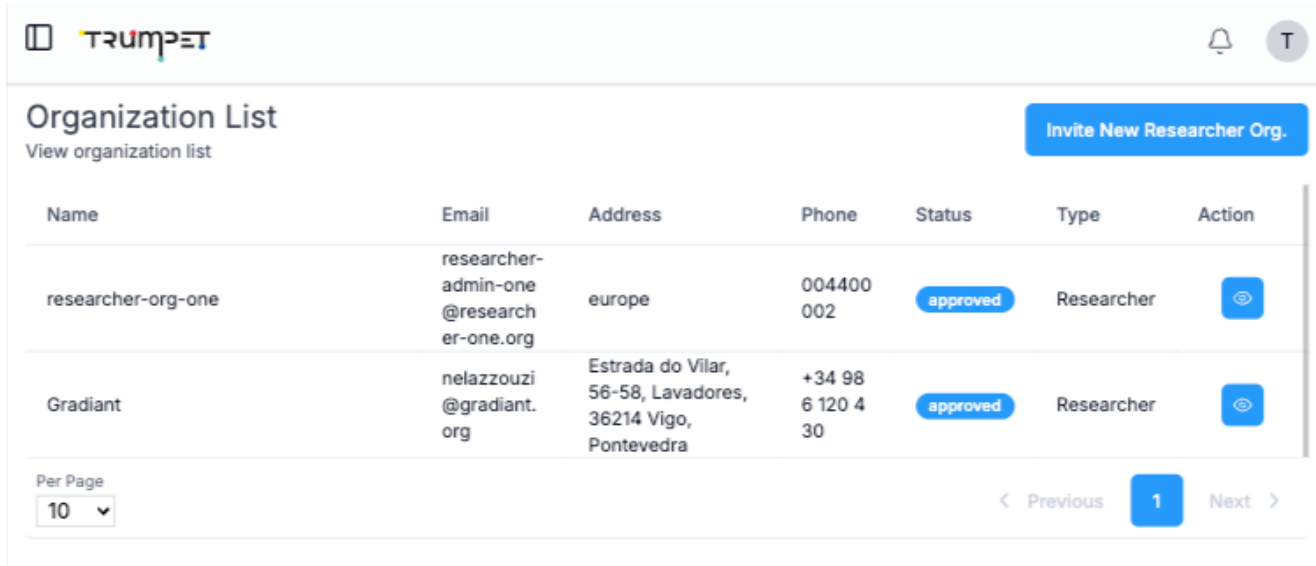
Results

Field	Values				
ID	4				
Status	pending				
Purpose	demo purpose				
Study ID	1				
Use Case	NSCLC				
Datasets	<table border="1" style="width: 100%; border-collapse: collapse; font-size: x-small;"> <thead> <tr> <th style="width: 50%;">Dataset Title</th> <th style="width: 50%;">Organization Name</th> </tr> </thead> <tbody> <tr> <td>Demo dataset for NSCLC_IRST</td> <td>do-org-one</td> </tr> </tbody> </table>	Dataset Title	Organization Name	Demo dataset for NSCLC_IRST	do-org-one
Dataset Title	Organization Name				
Demo dataset for NSCLC_IRST	do-org-one				
Model	Neural Network				
Label	NSCLC_DURATION_MONTHS				
Pet	FedAvg				
Pet Config	<code>{}</code>				
Samples	80				
Expiration Date	Oct 11, 2025				

Figure 33: Data Owner Flow - Cloud - Approve or Reject Training Request

5.3.3 Governance/Admin flow (Cloud)

View organizations: Governance Admin sees the list of registered and pending organizations.



The screenshot shows the 'Organization List' page in the TRUMPET admin interface. It features a table with columns for Name, Email, Address, Phone, Status, Type, and Action. Two organizations are listed: 'researcher-org-one' and 'Gradiant'. Both have a status of 'approved' and are of type 'Researcher'. A 'Per Page' dropdown is set to 10, and the page navigation shows '1' of 1 pages.

Name	Email	Address	Phone	Status	Type	Action
researcher-org-one	researcher-admin-one@researcher-one.org	europa	004400002	approved	Researcher	
Gradiant	nelazzouzi@gradiant.org	Estrada do Vilar, 56-58, Lavadores, 36214 Vigo, Pontevedra	+34 986 120 430	approved	Researcher	

Figure 34: Governance Admin Flow - Cloud - View Organisation List

Send invitations: Governance Admin invites an organization by entering its email in a pop-up.

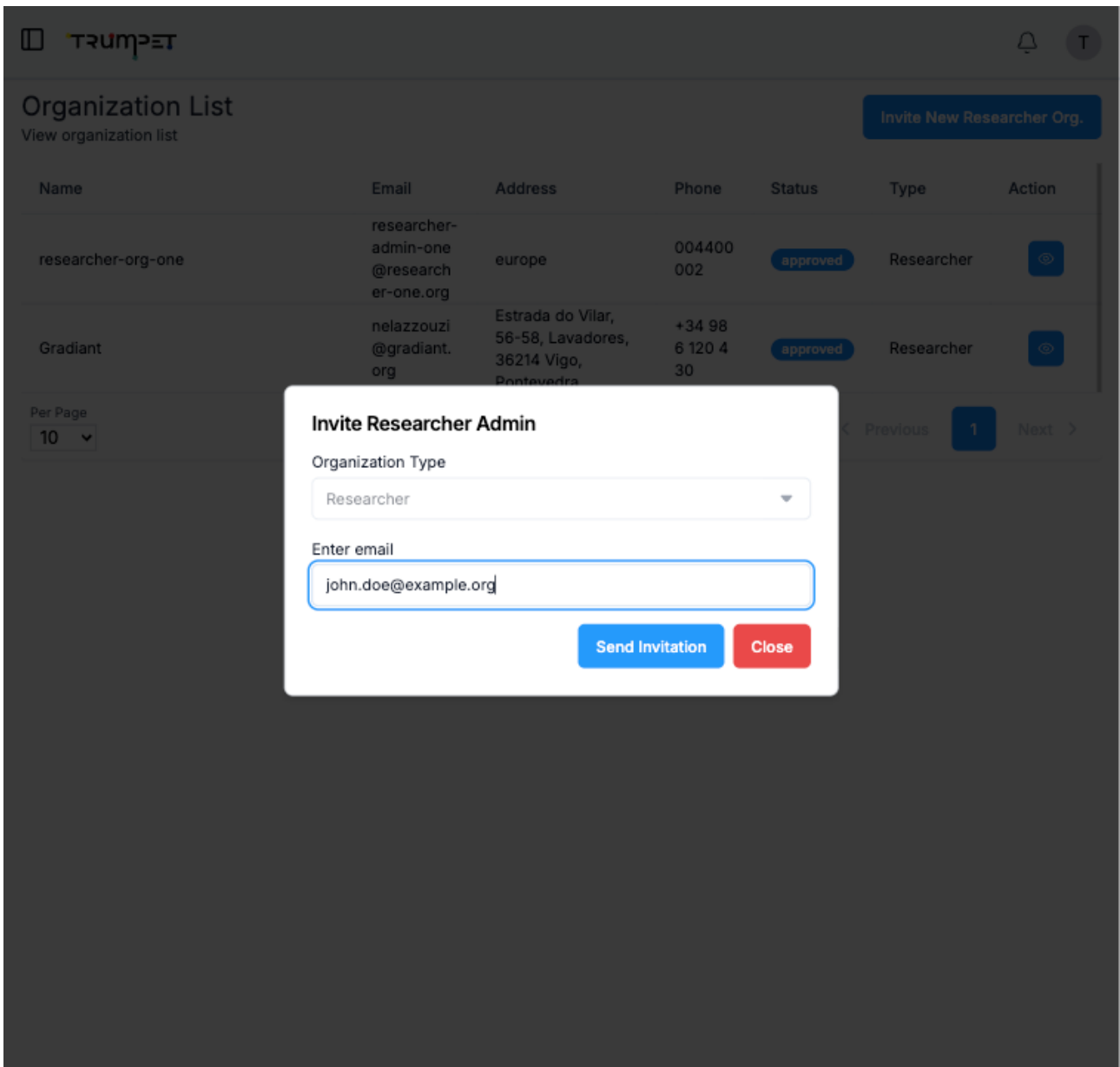


Figure 35: Governance Admin Flow - Cloud - Invite Organisation

Verify email: The Organization Admin receives the email and clicks the verification link.

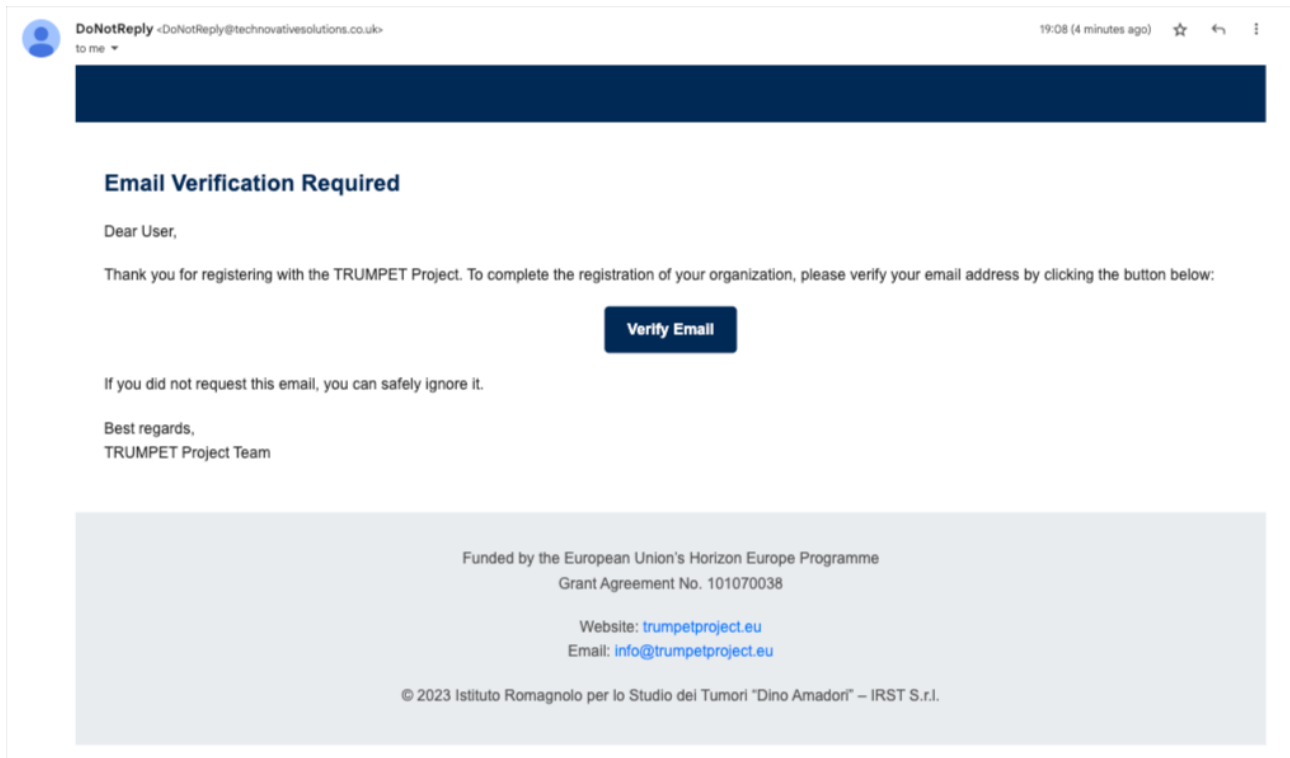


Figure 36: Governance Admin Flow - Email Client - Email Invitation is Sent to the Organisation

Register: The Organization Admin completes and submits the registration form.

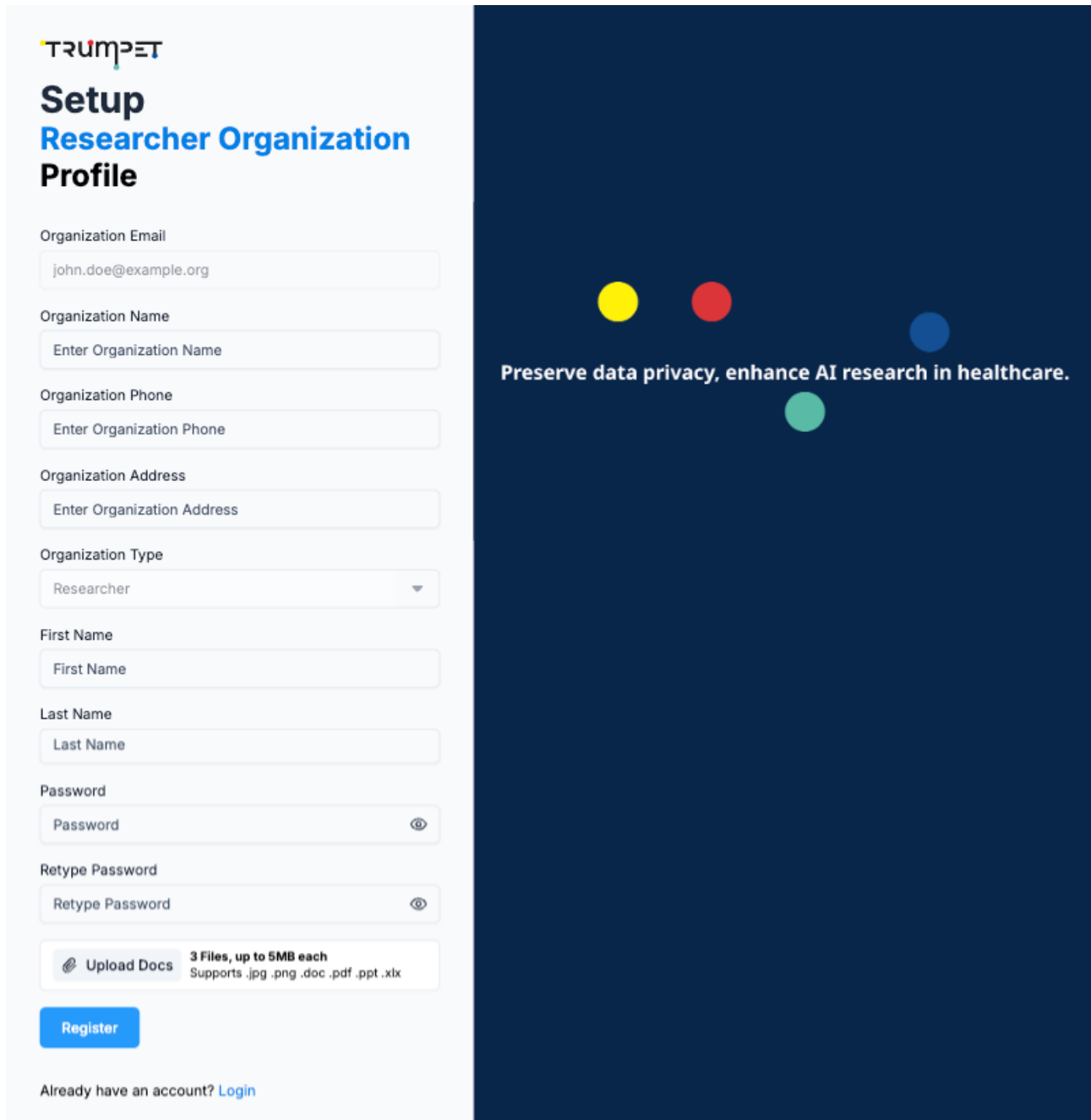
The image shows a registration form for a researcher organization. The form is titled "Setup Researcher Organization Profile" and includes fields for Organization Email, Organization Name, Organization Phone, Organization Address, Organization Type (a dropdown menu set to "Researcher"), First Name, Last Name, Password, and Retype Password. There is also an "Upload Docs" section with a file upload icon and text indicating "3 Files, up to 5MB each" and supported file formats (.jpg, .png, .doc, .pdf, .ppt, .xlsx). A blue "Register" button is at the bottom of the form. Below the button, there is a link for "Already have an account? Login". To the right of the form is a dark blue banner with the text "Preserve data privacy, enhance AI research in healthcare." and four colored circles (yellow, red, blue, green) arranged in a pattern.

Figure 37: Governance Admin Flow - Cloud - Organisation Admin Fills up Registration Form

Review pending registrations: Governance Admin monitors the queue of pending organizations.

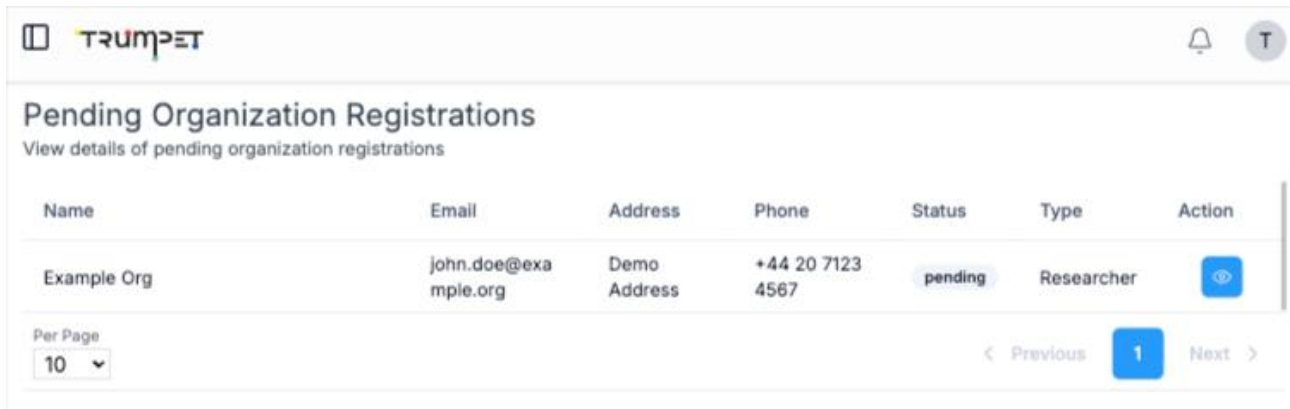


Figure 38: Governance Admin Flow - Cloud - View Pending Organisation Registrations

Open details: Governance Admin reviews the details of a specific pending organization.

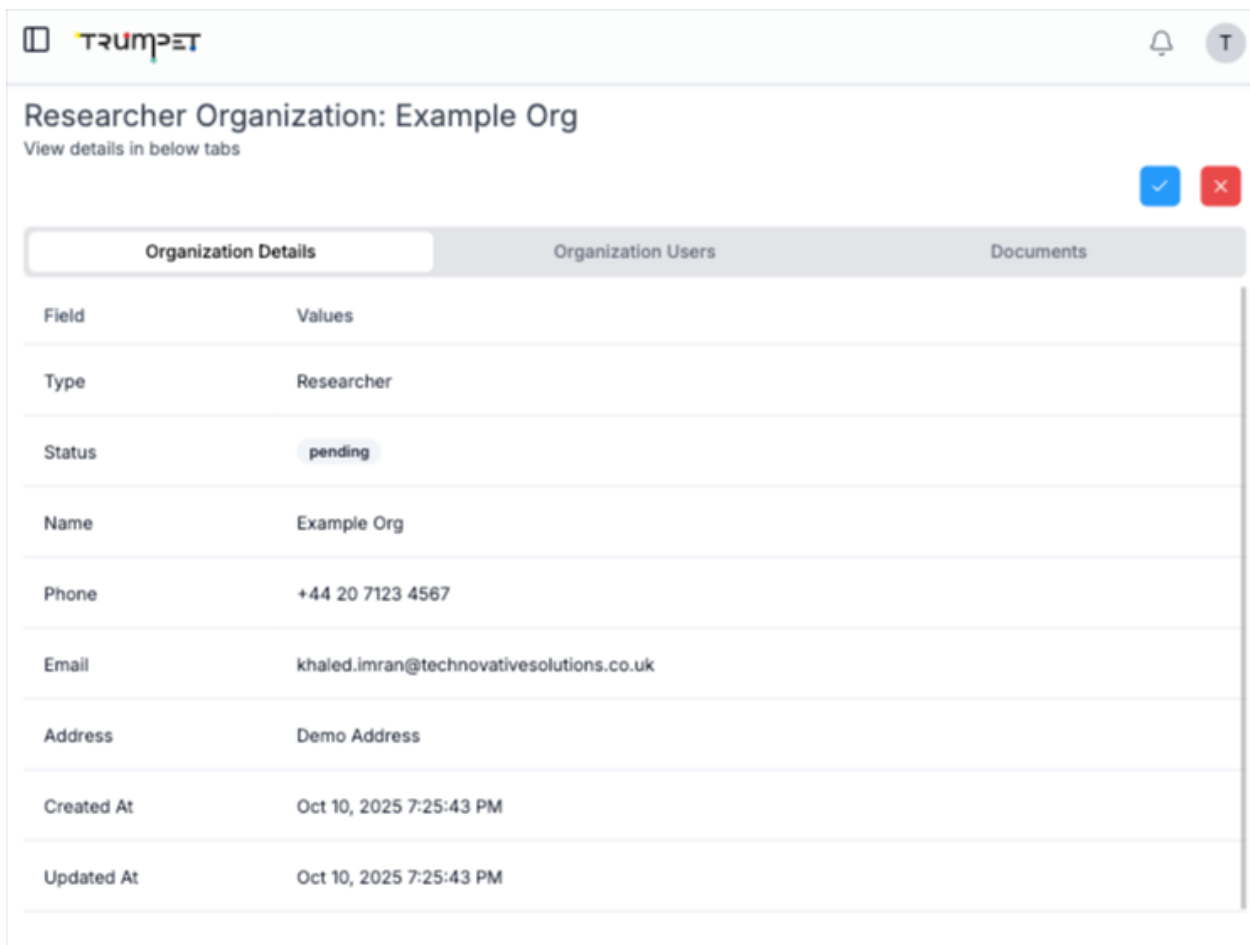


Figure 39: Governance Admin Flow - Cloud - View Pending Organisation Details

Approve or reject: Governance Admin accepts or declines the registration; the organization is notified and can log in if approved.

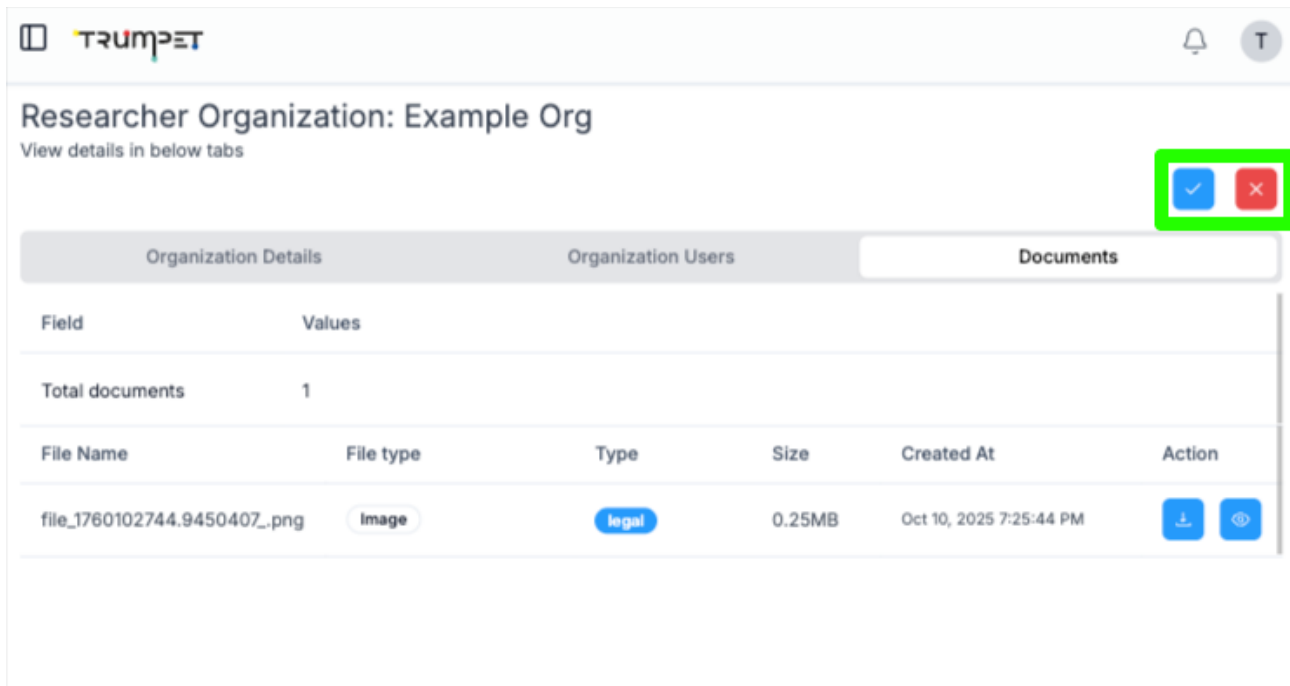


Figure 40: Governance Admin Flow - Cloud - View Pending Organisation Docs and Approve or Reject

5.3.4 Federated training round-trip (system view)

Researcher starts training

The researcher clicks “Train” in the Cloud Dashboard to request a federated job.

Cloud fans out the request

The Cloud Dashboard creates the FL job with the Cloud FL Aggregator and, at the same time, notifies the Data Owner Node (DON) Dashboard about the study request (model, sample size, use case, labels).

Data Owner validates the request

The DON Dashboard checks policy, PET profile, sample size, and label requirements. If anything is missing or non-compliant, it rejects the request with a clear reason.

Prepare data locally (no data leaves)



TRUStworthy Multi-site Privacy Enhancing Technologies

If valid, the DON reshapes local HAPI FHIR data into an ML-ready view (only inside the hospital).

Start local training

The DON Dashboard triggers the DON FL Core to begin training on the local data.

Federated rounds (privacy-preserving)

- The Cloud FL Aggregator sends the current global model and plan to the DON FL Core.
- The FL Core trains on its own data in-memory and produces a model update (gradients/weights).
- Only the update is sent back to the Aggregator—never raw patient data.

(OPTIONAL) : If a Pet is chosen, it is applied during the training or the aggregation part

Secure aggregation

The Aggregator combines updates from all participating sites to produce the next global model. This loop may repeat for several rounds.

Publish results

When training completes, the Aggregator posts the final model and metrics to the Cloud Listener.

Real-time delivery to the dashboard

The Cloud Listener pushes results to the Cloud Dashboard via WebSocket, and the researcher sees the final model and artifacts immediately.

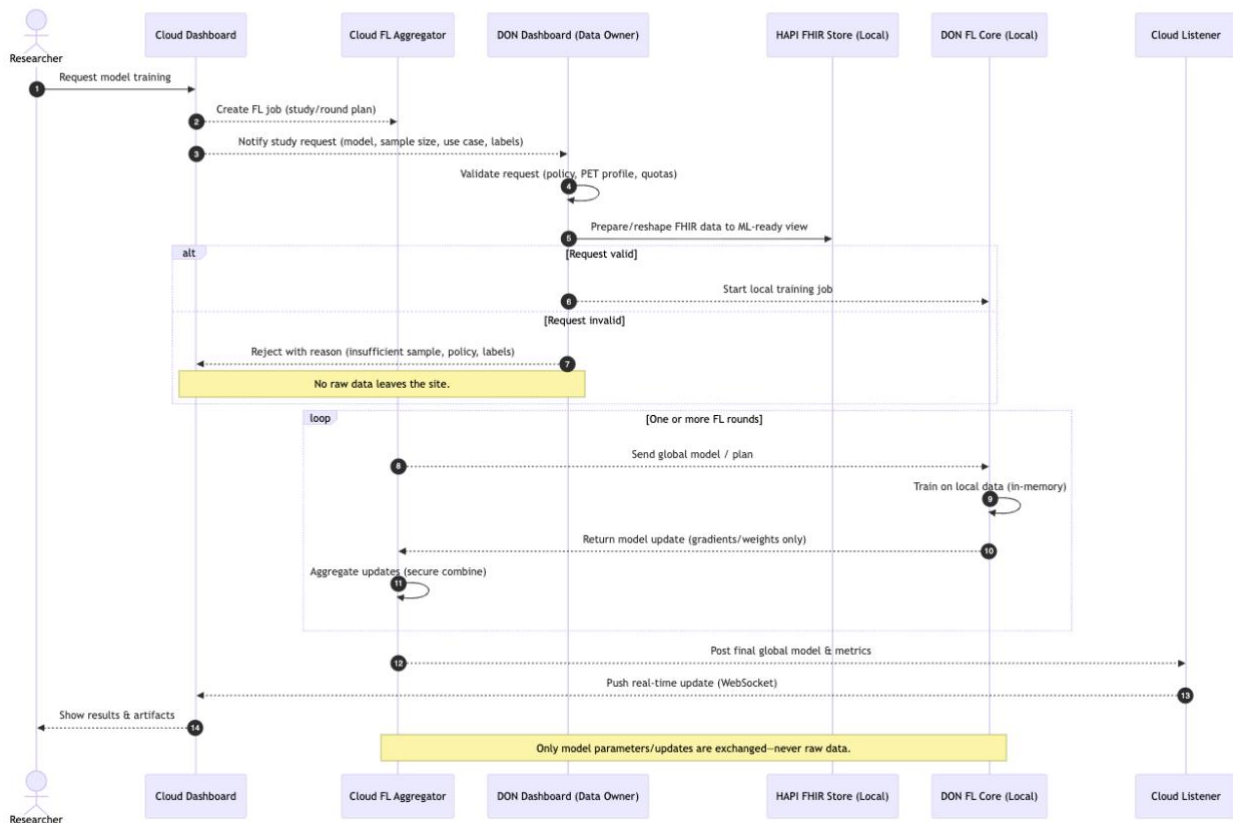


Figure 41: FL System Flow

5.4 API Design & Versioning

This section explains how TRUMPET’s APIs are designed, documented, secured, and versioned so that cloud services, Data Owner nodes, and FL Core can evolve without breaking running studies.

5.4.1 API styles and layering

We keep a small, stable set of public APIs and a larger set of internal service interfaces. The goal is a clean edge for partners and dashboards, with flexible internals for change.

Public APIs (stable)

- REST endpoints for dashboards and partners — CRUD for Users, Organisations, Datasets, Studies, Files, and more. Follows Google-style JSON, consistent status codes, standard error envelopes, and is documented in OpenAPI/Swagger.

Internal & system APIs

- **Action/command endpoints** — Non-resource-based APIs for operations that don’t fit pure CRUD (e.g., “start-training”, “send-invitation”).

- **Listener APIs (training webhooks)** — Authenticated endpoints that receive round updates, metrics, and completion messages from the FL pipeline.
- **Data Owner Node APIs** — Cloud↔node control plane for study invitations, approvals, and key management.
- **HAPI Proxy APIs** — Authenticated upload/retrieval of FHIR resources to the on-prem HAPI FHIR server; enforces local policy and audit.

5.4.2 Resource model and response rules

Resources use predictable, pluralised paths (e.g., /api/users, /api/datasets/:id, /api/studies/:id/datasets). Standard methods (GET/POST/PUT/PATCH/DELETE), pagination, filtering, and 4xx/5xx semantics are uniform across resources. Example sets are already defined for Roles, Users, Organisations, Datasets, Studies, and Models.

5.4.3 Security & Access

- **Token-based auth with RBAC:** Researchers, Data Owners, and Governance admins authenticate with tokens; their roles determine their abilities. Tokens can be rotated or revoked centrally. Where appropriate, node-level actions may also require SSH key verification.
- **Privacy gates at the Data Owner node:** The node is not directly reachable from the Internet. Reverse proxy rules (e.g., Traefik) expose only the required, authenticated endpoints—typically via explicit port forwarding. All other traffic is blocked.
- **Secure cloud↔node calls:** Every cloud–node request is authenticated with signed tokens and audited end-to-end. Node-to-node communication (when applicable) is also token-secured.

5.4.4 Example version map

Surface	Style	Current	Notes
Cloud Public API	REST	v1	CRUD for users/orgs/datasets/studies/models; additive extensions only.
FL Control API	REST	v1	Training orchestration and model lifecycle between Cloud and FL adapter.
Cloud↔Data Owner	REST	v1	Topics setup training.

Table 14: API Versioning

5.5 Database Design & Multi-Site Considerations

This section describes how we store, protect, and synchronize data across TRUMPET Cloud and multiple Data Owner sites. The goal is simple: keep patient data on-premise, expose only safe metadata to the cloud, and make the system resilient and easy to operate across many hospitals.

5.5.1 Logical split: Cloud vs Data Owner

- **TRUMPET Cloud (metadata & coordination):** Stores researcher identities, organisations, published dataset descriptors, studies, and the state of federated learning jobs. No raw clinical data is stored here.
- **Data Owner Node (clinical data & computation):** Runs the local HAPI FHIR server behind a hardened reverse proxy, executes on-demand analytics, and trains local model updates. Patient data never leaves the hospital. Also includes a local relational database for non-FHIR operational data (e.g., users, roles, dataset records).

5.5.2 Cloud relational model

We use PostgreSQL as the primary relational store for cloud services. The schema centres on Users, Organisations, Datasets, Studies, Roles, and collaboration links, reflecting how researchers discover datasets, form studies, and run training.

A high-level ER view shows: Organisations ↔ Users (role-scoped), Datasets ↔ Organisations (published descriptors), StudyAgreements ↔ Datasets (selection and query context), plus audit entities.

5.5.3 Data Owner persistence

Each Data Owner installs a HAPI FHIR instance (Questionnaire / QuestionnaireResponse today) shielded by a HAPI Proxy for authentication and network isolation. Uploads from the hospital IT system are intercepted, authenticated, and cached (e.g., via Redis) before persisting in the FHIR store. Analytics and model-training packages run locally and only emit privacy-safe statistics or encrypted/noisy model matrices to the cloud.

During onboarding, partners size and deploy local FHIR repositories and indexes following the common FHIR model; HAPI FHIR is the backbone for these private repositories.

5.5.4 How we store and organise datasets

- We store clinical records in the HAPI FHIR server as QuestionnaireResponse resources.
- Each QuestionnaireResponse points to a specific Questionnaire (its canonical URL/ID).



TRUStworthy Multi-site Privacy Enhancing Technologies

- In the Data Owner relational database (PostgreSQL), each dataset record references the Questionnaire it belongs to (and optional tags/labels).
- This mapping links many QuestionnaireResponse resources to one dataset, keeping datasets clearly separated by their Questionnaire (and tags if used).
- Result: you can manage multiple datasets in the same storage—raw responses live in FHIR, while dataset definitions and metadata live in PostgreSQL.

6 Privacy Measurement & Validation Hooks

This section details how privacy is measured within the TRUMPET project, highlighting the integration of third-party libraries in the TRUMPET Cloud and the Data Owner Node. In essence, we view the privacy tools developed in WP3 as modular capabilities that operate behind well-defined interfaces. These capabilities are activated precisely when risks arise, specifically during the study design, approval, execution, and reporting phases.

We have included a diagram in the next section to illustrate this concept. The diagram visually represents the boundary and call points necessary for study agreements and training processes. This diagram underscores the architecture and workflow of privacy measurement in TRUMPET, ensuring our core code interacts with abstractions rather than tying it to specific implementations.

6.1 FLCore Integration and Interfacing

Adapter first, platform second

TRUMPET strictly consumes the **FLCore** (the “FLCore Server”) through an **Adapter**. We treat FLCore as a third-party dependency—useful, but replaceable—so our business logic is coded against an abstract FLCore boundary (create study, load data, push global model, pull round updates, etc.). This is explicit in our design notes and dependencies: the FL Core is integrated via an adapter pattern in WP4. It is not a hard build-time dependency of the existing platform.

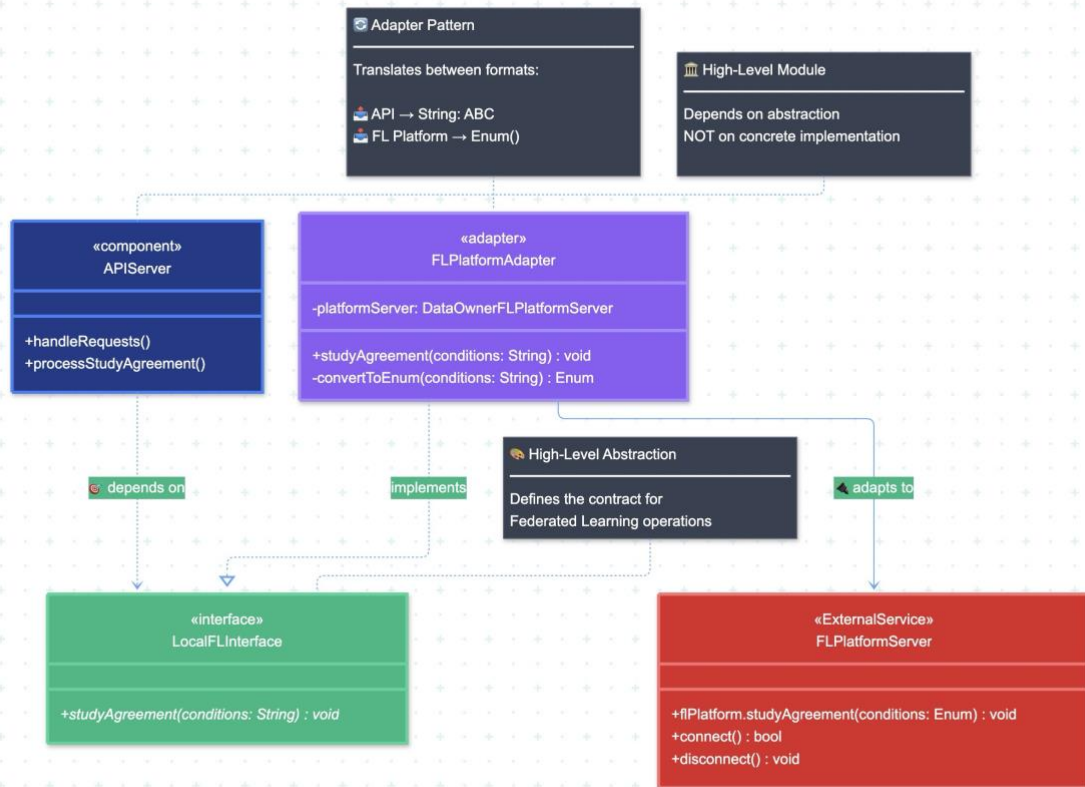


Figure 42: Adapters to implement partners' services

In the provided sketch, the adapter performs essential translations between human-readable study constraints and the FLCore’s canonical enums and payloads.

For example, the call `local_interface.study_agreement(conditions: "ABC")` translates to `fl_platform.study_agreement(conditions: Enum())`. This process ensures that data from the Cloud/API Server is accurately converted for use within FLCore while maintaining the ability to revert for audit purposes.

What stays internal vs. what’s exposed

Internal only → FLCore management calls (*create FL study, register clients, start rounds, load local data*) are not Internet-facing. They are invoked from the API Server toward the adapter (see the “API Server ↔ FL Platform Adapter” box in the diagram) and from the Data Owner Nodes toward their local FL server.

Exposed API → The only Internet-facing surface is the Communication API on port 8081. Our API Gateway and Authentication service fronted it with a token validation. Tokens can

be revoked centrally, and **CAC (Centralised Access Control)** enforces CORS and versioning.

Where privacy hooks meet FL

Pre-round: Before the adapter opens a training round, it queries *PETService* to validate that the planned aggregation method + DP/HE/SMPC settings keep the study within budget. *(It uses the PET/FL Core modules that wrap—not bind to—the platform.)*

Per round: After each client update arrives at the FLCore, the adapter streams round metadata to the Aggregator Service to update the metric and record spending.

6.2 Test Data & Attack Simulations

To approximate the potential privacy leakage during federated training, the FL Core includes a module that simulates inference attacks by hypothetical adversaries within the federation—either from the perspective of the Aggregator or a Data Owner.

Evaluating the effectiveness of these attacks in terms of data re-identification requires access to the real target data to measure performance accurately. Since the actual data must remain within the Data Owners' domain, one option would be to use a synthetic dataset that can be shared with the simulated attacker for testing. However, synthetic data may not accurately represent the real data and could mask part of the privacy leakage.

For this reason, we adopt an alternative approach that utilizes the victim's real data without transferring it outside their domain. When specified in the study agreement, we simulate an adversarial node—either the Aggregator or a Data Owner—that collects shared model weights during federated training to build an attack model. However, this attacker never tests this attack model using real data.

Instead, after the agreed number of FL training rounds, the attack model is shared with the victim, who then evaluates its effectiveness using their real data. This method enables a practical approximation of privacy leakage while preserving data privacy.

6.3 Result Capture & Reporting

Once the simulated attack is tested within the victim's domain using their own data, a set of performance metrics is computed to assess how successful the attack was. These metrics are accessible only to the victim, ensuring that sensitive information remains private.

The evaluation includes standard classification metrics for membership inference:

- **Accuracy:** The proportion of correctly predicted instances out of all instances.

- **Precision:** The fraction of correctly identified instances among all cases predicted as part of the target data, reflecting the attack's reliability in positive predictions.
- **Recall:** The fraction of actual target instances that were correctly identified, indicating the attack's ability to capture true positives.
- **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure when both metrics are important.
- **AUC (Area Under the Curve):** Evaluates the attack's performance across different classification thresholds, summarizing its ability to distinguish between members and non-members of the target dataset.

Together, these metrics provide a comprehensive view off the attack's effectiveness in re-identifying sensitive data.

7 Deployment & Environments

This section standardises how TRUMPET is deployed across cloud and hospital premises, how we package and configure components, how changes progress through environments, and how organisations are granted access.

7.1 Topologies (Cloud / On-Premises)

7.1.1 Cloud (TRUMPET Cloud)

Purpose— Public-facing dashboards and APIs for governance, researcher orgs, and data owners.

Entry point— A CAC fronts cloud microservices, handling routing, auth, observability, CORS(Cross-Origin Resource Sharing) and API versioning. It also brokers requests that may fan out to multiple backends.

Core cloud services:

- **Authentication Service** — account lifecycle, token issuance/revocation, role-based access, DO registration/identity, Researcher registration/identity.
- **Advertised Dataset Service** — catalogue of datasets published by DOs.
- **Model Training & Federated Model Management** — orchestration endpoints for study execution and aggregation of federated updates.
- **Analytics Aggregator** — secure statistics for dataset assessment.
- **Communication**— Cloud↔DO interaction is API—and message—driven; asynchronous websockets are used to decouple study/job distribution and results collection.

7.1.2 Data Owner Node (On-Premises)

Purpose— A sealed node inside the hospital network running the DO Dashboard, services and proxies. Patient data remains on-premise; only privacy-constrained outputs leave the site.

Main components:

- **DO dashboard + Backend** — user/role management, privacy controls, monitoring, study handling.
- **HAPI-FHIR Proxy** — a reverse proxy that mediates access to the local FHIR server and enforces token headers.
- **DO FLCore** — local study execution/training service started via Docker Compose.
- **Reverse Proxy (Traefik)** — TLS termination with automated certificate issuance/renewal and selective API exposure (“API hiding”) for DO services.

Operator access pattern— During bring-up, operators typically reach the dashboard via **SSH port-forwarding** on **3000/9000/9001** before exposing endpoints through the **reverse proxy**.

7.1.3 End-to-End Flow

Researchers operate solely through Cloud UIs/APIs. The cloud distributes FL jobs; DO nodes pull/execute them under PET constraints and post back updates. The gateway authorises and meters every cross-boundary call.

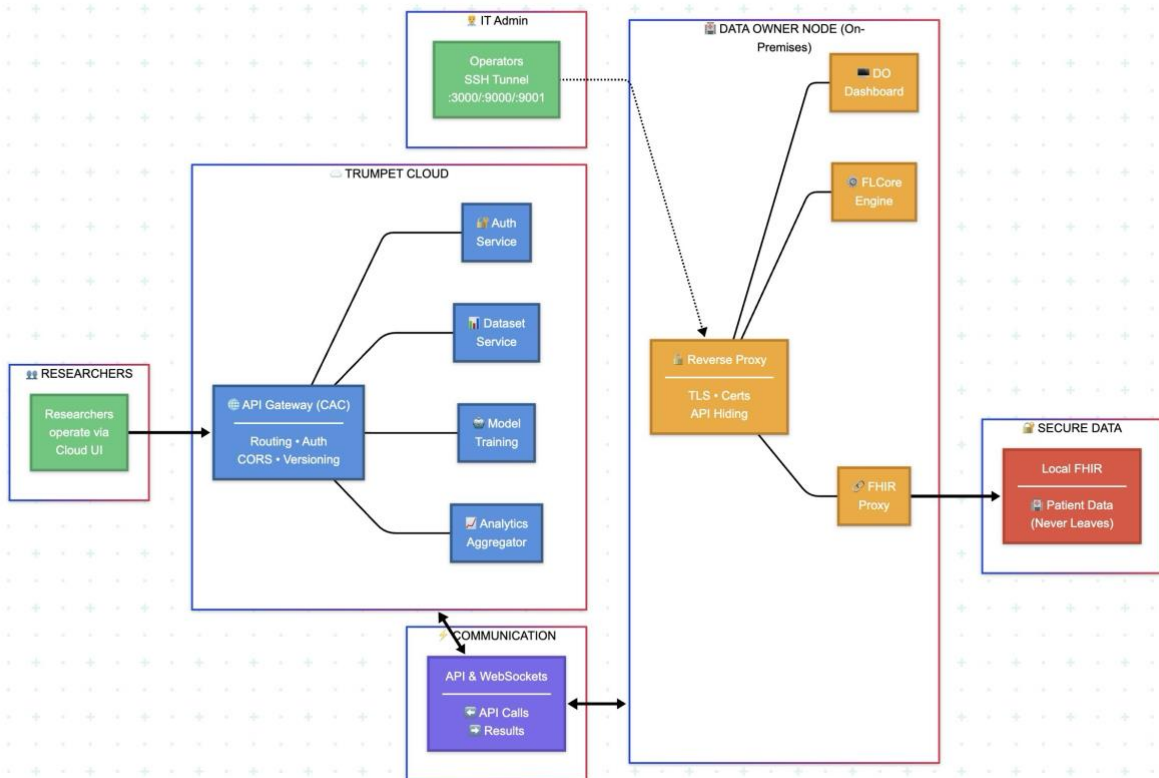


Figure 43: Deployment Architecture - Cloud and On-Premises Topologies

7.2 Packaging, Configuration, Secrets — with Deployment Guides

This section documents how the **Data Owner Node** and **TRUMPET Cloud** are packaged, configured, and deployed. It also details how secrets are handled and how the reverse proxy (Traefik) is used at the Data Owner edge to automate TLS and restrict exposed APIs.

7.2.1 Data Owner Node (On-Prem)

Packaging:

- **Runtime:** Docker + Docker Compose.

- **Topology:** All DO services run on an isolated Docker network; only the reverse proxy (Traefik) publishes ports externally.
- **Primary containers:**
 - DO Dashboard (frontend)
 - DO Backend/API
 - HAPI-FHIR Proxy (fronts the hospital FHIR server)
 - DO FLCore (executes local training)
 - Postgres, Redis (stateful services; volumes mapped on host)
 - Traefik (TLS termination + routing for approved endpoints only)

Configurations:

All settings are provided via a single .env file stored alongside the Compose file. Variables are as follows:

- **Django / Backend:** SECRET_KEY, ALLOWED_HOSTS, DEBUG, BACKEND_SERVICE_BASE_URL
- **Database:** DB_HOST, DB_PORT, DB_NAME, DB_USER, DB_PASSWORD
- **Cache/Queue:** REDIS_HOST, REDIS_PORT
- **FHIR:** HAPI_BASE_URL (internal URL to hospital FHIR or a proxy), HAPI_PROXY_BASE_URL (public route via Traefik)
- **Federated Learning:** DO_FL_CORE_BASE_URL (internal), allow-lists for outbound coordination (ALLOWED_COORDINATORS, ALLOWED_WEBHOOKS)
- **Frontend:** NEXT_PUBLIC_* variables for browser clients (public base API, HAPI proxy URL, etc.)
- **Cloud reference:** CLOUD_BASE_URL (used for registration and callback targets)

Network discipline: Internal container-to-container URLs must use service names on the Docker network (e.g., http://backend:9000), while anything exposed to users or the Cloud is routed through Traefik.

Secrets:

- Inject secrets only via environment variables or secret stores (never in images or Git).
- Rotate FL and API tokens on each onboarding, at least quarterly.

Reverse Proxy (Traefik):

- **Purpose:** TLS termination with automatic certificate issuance/renewal (Let's Encrypt) and API hiding (only intended DO APIs are published).



TRUStworthy Multi-site Privacy Enhancing Technologies

- **Behaviour:** Internal services remain on the Docker network; only approved routers (e.g. /ping) are exposed via HTTPS.
- **Artefacts:** Traefik Compose file, dynamic routing config, and a .env holding:
 - WHOAMI_HOSTNAME (public DNS for the DO)
 - LETS_ENCRYPT_CONTACT_EMAIL
- **Healthcheck:** https://<WHOAMI_HOSTNAME>/ping/ must return 200 after first bring-up.

Deployment (Data Owner Node):

1. **Install Docker & Compose** on the DO server.
2. **Clone the DO bundle**, create .env from the provided example(s), and fill all required variables.
3. **Create the internal Docker network** (e.g., docker network create trumpet_proxy).
4. **Start core services:** make trumpet-run (or docker compose up -d for your stack).
5. **Initial access (optional):** create SSH tunnels to 3000/9000/9001 to validate the dashboard and APIs before exposure.
6. **Deploy Traefik edge:** set WHOAMI_HOSTNAME and Let's Encrypt contact; docker compose up -d; confirm https://<host>/ping/.
7. **Register the node in TRUMPET Cloud** (see section 7.4) and complete the token exchange.

7.2.2 TRUMPET Cloud (Hosted)

Packaging:

- **Runtime:** Docker (Compose in single-host setups; supported for HA).
- **Core services:**
 - Centralized Access Control(CAC)
 - AuthN/AuthZ (JWT), Role & Policy store (RBAC)
 - Federated Aggregator & Study Orchestrator (exposes port **8081** for FL comms)
 - Dataset Advertiser, Analytics Aggregator
 - Postgres (app DB), Listener/Webhooks
 - Website (Next.js)

Configuration:

- **Single .env model** per deployment. Keys commonly required:
 - **Server:** PORT, HOST, ALLOWED_HOSTS, TRUMPET_CLOUD_WEBSITE_HOST, WHOAMI_HOST
 - **Database:** DB_HOST, DB_PORT, DB_NAME, DB_USERNAME, DB_PASSWORD



TRUstworthy Multi-site Privacy Enhancing Technologies

- **JWT:** JWT_SECRET, JWT_ALGORITHM, JWT_EXPIRY
- **Federated Learning:** FL_COMMUNICATION_PORT, FL_TOKEN_EXPIRY, FL_AGG_TOKEN
- **Service URLs (local dev on Docker Desktop):**
 - TC_FL_CORE_BASE_URL
 - LISTENER_WEBHOOK_URL
 - COORDINATOR
 - DO_ENDPOINTS
- **Email:** SMTP or Azure (EMAIL_SERVICE_TYPE), sender addresses, templates
- **Misc:** RATE_LIMIT_*, PER_PAGE, LOGIC_PATH, MODELS_FOLDER, FILE_UPLOAD_ABS_DIR

Example .env (Cloud):

```
PORT=8000
HOST=0.0.0.0
ALLOWED_HOSTS=your-host-to-be-allowed
TRUMPET_CLOUD_WEBSITE_HOST=http://localhost,your-host
WHOAMI_HOST=your-domain.com
DB_USERNAME=admin
DB_PASSWORD=admin
DB_NAME=trumpet-cloud
DB_HOST=cloud-postgres
DB_PORT=5432
JWT_SECRET=your_secret_key
JWT_ALGORITHM=HS256
JWT_EXPIRY=3600
TC_FL_CORE_BASE_URL=http://tc.host
LISTENER_WEBHOOK_URL=http://listener.host
COORDINATOR=host.docker.internal
DO_ENDPOINTS=http://host.docker.internal
LOGIC_PATH=domain_layer/logics
EMAIL_HOST=sandbox.smtp.mailtrap.io
EMAIL_PORT=2525
EMAIL_USERNAME=your-mailtrap-username
EMAIL_PASSWORD=your-mailtrap-password
SENDER_EMAIL=example@domain.tld
EMAIL_SUBJECT= "Test Subject"
FILE_UPLOAD_ABS_DIR=files
RATE_LIMIT_REQUEST_PER_WINDOW=100
RATE_LIMIT_TIME_WINDOW_IN_SECOND=60
RATE_LIMIT_ALGORITHM=sliding-window
FL_COMMUNICATION_PORT=8081
PER_PAGE=10
AZURE_CONNECTION_STRING=endpoint=<azure_connection_string>
AZURE_SENDER_EMAIL=DoNotReply@example.com
```

Deployment — TRUMPET Cloud via Docker:

Use the following procedure, with OS-specific compose files if provided.

1) Environment Setup

Copy example env (Linux/Windows)

```
cp env.example .env
```

1. Edit `.env`:
 - o Add your **frontend host** to `ALLOWED_HOSTS` (comma-separated).
 - o Review and set all DB, JWT, FL, and email settings.
2. Save `.env`.

2) Start the Stack

Linux or Windows

```
docker compose -f docker-compose.yml up --build -d
```

This will:

- Build all images as defined in the selected Compose file.
- Start the API, website, databases, OpenFGA, and supporting services.
- Allow you to override service definitions or resources by editing the Compose file(s).

3) Operate the Stack

- Stop all containers

```
docker compose down
```
- Check logs

```
docker compose logs -f <service-name>
```
- Run DB migrations/seeders (if applicable)

```
docker compose exec api <migration-command>
```

TLS in Cloud: If the Cloud will be internet-facing directly, front it with a Traefik reverse proxy that terminates TLS and enforces rate-limits.

7.3 Access & Onboarding

7.3.1 Cloud Access

- **Account lifecycle:** user registration → email verification → Governance approval.
- **Roles:** Governance Admin, Researcher Org Admin/Member, Data Owner Admin/Operator.
- **FL API:** the FL communication API listens on 8081; tokens are short-lived and revocable.

7.3.2 Data Owner Onboarding

- **Provision the host** (CPU/RAM per sizing guide) and install Docker & Compose.
- **Clone DO bundle** and create `.env` from example; set DB, Redis, `CLOUD_BASE_URL`, `HAPI`, `FL`, and `NEXT_PUBLIC_*` values.
- **Start services** on the internal Docker network; validate locally (SSH tunnels).

- **Deploy Traefik**; configure DNS + Let's Encrypt; verify `https://<host>/ping/`.
- **Register DO** in the Cloud; exchange tokens; confirm the DO appears in Cloud dashboards.

7.3.3 **Researcher Onboarding**

- Request account → approval → sign in to the Cloud website.
- Create a study, select eligible DOs, and submit for execution.
- Monitor rounds; artefacts and metrics remain in the Cloud; raw data stays at DOs.

8 Pilot Preparation & Execution

This section explains how we prepared, ran, and iterated our pilots across three oncology use cases. First, we summarise each case study, then describe the pilot plan, readiness checks, and what we learned during internal dry runs. Final validation results and detailed metrics will be reported separately in D4.5.

8.1 Case Studies (HNC / NSCLC / SBRT)

HNC

Goal: Discover and cluster patterns of late toxicities after conventional external radiotherapy (e.g., intensity-modulated RT) for head & neck patients, and explore drivers across treatment plans and history.

Why FL & PETs: The feature set is clinically rich and highly sensitive. We therefore run analytics on site and exchange only privacy-processed updates (DP/SMPC/HE combinations), with privacy budgeting visible to Data Owners and researchers.

NSCLC

Goal: Build collaborative models that integrate clinical, histopathology, imaging and molecular markers to predict response in metastatic non-small cell lung cancer.

Why FL & PETs: The multi-modal nature and genomic attributes push us into higher privacy-risk territory, so we rely on armoured FL with PETs and metric-driven privacy budgeting.

SBRT

Goal: Build decision support to identify metastatic patients who would benefit from stereotactic body radiotherapy (SBRT), including a classifier.

Why FL & PETs: Eligibility modelling is clinically impactful but privacy-sensitive; we therefore train across sites without pooling raw data, and quantify residual risk with our privacy metric.

Common ground across all three:

- Same FL backbone (AFL) with PETs: SMPC+DP, SMPC+HE, and coded SMPC, selected per study needs.
- Shared privacy metrics and tracking are exposed in the Data Owner Control Panel and Researcher Dashboard.
- FHIR-first repositories and adapters at each hospital to harmonise inputs; HAPI FHIR is the backbone on premises.

8.2 Pilot Preparation & Execution

A. Two-Round Pilot Strategy

We followed a two-round, iterate-and-harden plan:

1. **Round-1 (Prototype & Early Feedback):** Stand up AFL pipelines, run end-to-end studies on smaller “development” cohorts, collect usability/privacy/performance indicators, and fix obvious bottlenecks.
2. **Round-2 (Finalised Platform & Full Cohorts):** Scale datasets, switch on stronger PET combinations where needed, finalise dashboards, and lock measurement protocols before external privacy testing.

Milestones and deliverables (e.g., D4.1 data integration, D4.2 platform reports, D4.4 validation reports) gate each round; privacy tool releases (D3.2a/b) and PET benchmarking (D2.2a/b) feed directly into pilot upgrades.

B. Internal Small-Scale Validation (Before Final Runs)

Before final validation, we executed internal dry runs using the **same partner datasets** (FHIR harmonised) but on narrowly scoped cohorts. We updated a subset of models to improve stability and the resulting matrices (accuracy/ROC for NSCLC, clustering compactness for HNC, calibration for SBRT). Full validation outcomes will be published in **D4.5**.

C. Data & Interoperability Readiness

- **FHIR alignment & ingestion:** Each site deployed a local TRUMPET repository (HAPI FHIR), with ETL/adaptor pipelines to load anonymised/pseudonymised records into **Questionnaire/QuestionnaireResponse-centric** bundles where applicable; indexes enable cohorting and fast study sampling.
- **Access boundaries:** Data never leaves the Data Owner node; only PET-processed model updates/partial stats traverse to the aggregator.

D. Privacy & Security Readiness

- **PET menu & selection:** For each study, we select from SMPC+DP, SMPC+HE, and coded SMPC, balancing accuracy, compute, and communication costs under the threat model.
- **Independent testing:** An external expert conducts privacy exercises in a controlled setting, attacking simulated hospital data through the platform; findings are graded and fed back into hardening. (See section <> for a complete reference.)

E. Platform & Operations Readiness

- **Dashboards:** Researcher and Data Owner dashboards were iterated from Round-1 feedback; we kept the workflow familiar (web-friendly) while making PETs and privacy transparent.



TRUStworthy Multi-site Privacy Enhancing Technologies

- **Training & manuals:** We prepared a pilot plan and training program covering both roles (Data Owner, Researcher), plus a comprehensive user manual for live operations.

What changed between rounds

Minor schema & labelling fixes (caught in dry-runs) improved cohort consistency.

9 Security, Pen-testing & Deployment Hardening

9.1 Pen-testing Summary

Scope & windows

We commissioned Tarlogic/Inprosec to run both an external perimeter test and an internal application test against TRUMPET Cloud and exposed DO-node services. The external test covered public IPs (e.g., 40.66.43.192, 40.66.43.2) during **27–31 Jan 2025**; the internal test covered app/API targets on **03–13 Feb 2025** across 40.66.43.192, 40.66.43.2, 51.103.92.161. A short recertification pass was executed **29 Sep–01 Oct 2025**.

Method

Tarlogic applied an OWASP-aligned methodology (discovery, active testing, exploitation where safe), supported by tools such as Nuclei, dirb, sqlmap and bespoke payloads; results were rated with CVSS and a remediation complexity metric.

Key findings

External perimeter

- **Use of software with known vulnerabilities (OpenSSH)** — version 9.2p1 on 40.66.43.192 was flagged for multiple CVEs, including CVE-2024-6387 (RegreSSHion); risk of RCE; advised to update to ≥ 9.9 . Brute-force on SSH is also feasible without lockouts/IP filtering.
- Additional hygiene issues noted (information leakage, basic auth, SlowLoris susceptibility, plaintext channels).

Internal / application

- **Excessive data exposure** — `/api/users/{id}` returned email and clear-text password; trivial enumeration via low ID space.
- **Pickle deserialization vulnerability** : allow for arbitrary code execution during the deserialization process.
- **Other issues** include Django debug info exposure, path traversal via Agreement-ID, SSRF via arbitrary webhook target, weak file-upload controls, insecure file naming enabling arbitrary pickle retrieval, missing cookie flags, and session exposure.

Recertification

- RCE endpoint no longer accepted user-provided serialised input; stale code path identified (not reachable).
- The endpoint no longer returned the user password; reviewers asked for stronger authorisation checks.

- Pickle is no longer used in the communications and has been substituted by the *msgpack* library

Remediation status

All vulnerabilities identified across the external and internal reports have been **mitigated and patched**. In particular:

- OpenSSH upgraded and hardening applied (key-only auth; rate-limit/ban; IP allow-lists).
- The users' API was refactored to least-data responses with explicit DTOs, authorisation checks, and ID randomisation. (The issue was raised internally; the password leak was already suppressed at recert.)
- SSRF, file-upload, path traversal, cookie/session flags, and debug endpoints were all closed with input validation, allow-lists, canonical path checks, and secure session settings. (Items catalogued in internal report.)

9.2 Vulnerability Management in SDLC

We integrated vulnerability management into everyday engineering, so issues caught in the pentest become unit hygiene:

Dependencies & SBOM

- Centralised monorepo dependency management with pinned versions and routine updates; SBOM generated per release for downstream consumption. (Architecture practice reflected in WP1 design.)

Secrets governance

- No secrets in code or images; pre-commit and CI scanners enforce this.
- Secrets are rotated and stored in vaulted backends; short-lived tokens for inter-service calls (Cloud & DO nodes).

Secure coding & testing

- Enforced design-before-code, code review, linting, integration/acceptance tests, and standardised git-flow—controls already defined in the platform risks/mitigation table.
- Mandatory security checklists for APIs (input validation, DTOs, authZ) and an OWASP WSTG spot-check on sensitive changes.

Post-release

- Continuous vuln feed monitoring; backporting windows defined.
- Rapid patching playbooks cover infra packages (e.g., OpenSSH), app deps, and container rebuilds.

10 Conclusions

Over the course of this work, we turned a difficult promise—privacy-preserving, multi-site AI over health data—into a working platform with clear boundaries, clear roles, and clear guarantees. On the cloud side, we converged on a layered, microservices architecture centred on a Centralised Access Gateway, and a set of Application and Core services. On the hospital side, we delivered a Data Owner Node that lives safely inside the local network, fronted by a dashboard, a hardened HAPI FHIR path for data intake, and FL runtimes that never let raw patient data leave the premises. Together, these pieces form one coherent system that researchers can use without seeing the operational scaffolding and that data owners can trust without surrendering control.

From the beginning, we designed for people as much as for components. Data Owners can manage datasets as FHIR Questionnaire/QuestionnaireResponse resources, while researchers get a straightforward way to discover advertised datasets, plan privacy methods, and train models through a familiar workflow. This division of concerns—governance and privacy on the supply side, productive analysis on the demand side—proved essential to making federated work feel routine rather than experimental.

Technically, the platform meets the requirements we set: it is modular, portable, and replaceable by design. We isolated PETs and FL algorithms behind stable interfaces so they can evolve independently, and we used asynchronous messaging between the cloud and sites to keep availability high without compromising consistency where it matters. The result is an implementation that can be tuned per use case (privacy, performance, accuracy) without architectural surgery.

We anchored delivery in an Agile SDLC, iterating from a pragmatic "monolith-first-and-then-microservices" starting point to a cleaner split as domains stabilised. The monorepo, layered boundaries (domain/application/infrastructure), and release gates around security and quality gave us predictable velocity without eroding engineering discipline. Those choices also made it easier to wrap third-party tools behind TRUMPET interfaces, so we avoid one-way doors and can swap technology as needs change.

Clinically, we prepared and executed pilots across NSCLC, HNC, and SBRT scenarios using real partner datasets. We staged them in two rounds to fix issues early, tune models, and prove end-to-end readiness. That exercise did more than validate code; it pressure-tested onboarding, governance, and study flows in conditions that look like real life. The complete validation readout moves to D4.5, but the platform state at the end of this report demonstrates that the clinical questions we targeted can be addressed without moving data and without lowering the privacy bar.



TRUStworthy Multi-site Privacy Enhancing Technologies

Security and privacy were treated as deliverables, not afterthoughts. We embedded privacy measurement next to FLCore interfaces, kept only the FL communication API exposed with token-based controls, and ran external penetration testing to verify our assumptions. Every finding from pentesting has been mitigated and patched, and the cloud and data owner node baselines now ship with hardened defaults (identity, network isolation, data-at-rest and in-motion protections). These measures align with the project's objective to provide verifiable privacy guarantees—moving us closer to a shared language with regulators for certification conversations.

Strategically, the work advances TRUMPET's original ambition: unlock high-value, siloed datasets for research in a way that respects GDPR and the people behind the data. We delivered the platform, the operating model, and the privacy instrumentation to make that possible, while keeping an eye on the bigger ecosystem. Those tracks set us up to improve PET combinations, broaden the AI model library, and push toward measurable, certifiable privacy in federated settings.

In short: the platform is real, the pilots are live, the security bar is higher, and the road ahead is clear.