# 21017 - EARS

Environment Adaptive Recommendation System

## WP5 - Integration and Validation

## D5.3 Report on Integration with 3rd Parties Services and Testing

Deliverable Type: Document
Classification: Confidential
Due Date: August 2026

## Document Information

| Document Version | Date | Remarks |
|---|---|---|
| V1 | 14.08.2026 | First Version |

**Confidentiality Notice**

This document and the information contained herein are intended for the EARS Consortium and shall not be copied or disclosed outside the Consortium without the required authorization.

## Table of Contents

1 Introduction, Scope and Integration Approach ..... 5
1.1 Purpose and Scope of the Deliverable ..... 5
1.2 Relation with Previous Work Packages ..... 6
1.3 Objectives of the Integration Activity ..... 6
2 Overview of the EARS Integration Framework ..... 8
2.1 Interface, Data Exchange and Access Contract ..... 8
2.1.1 Portuguese Use Case ..... 8
2.1.2 Turkish Use Case ..... 9
2.1.3 Spanish Use Case ..... 11
3 Integration Scenarios ..... 12
3.1 Integration with external Recommendation Engines ..... 12
3.1.1 Portuguese use case ..... 12
3.1.2 Turkish Use Case ..... 13
3.1.3 Spanish Use Case ..... 13
3.2 Integration with external AI services ..... 14
3.2.1 Portuguese use case ..... 14
3.2.2 Turkish Use Case ..... 14
3.2.3 Spanish Use Case ..... 14
3.3 Integration with enterprise systems ..... 15
3.3.1 Portuguese use case ..... 15
3.3.2 Turkish Use Case ..... 15
3.3.3 Spanish Use Case ..... 16
3.4 Integration with cloud services ..... 16
3.4.1 Portuguese use case ..... 16
3.4.2 Turkish Use Case ..... 16
3.4.3 Spanish Use Case ..... 17
3.5 Integration workflow examples ..... 17
3.5.1 Portuguese use case ..... 17
3.5.2 Turkish Use Case ..... 19
3.5.3 Spanish Use Case ..... 23

4 Integrating Testing methodology ..... 25
4.1 Interface and connectivity Testing ..... 25
4.2 Functional End-to-End Testing ..... 25
4.3 Performance Validation ..... 25
5 Integration Validation Results ..... 25
5.1 Interface and Connectivity Results ..... 25
5.2 End-to-End Scenario Results ..... 26
5.3 Partner-Model Communication Results ..... 26
5.4 Issues and Corrective Answers ..... 26
6 Conclusion ..... 26

# 1 Introduction, Scope and Integration Approach

## 1.1 Purpose and Scope of the Deliverable

The purpose of this deliverable is to document the activities carried out in Task T5.3 – Integration with 3rd Parties Services and Testing, focusing on the validation of the EARS platform as an open, interoperable, and extensible environment capable of integrating external services and recommendation engines through standardized interfaces.

Building upon the architectural foundations established in previous work packages, particularly the definition of the federated platform architecture, the implementation of the data integration framework, the API services, and the common business layer, this document demonstrates how the EARS platform can be connected with third-party applications while preserving interoperability, security, scalability, and data privacy.

The deliverable presents the integration mechanisms that enable external organizations to interact with the EARS ecosystem using open standards and well-defined APIs. It describes the supported communication interfaces, authentication mechanisms, service discovery principles, and data exchange models that facilitate seamless integration without requiring modifications to the internal implementations of external systems. In this way, EARS promotes an open ecosystem where new services can be incorporated with minimal integration effort while maintaining a consistent operational model.

Another key objective of this document is to report the testing and validation activities performed during the integration phase. These activities include functional verification of API interfaces, interoperability assessments between heterogeneous systems, end-to-end integration scenarios, performance evaluation under representative workloads, and security validation of external communications. The results of these tests provide evidence that the platform satisfies the requirements for stable and reliable operation in multi-partner environments.

The scope of this deliverable covers the complete third-party integration lifecycle, from the definition of standardized communication interfaces to the execution of integration and validation tests. The document includes the integration strategy adopted by the project, the interoperability mechanisms implemented within the EARS platform, representative integration scenarios involving external recommendation services, and the validation of these scenarios through comprehensive testing procedures.

Finally, this deliverable demonstrates that the EARS platform has evolved from a consortium-oriented research platform into an open integration environment capable of supporting future collaborations with external organizations. By adopting widely accepted standards and modular

integration principles, the platform provides a solid foundation for extending the EARS ecosystem beyond the project lifetime and facilitating the incorporation of new services, domains, and stakeholders.

## 1.2 Relation with Previous Work Packages

The integration activities consolidated the outputs of previous work packages by combining the federated recommendation model, explainability engine, and verbal explanation components into a unified and deployable EARS integration framework. Task T5.3 – Integration with 3rd Parties Services and Testing represents the final step in the technical evolution of the EARS platform, consolidating the outcomes of the previous work packages into a validated, interoperable, and production-oriented environment. While earlier activities focused on defining the architecture, developing the Artificial Intelligence models, implementing the integration framework, and deploying the platform, this task demonstrates the platform's capability to interact with external systems through standardized interfaces and validates its readiness for adoption by third-party organizations.

The foundations of the integration framework were established in **WP3**, where the overall system architecture, the federated learning topology, the information exchange model, and the AI models were specified. These activities defined the architectural principles governing secure communication, distributed intelligence, explainable AI, and interoperability among the different recommendation engines participating in the EARS ecosystem. The UML models and communication workflows also provided the basis for the implementation of standardized interfaces and interaction patterns between distributed services.

Building upon these architectural specifications, **WP4** transformed the conceptual design into an operational software platform. The work performed on the Data Integration and API framework established the mechanisms for secure data exchange, federated communication, and standardized API services. Subsequently, the Business Layer and Common Services introduced a unified service architecture capable of orchestrating heterogeneous recommendation services through a common access layer. Finally, the EARS Platform integrated these components into a centralized web-based environment that provides unified access to the recommendation services developed by the consortium while preserving the autonomy of each partner's infrastructure.

## 1.3 Objectives of the Integration Activity

The main objective of Task T5.3 – Integration with 3rd Parties Services and Testing is to validate the EARS platform as an open and interoperable environment capable of integrating external services and recommendation engines through standardized interfaces. This activity demonstrates that the platform can support collaboration beyond the project consortium while maintaining security,

scalability, and flexibility. The objective of the integration activity was to provide a standardized, partner-independent integration mechanism through a standalone FED-XAI microservice, enabling secure deployment, interoperable APIs, and consortium-wide integration.

The specific objectives of the integration activity are:

- To validate the integration of third-party services using the APIs and communication interfaces developed in previous work packages.
- To ensure interoperability through the adoption of open standards, standardized data formats, and common communication protocols.
- To verify the correct operation of authentication, authorization, and secure data exchange mechanisms during external integrations.
- To evaluate the platform's performance, scalability, and reliability under representative integration scenarios.
- To validate end-to-end communication between the EARS platform and external systems through functional and interoperability testing.
- To demonstrate that the platform provides a flexible and extensible framework that enables future partners to integrate their own services and recommendation engines with minimal implementation effort.

# 2 Overview of the EARS Integration Framework

## 2.1 Interface, Data Exchange and Access Contract

### 2.1.1 Portuguese Use Case

The architecture follows a service-oriented integration model.

The WordPress/WooCommerce environment provides the storefront, commerce data, and management interface. Its recommendation plugin requests and displays recommendations for generic and product-page placements, and exposes authorised controls for local training and display management.

The independently deployed recommendation and explainability service runs on a GPU-enabled Vultr virtual machine. It prepares catalogue and user-state data, runs the hybrid recommendation model, manages the local partner model, and prepares evidence for customer-facing explanations. The EARS platform connects to this service as the project-level integration layer, proxying recommendation requests where applicable and managing the partner model lifecycle. Azure OpenAI is used only to transform structured recommendation evidence into explanatory text; it does not select, score, or rank products.

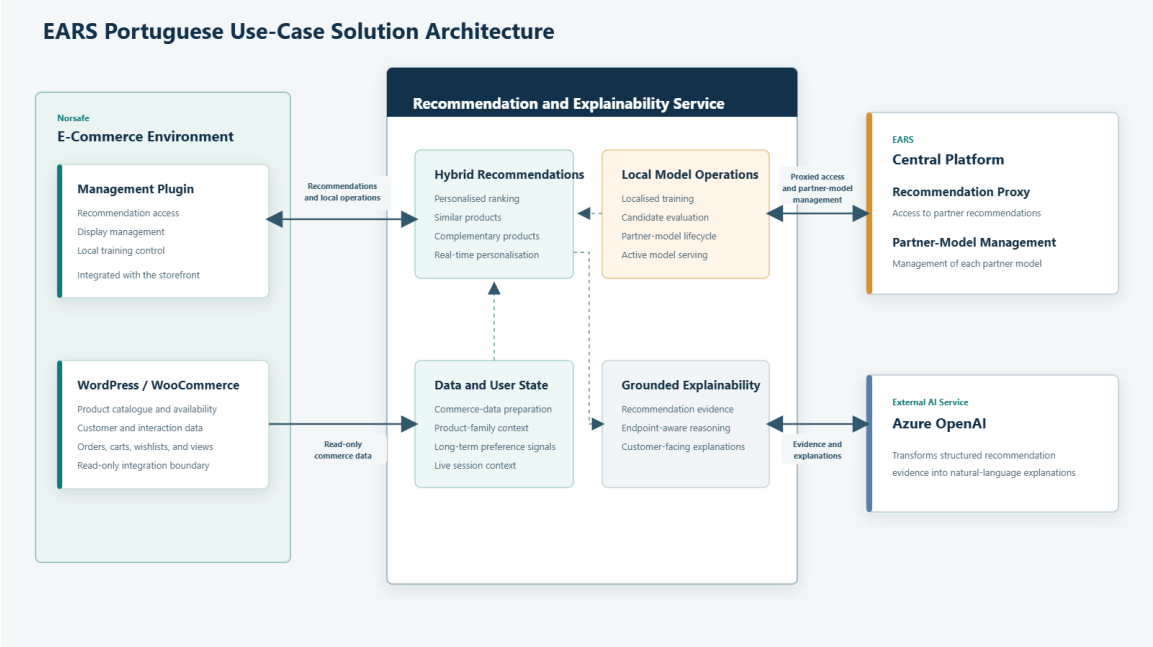



*Figure 1- Portuguese Use Case Architecture*

The interfaces use REST-style HTTP APIs over HTTPS. Customer-facing recommendation retrieval is synchronous, while training and model-management actions are initiated through authenticated requests and executed asynchronously within the recommendation service.

The WooCommerce data-access interface is read-only from the recommendation service perspective. It provides the data required to build and refresh the local recommendation state without allowing the service to create, modify, or delete commerce records.

Access is restricted by responsibility. The WordPress/WooCommerce plugin authenticates to the recommendation service using service credentials, while training control and configuration management are limited to authorised e-commerce management users. The EARS platform uses partner-level credentials, and the recommendation service uses the Azure OpenAI credentials configured for the Portuguese deployment.

Customer-facing interfaces exchange JSON payloads. Requests include only the context required for the relevant placement, such as authenticated-user context, the viewed parent product, and requested recommendation sections. Responses return parent-product identifiers, scores, availability information, recommendation metadata, and, where enabled, generated explanations.

The commerce interface provides structured catalogue, variant, pricing, order, interaction, and product-relation data. Variant-level information is consolidated into parent-product representations because the storefront routes customers to parent products.

Service locations are explicitly configured for each environment. The plugin is configured with the recommendation-service location, while the recommendation service is configured with the EARS platform, commerce data-access, and Azure OpenAI locations. This keeps test and production deployments separate without unnecessary dynamic discovery.

API versions and endpoint changes are managed through the service contract and deployment process, with partner-affecting changes validated before deployment. The project OpenAPI documentation and associated request collection remain the authoritative reference for endpoint paths, payload schemas, response fields, authentication headers, and example calls. This deliverable therefore focuses on integration responsibilities and interface boundaries rather than duplicating the endpoint specification in D5.1.

### 2.1.2 Turkish Use Case

The Turkish use case follows the same service-oriented integration model, with the EARS platform acting as the project-level integration layer in front of several independently operated Turkish partner services.

Two recommendation services participate in the Turkish use case: a media recommendation service for the PuhuTV video-on-demand platform and a forum recommendation service for the Donanım Haber community. Both are built on the Two-Tower recommendation architecture developed within EARS and are deployed and operated by their respective partners. A federated explainability service (adesso FED-XAI) is integrated as a separate service to generate explanations for the recommended items.

The EARS platform exposes a single, uniform entry point and routes each request to the appropriate Turkish partner service. It normalises the responses, performs partner health checks and keeps the partner deployments decoupled from the client applications. Each partner service remains the source of truth for its own recommendation model and data.

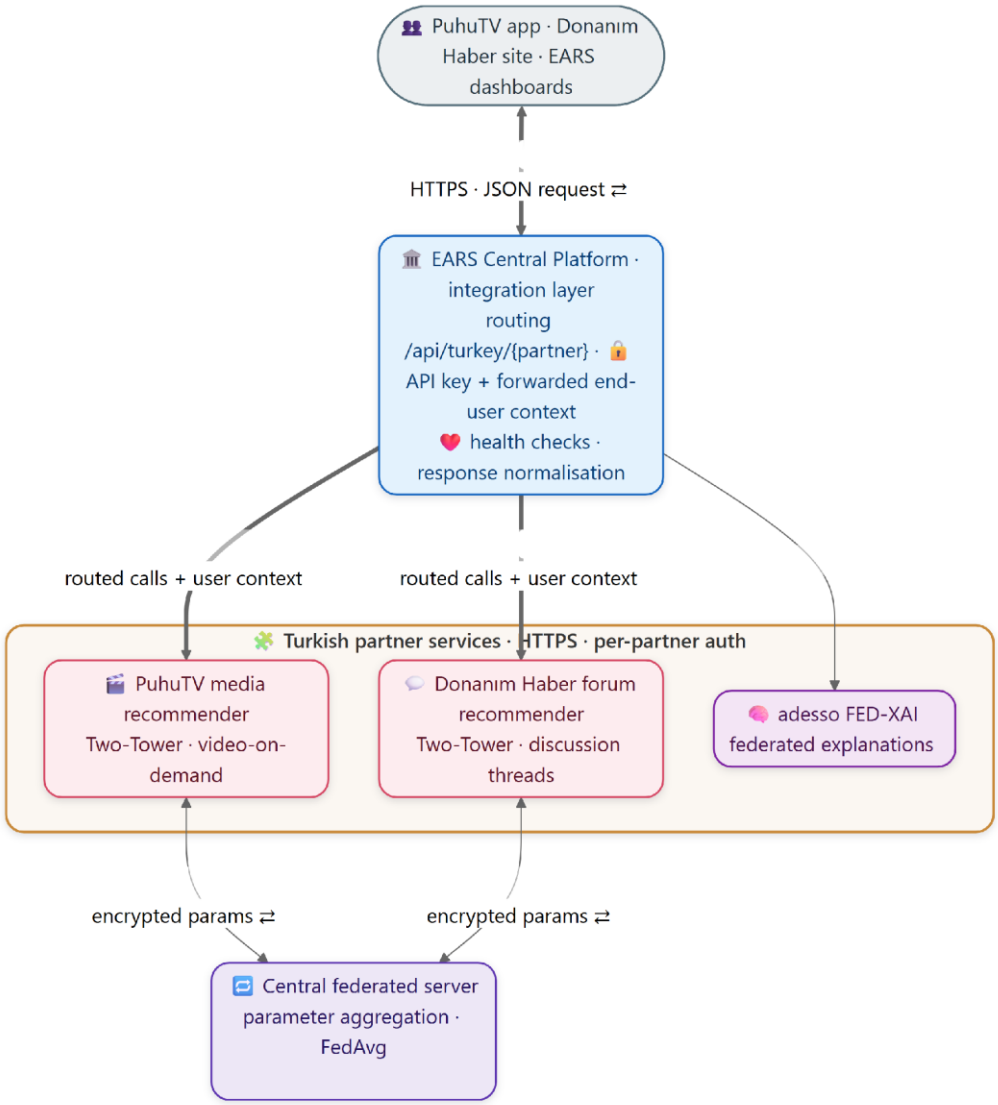


Figure – Turkish Use Case Architecture

The interfaces use REST-style HTTP APIs over HTTPS. Recommendation retrieval is synchronous, while federated training and model-management actions are initiated through authenticated requests and executed asynchronously by the central federated server and the partner services.

Access is controlled in two layers. The EARS platform authenticates to each partner service with platform-level credentials (an API key agreed per partner), while the end-user context required for personalisation is forwarded as claims on top of that platform identity. This keeps the platform-to-partner trust separate from the end-user session and allows each partner to apply its own authorisation rules.

Data exchange follows privacy-by-design principles. Raw user interaction data — PuhuTV viewing activity and Donanım Haber forum activity — never leaves the partner environment. Only model parameters and the information required for a specific recommendation or explanation request are exchanged. Customer-facing interfaces exchange JSON payloads: requests carry only the context needed for the placement, such as the authenticated-user context and the relevant item, and responses return ranked item identifiers, scores, recommendation metadata and, where enabled, the generated explanation.

Service locations are configured explicitly for each environment, so test and production deployments remain separate without dynamic discovery. API versions and endpoint changes are managed through the service contract and deployment process, with partner-affecting changes validated before deployment.

The project OpenAPI documentation and the associated request collection remain the authoritative reference for endpoint paths, payload schemas, response fields and authentication headers. This section therefore focuses on integration responsibilities and interface boundaries rather than duplicating the endpoint specification provided in D5.1.

### 2.1.3 Spanish Use Case

The Spanish use case follows the same service-oriented integration model, with the EARS platform acting as the project-level integration layer in front of three independently operated partner services: Glintt (healthcare), Square1 (media and content) and Dextromédica (hospital logistics). Each partner deploys its own Local Recommendation Service (LRS), which remains the source of truth for its domain-specific model and data, while the EARS platform exposes a single, uniform entry point, routes each request to the appropriate partner LRS, normalises responses and performs partner health checks.

The interfaces use REST-style HTTPS/JSON APIs, grouped into inference APIs for recommendation requests and federated learning APIs used by the EARS Federated Learning Core to coordinate training rounds, collect model updates and distribute new global models. Recommendation retrieval is synchronous, while federated training and model-management actions are initiated through authenticated requests and executed asynchronously between the partner LRS instances and the central federated server.

Access is controlled in two layers, following the pattern used in the Turkish use case: the EARS platform authenticates to each Spanish partner service with platform-level credentials (mutual TLS combined with OAuth2/OIDC tokens), while the end-user context required for personalisation is forwarded on top of that identity. Role-based access control (RBAC) ensures that only authorised services can submit updates, pull models or call inference endpoints, and audit trails are kept for all security-relevant events.

Data exchange follows privacy-by-design principles: raw clinical, media and logistics data never leaves the respective partner's environment, and only curated feature extracts, model parameters and aggregate metrics are exchanged with the EARS platform. Customer- and professional-facing interfaces exchange JSON payloads carrying only the context needed for the placement (e.g. clinical context, content topic or logistics scenario), and responses return ranked recommended items together with explanation artefacts. As with the Portuguese and Turkish use cases, the project OpenAPI documentation and associated request collection remain the authoritative reference for endpoint paths, payload schemas, response fields and authentication headers; this section focuses on integration responsibilities and interface boundaries rather than duplicating that specification.

# 3 Integration Scenarios

## 3.1 Integration with external Recommendation Engines

### 3.1.1 Portuguese use case

The Portuguese use case integrates its recommendation service as an external recommendation engine within the EARS ecosystem. The service is independently deployed and exposes recommendation capabilities through documented API interfaces, allowing EARS to request personalised and context-aware recommendations without coupling to the internal training, data-processing, or explainability implementation.

The integration supports general recommendation requests and product-page recommendation scenarios. The latter can return similar products and complementary products as independently selectable sections. The detailed interface and integration contract are defined in the corresponding EARS integration deliverable; this section records its application in the Portuguese use case.

### 3.1.2 Turkish Use Case

The Turkish use case integrates external recommendation engines from more than one domain. The PuhuTV media recommendation service is integrated as an external recommendation engine for video-on-demand content, and the Donanım Haber forum recommendation service is integrated for community discussion content. Both are independently deployed and expose their recommendation capabilities through documented REST API interfaces, so EARS can request personalised, context-aware recommendations without coupling to their internal training, feature-processing or data pipelines.

The PuhuTV media recommendation engine is based on the Two-Tower architecture developed within EARS. It represents users and video-on-demand items in a shared embedding space and returns ranked content recommendations for a given user and context. The integration supports general recommendation requests and item-related recommendation scenarios, and returns ranked results in the standard EARS response format so they can be consumed by other EARS components.

This standardised integration allows each Turkish recommendation engine to evolve independently while remaining interoperable with the rest of the EARS ecosystem. The detailed interface and integration contract are defined in the corresponding EARS integration deliverable (D5.1); the deployment of the Donanım Haber forum recommendation engine is described in its own subsections of this document, and this section records the integration of the external recommendation engines in the Turkish use case.

### 3.1.3 Spanish Use Case

The Spanish use case does not rely on a single external recommendation engine but on three independently deployed Local Recommendation Services (LRS), one per participating domain and partner: Glintt (healthcare), Square1 (media and content) and Dextromédica (hospital logistics). Each LRS trains and serves its own domain-specific model and exposes its recommendation capabilities through documented HTTPS/JSON inference APIs, so the EARS platform can request personalised, context-aware recommendations without coupling to the underlying clinical, editorial or logistics implementation.

This follows the same integration pattern used for the Portuguese and Turkish external recommendation engines, applied independently across three heterogeneous domains rather than a single one. Each Spanish LRS enriches incoming requests with local data, executes its domain-specific model, and returns a ranked list of recommended items together with explanation artefacts, allowing hospital applications, content-creation tools and logistics dashboards to consume EARS recommendations with minimal coupling. The detailed interface and integration contract are defined

in the corresponding EARS integration deliverable (D4.2); this section records the application of the external-recommendation-engine pattern across the three domains of the Spanish use case.

## 3.2 Integration with external AI services

### 3.2.1 Portuguese use case

Azure OpenAI is integrated into the Portuguese recommendation service to generate human-readable explanations for recommendation results. The recommendation model performs ranking locally using the available catalogue, interaction, and user-preference signals. Once a result has been selected, the service compiles structured evidence from the signals that contributed to that recommendation and passes this explanation brief to Azure OpenAI.

The external AI service is therefore used for explanation generation rather than for ranking or decision-making. This preserves a clear separation between the recommendation process and the generation of user-facing language, while ensuring that explanations remain grounded in the evidence used by the recommender.

### 3.2.2 Turkish Use Case

The federated explainability service (adesso FED-XAI) is integrated into the Turkish use case as an external AI service used to generate explanations for recommendation results. The recommendation engines rank items locally using their own user, content and interaction signals. Once a result has been produced, the relevant evidence is passed to the FED-XAI service, which returns a federated explanation for the recommended items.

As in the other use cases, the external AI service is used for explanation generation rather than for ranking or decision-making. This preserves a clear separation between the recommendation process and the generation of user-facing explanations, while keeping the explanations grounded in the evidence used by the recommenders. The service is integrated through a standardised REST API with API-key authentication, so it can be deployed and called independently by the partner services participating in the Turkish use case.

### 3.2.3 Spanish Use Case

The Spanish use case integrates the federated explainability layer (FedXAI) as an external AI service used to generate human-readable justifications for recommendation results across the three domains. Each LRS (Glintt, Square1, Dextromédica) performs ranking locally using its own clinical, content or logistics signals; once a result has been produced, the supporting evidence is passed to the FedXAI layer, which combines TSK-FRBS fuzzy-rule inference with an LLM-based verbaliser and LIME/SHAP feature-attribution techniques to produce a domain-tailored explanation.

As in the other use cases, the external AI service is used strictly for explanation generation rather than for ranking or decision-making, preserving a clear separation between the recommendation process and the generation of user-facing explanations. This keeps healthcare, media and logistics explanations grounded in the evidence used by each domain's recommender, while allowing the FedXAI service to evolve and be deployed independently of the partner recommendation engines.

## 3.3 Integration with enterprise systems

### 3.3.1 Portuguese use case

The Portuguese use case is integrated with the Norsafe WordPress/WooCommerce environment. The website sends authenticated requests to the recommendation service when recommendations are required, including the user context, current product-page context, and relevant online activity where available. The service returns ranked products, availability information, confidence-related values, and explanatory text for display on the website.

A read-only WooCommerce DB Access API supplies the recommendation service with the data required for training and serving. This includes customer, catalogue, product-variation, interaction, order, wishlist, cart, and product-relation data. The service consolidates variant-level information into the corresponding parent products, ensuring that recommendations correspond to the products presented by the website while still benefiting from the detailed characteristics expressed through user purchases and cart activity.

### 3.3.2 Turkish Use Case

The Donanım Haber use case integrates its recommendation service as an external recommendation engine within the EARS ecosystem. The service is deployed independently and exposes its recommendation capabilities through documented REST API interfaces, allowing EARS components to request personalised forum content recommendations without coupling to the underlying recommendation models, feature engineering, or data processing pipeline.

The recommendation service generates personalised recommendations for forum users by leveraging user interaction history, profile information, and contextual features. The engine is based on the Two-Tower recommendation architecture developed within the EARS project, enabling efficient retrieval of relevant discussion threads through a shared embedding space for users and forum content.

The integration supports personalised recommendation requests for forum discussions and related content. Recommendation requests are submitted through the standard EARS interface, and the service returns ranked recommendation results in a structured format that can be consumed by external EARS components. This standardised integration enables interoperability with other services

participating in the EARS ecosystem while allowing the recommendation engine to evolve independently.

The detailed API specification and integration contract are described in the corresponding EARS integration deliverables. This section documents the deployment and application of the external recommendation engine within the Donanım Haber pilot.

### 3.3.3 Spanish Use Case

The Spanish use case is integrated with three heterogeneous sets of partner enterprise systems, one per domain. Glintt connects to hospital information systems, Electronic Health Record (EHR) repositories and medical-imaging systems to support oncology diagnosis and treatment-planning recommendations, keeping clinical identifiers separated from analytical datasets in line with hospital ethics and DPIA requirements. Square1 connects to content-management and social-media monitoring systems, ingesting public sentiment streams, user feedback and content metadata to inform editorial and content-creation strategy.

Dextromédica connects to hospital ERP and inventory-management systems together with IoT/RFID asset-tracking sensors, supplying stock levels, consumption rates, reordering data and real-time material-location logs used for hospital logistics recommendations. In every case, the partner enterprise systems are accessed by the local LRS through curated, pre-processed feature extracts; raw operational and clinical records remain within each institution's own infrastructure and are never exposed directly to the central EARS platform.

## 3.4 Integration with cloud services

### 3.4.1 Portuguese use case

The Portuguese recommendation service is deployed in a Vultr virtual machine with GPU access. The service runs in a containerised environment, including the recommendation model, preprocessing and serving components, source synchronisation logic, and API runtime. GPU availability supports the model workload and the operational requirements of the recommendation service.

The deployment configuration keeps runtime credentials and environment-specific settings outside the container image. This allows the same service image to be configured for the target environment without embedding access tokens, source configuration, or external AI-service credentials in the application artefact.

### 3.4.2 Turkish Use Case

The Donanım Haber recommendation service is deployed within the company's production infrastructure using a containerised architecture. The deployment includes the recommendation

model, feature extraction and preprocessing components, embedding generation, vector indexing, and REST API services required to provide personalised recommendations to the EARS ecosystem.

The service is designed as an independently deployable component and is orchestrated within the existing infrastructure, enabling scalability, monitoring, and straightforward integration with other EARS services. The recommendation engine accesses user and content data through the established data pipeline while exposing recommendation functionality through standard API interfaces.

Deployment-specific configuration, including database connections, API credentials, and environment variables, is maintained separately from the application container. This approach enables the same application image to be deployed across development, testing, and production environments without embedding environment-specific configuration or sensitive credentials within the container image.

### 3.4.3 Spanish Use Case

The Spanish use case follows a hybrid edge-cloud deployment model. Each partner hosts its LRS in an isolated compute environment suited to its domain: Glintt runs an on-premise cluster with GPU support for medical-imaging workloads; Square1 runs containerised NLP services on a private cloud; and Dextromédica runs an edge infrastructure with embedded IoT integrations for real-time asset tracking. This keeps sensitive clinical, content and logistics data local while still allowing each service to be independently scaled and maintained.

Model coordination takes place through a central federated aggregator hosted on secure cloud infrastructure, which receives encrypted model updates from the three partner nodes and distributes new global model versions back to them. All core services, including the partner LRS instances and the Federated Model Server, are containerised and can be orchestrated with Kubernetes-type technologies for horizontal scalability and resilience, while runtime credentials and environment-specific configuration are kept outside the container images, mirroring the approach used in the Portuguese and Turkish deployments.

## 3.5 Integration workflow examples

### 3.5.1 Portuguese use case

The first workflow concerns source synchronisation and model maintenance. WooCommerce data is exposed through the read-only DB Access API and retrieved by the recommendation service as structured source frames. These frames are used to build the active catalogue, user-preference signals, product-family relationships, and training inputs required by the model.

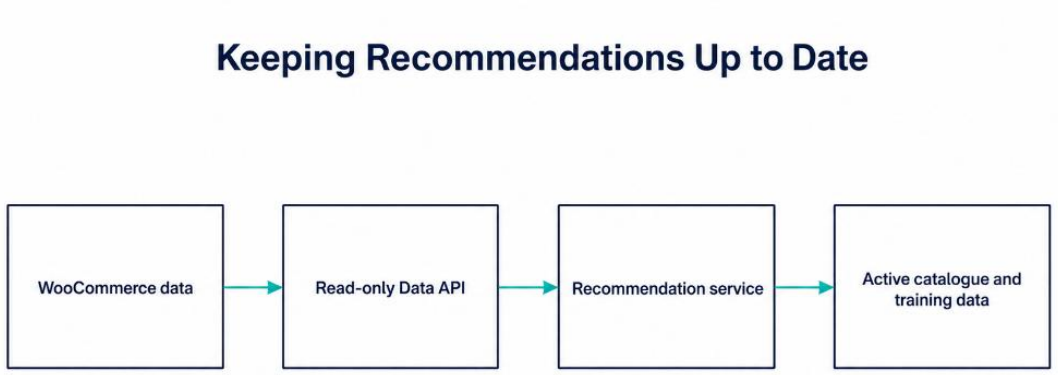



*Figure 2 - Portuguese UC - Workflow 1 - Catalog Synchronism*

The second workflow concerns website recommendations. When a user visits the Norsafe website, the website sends an authenticated request to the recommendation API. For generic recommendations, the request may contain the user identifier and current online intent, such as recently viewed products. The service ranks suitable parent products, applies the relevant display policy, and returns the selected results to the website.

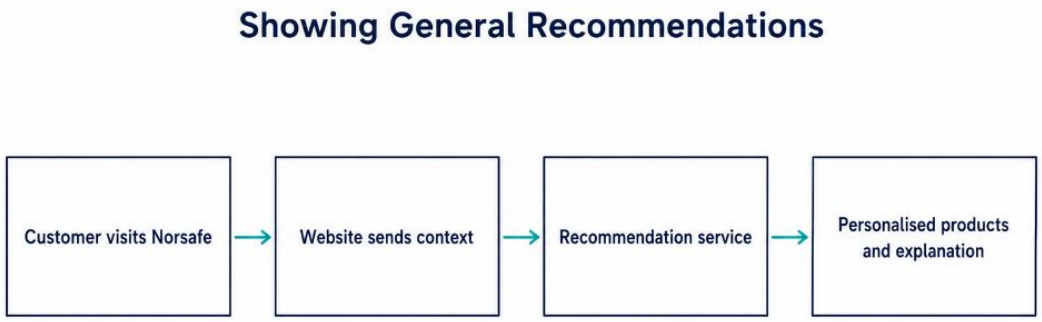



*Figure 3 - Portuguese UC - Workflow 2 - Generic Recommendation Example*

The third workflow concerns product-page recommendations and explanations. When a customer views a product, the website can request similar products, complementary products, or both. The service evaluates the selected product in relation to the catalogue and the available user context, returns the requested recommendation sections, and generates explanations from the structured evidence associated with each result.

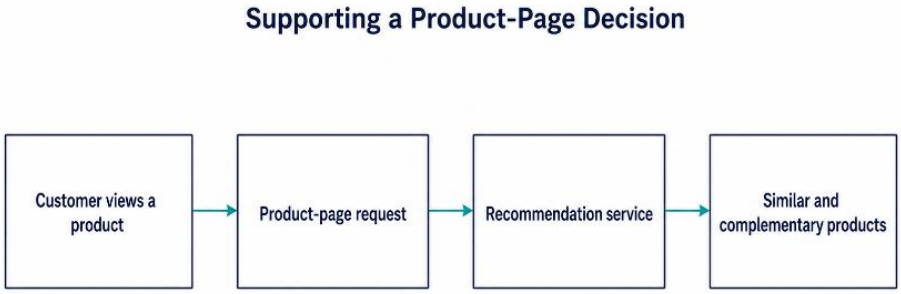
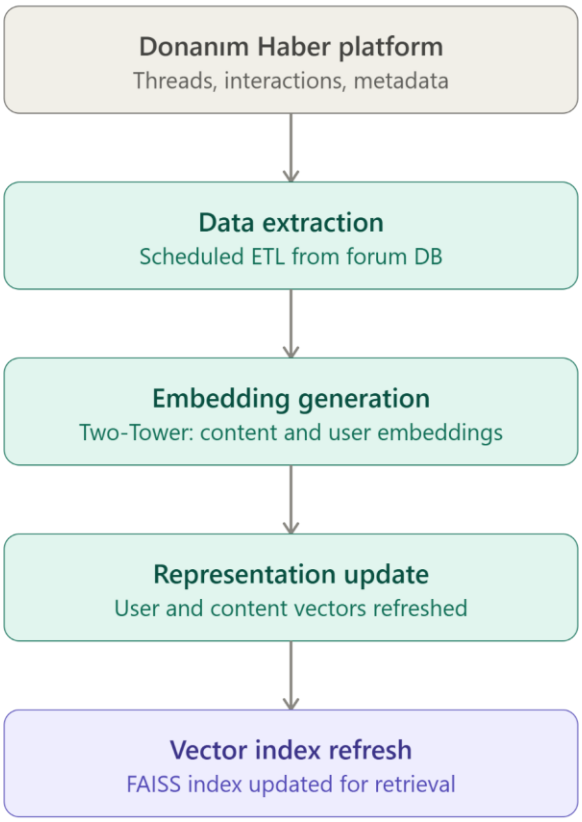



*Figure 4 - Portuguese UC - Workflow 3 - Product-Page Recommendation Example*

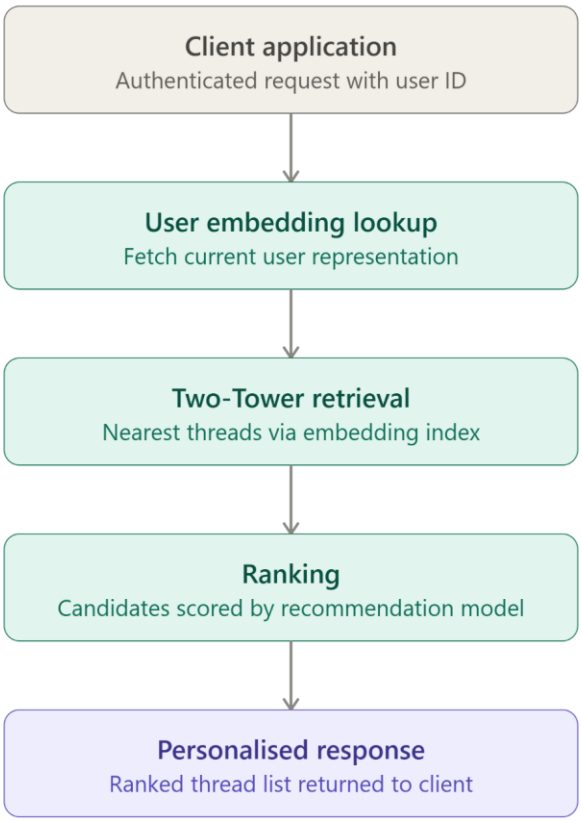
### 3.5.2 Turkish Use Case

The first workflow concerns forum data synchronisation and recommendation model maintenance. User interactions, discussion threads, and content metadata are periodically extracted from the Donanım Haber platform and processed through the recommendation pipeline. Content embeddings and user representations are updated, and the vector index used for retrieval is refreshed to ensure that recommendations reflect newly created discussions and evolving user interests.



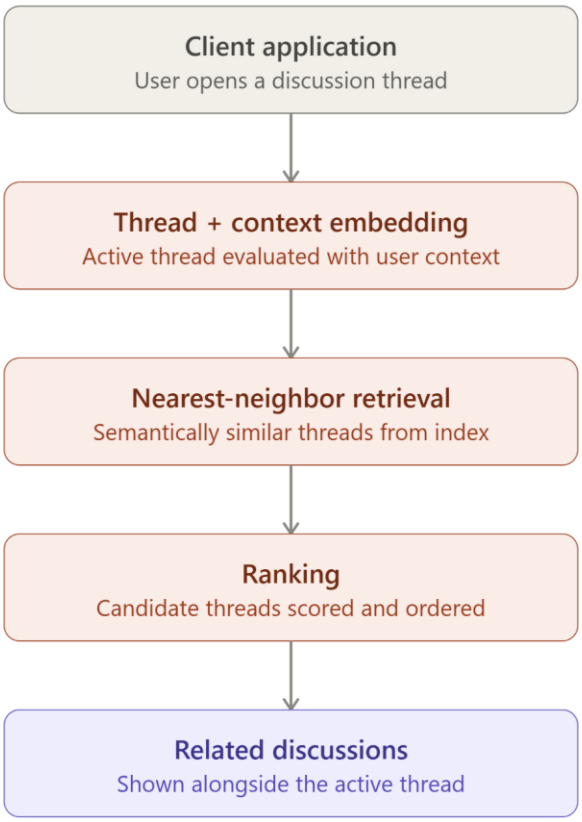

*Workflow 1 – Forum Data Synchronisation*

The second workflow concerns personalised forum recommendations. When a registered user accesses the Donanım Haber platform, the client application submits an authenticated recommendation request containing the user identifier and relevant contextual information. The recommendation service retrieves the corresponding user representation, identifies the most relevant discussion threads using the embedding index, ranks the results according to the recommendation model, and returns a personalised list of forum discussions to the client application.




*Workflow 2 – Personalised Forum Recommendation*

The third workflow concerns related content recommendations. When a user opens a discussion thread, the client application requests recommendations for similar or related discussions. The recommendation service evaluates the currently viewed thread together with the available user context, retrieves semantically similar discussions from the embedding index, ranks the candidate threads, and returns the most relevant recommendations for display alongside the active discussion.




*Workflow 3 – Related Discussion Recommendation*

A fourth workflow concerns federated learning and model aggregation across the Turkish use case. Each Turkish partner — the PuhuTV media recommendation service and the Donanım Haber forum recommendation service — trains its Two-Tower model locally on its own interaction data. On each federated round, the partners send only their encrypted local model parameters to the central EARS federated server, which aggregates these updates (for example, using Federated Averaging) into a global model and redistributes the updated global model back to each partner. Raw user data never leaves the partner environment; only model parameters are exchanged.

This federated aggregation enables cross-domain knowledge transfer between the media and forum platforms: patterns learned on one platform strengthen the recommendations served on the other, which is particularly valuable for cold-start users and sparse-interaction items. Each partner then uses the updated global model for local inference, so end users receive improved recommendations without any exchange of raw interaction data between the platforms.

Aggregation on the central platform is configurable rather than fixed. Partner models can be combined using different strategies — Federated Averaging (FedAvg), FedAvgM, FedAdam, Trimmed Mean or SCAFFOLD — selected per partner through an aggregation policy, and each partner's contribution is weighted by its number of training samples. Every round is validated and evaluated by the platform: uploaded parameters are shape-checked before they are combined, the resulting global model is analysed and scored on a held-out evaluation set, and convergence is tracked against a target accuracy so that rounds can stop early. This lets the federated integration be tested and tuned per partner while raw interaction data stays local.

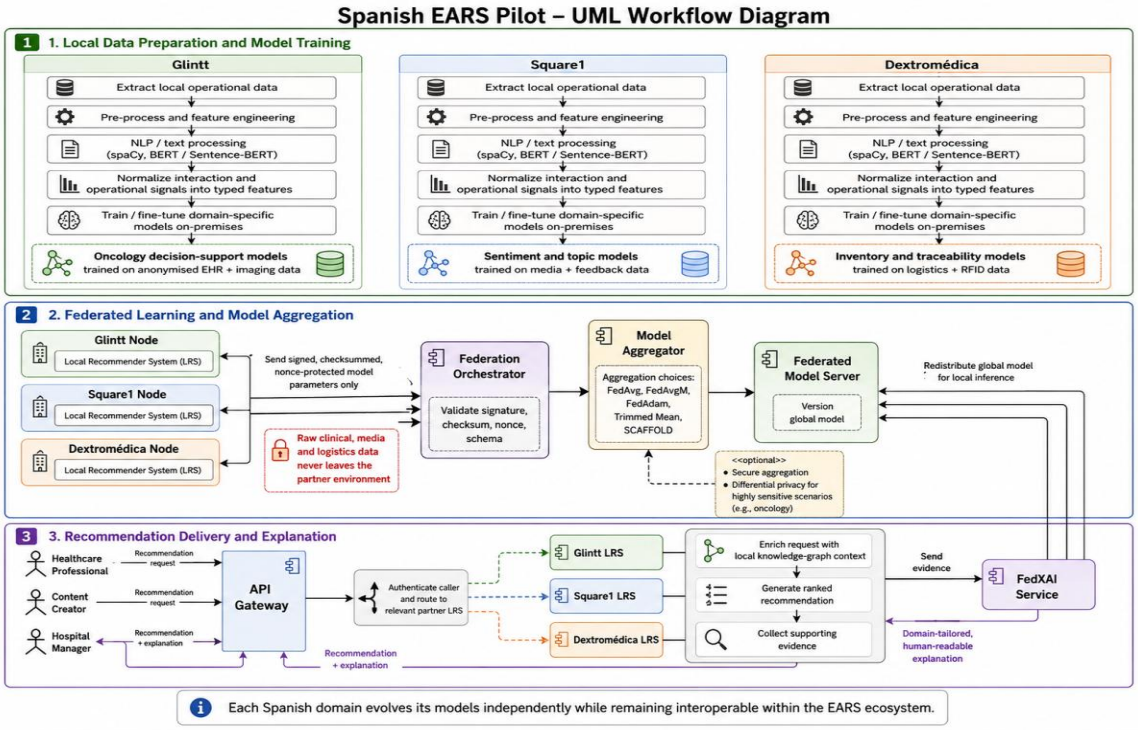
Workflow 4 – Federated Learning and Model Aggregation

### 3.5.3 Spanish Use Case

The first workflow concerns local data preparation and model training. Each Spanish partner extracts and pre-processes its own operational data locally, using natural-language processing for clinical notes, social-media content and text fields (spaCy, BERT/Sentence-BERT), and normalises interaction and operational signals into typed features. Domain-specific models are then trained or fine-tuned on-premises: Glintt trains oncology decision-support models on anonymised EHR and imaging data, Square1 trains sentiment and topic models on media and feedback data, and Dextromédica trains inventory and traceability models on logistics and RFID data.

The second workflow concerns federated learning and model aggregation across the three Spanish domains. On each federated round, the Glintt, Square1 and Dextromédica nodes send only their signed, checksummed and nonce-protected model parameters to the central EARS federated server; raw clinical, media or logistics data never leaves the partner environment. The Federation Orchestrator validates each update (signature, checksum, nonce and schema checks) before the Model Aggregator combines them, using algorithms selected per round such as FedAvg, FedAvgM, FedAdam, Trimmed Mean or SCAFFOLD, with optional secure aggregation and differential privacy for highly sensitive scenarios such as oncology data. The resulting global model is versioned in the Federated Model Server and redistributed to each LRS for local inference.

The third workflow concerns recommendation delivery and explanation. A request from a healthcare professional, content creator or hospital manager enters the EARS platform through the API Gateway, which authenticates the caller and routes the request to the relevant partner LRS. The LRS enriches the request with local knowledge-graph context, produces a ranked recommendation using its (possibly newly aggregated) model, and passes the supporting evidence to the FedXAI service, which returns a domain-tailored, human-readable explanation. This standardised workflow allows each Spanish domain to evolve its own models independently while remaining interoperable with the rest of the EARS ecosystem.




The diagram summarizes the operation of the Spanish EARS pilot in three main stages, from local data preparation to the final delivery of an explainable recommendation.

In the first stage, Local Data Preparation and Model Training, each Spanish partner works independently within its own infrastructure. GLINTT processes information related to the oncology field, Square1 analyzes media and public opinion data, and Dextromédica works with hospital logistics and traceability data. The data is preprocessed, transformed into features suitable for the models, and used to train or adjust domain-specific models. In this way, sensitive data always remains within the organization that generates it.

The second stage represents federated learning. The three local nodes send only model-related information, not the original data, to the central EARS server. The Federation Orchestrator verifies the integrity and validity of the received updates before the Model Aggregator combines them using the selected federation algorithm. The resulting global model is versioned on the Federated Model Server and subsequently redistributed to the local systems. This mechanism allows leveraging knowledge from different participants while maintaining data sovereignty and privacy.

Finally, the third stage illustrates the inference, recommendation, and explainability process. When a user—for example, a healthcare professional, a content creator, or a hospital manager—requests a recommendation, the request enters through the API Gateway, which authenticates the user and directs it to the corresponding local recommendation system. The local model incorporates the

available context, generates and prioritizes recommendations, and obtains the evidence that supports the outcome. This evidence is sent to the FedXAI service, which is responsible for generating a clear and understandable explanation tailored to the domain and user type.

Overall, the diagram aims to illustrate the fundamental idea behind the Spanish EARS architecture: each domain maintains autonomy over its data and local models but can participate in a common federated learning process and use standardized interfaces to generate interoperable and explainable recommendations.

# 4 Integrating Testing methodology

## 4.1 Interface and connectivity Testing

Interface validation focused on verifying REST endpoint availability, API-key authentication, endpoint documentation completeness, standardized HTTP responses, and successful OpenAPI specification generation

## 4.2 Functional End-to-End Testing

End-to-end integration testing verified the complete workflow from model access to global explanation generation while confirming consistent interaction between the integrated EARS components without changing their functional behaviour.

## 4.3 Performance Validation

Performance validation covers the responsiveness and scalability of the integrated platform. Three dimensions are addressed: the latency of recommendation serving through the platform, the processing time of each federated aggregation round on the central server, and scalability as the number of participating partner models per round increases. For the federated path, each round is exercised end to end — collecting the uploaded partner models, validating their parameter shapes, aggregating them into a global model, and scoring that model on a held-out evaluation set — so that both the aggregation cost and the resulting model accuracy can be observed as rounds progress and as more partners join.

# 5 Integration Validation Results

## 5.1 Interface and Connectivity Results

The integration activities resulted in a fully documented REST API with improved endpoint descriptions, complete HTTP response documentation, enhanced service metadata, and consortium-compliant OpenAPI documentation.

### 5.2 End-to-End Scenario Results

The end-to-end integration scenarios exercised the central platform as the single integration layer in front of the partner services. The platform routed recommendation requests to the correct partner service, applied the two-layer authentication and partner health checks, and returned normalised responses to the client. The federated path ran as a complete cycle: partner services uploaded their local model parameters, the central server verified and aggregated them into a global model, analysed and scored the result, and made the updated global model available for redistribution. Across these scenarios the platform operated consistently over routing, authentication, federated aggregation and result delivery, with every partner reached through the same standardized interface.

### 5.3 Partner-Model Communication Results

Partner interoperability was strengthened through standardized interface definitions, deployment documentation, Partner Integration Forms, and OpenAPI specifications that enable consistent communication between partner models and the FED-XAI service.

### 5.4 Issues and Corrective Answers

Several integration issues are handled on the platform side, without requiring changes to partner implementations. Because partner models are trained independently, uploaded updates can differ from the expected parameter shapes; the platform detects this through a shape-verification step, flags the affected models and excludes them from aggregation, so that a single incompatible update cannot corrupt the global model. An optional strict mode lets a round fail fast when invalid updates are present, and invalid models are recorded with the reason so they can be corrected at source. Because partner services can become ready at different times, incremental round-based aggregation folds new partners into the existing global model as they join, rather than requiring all partners to be present at once. And if the central federated server is temporarily unreachable, partner services continue to serve recommendations locally in a non-federated mode, preserving availability during aggregation cycles.

## 6 Conclusion

The activities presented in this deliverable demonstrate the successful integration and validation of the EARS platform with heterogeneous third-party services and recommendation engines. Through standardized REST APIs, common data-exchange mechanisms, documented authentication procedures, and OpenAPI specifications, the platform provides a consistent integration layer while allowing each participating service to maintain its own models, data, and operational infrastructure. The Portuguese, Turkish, and Spanish use cases further demonstrate that this integration approach

can be applied across different domains, including e-commerce, media, community platforms, healthcare, and hospital logistics.

The integration scenarios also confirm the modularity of the EARS architecture. External recommendation engines remain independently deployable, while external AI services are incorporated primarily for explainability without directly influencing the underlying ranking and decision-making processes. Enterprise and cloud integrations further demonstrate that partner-specific infrastructures can be connected to EARS through clearly defined service boundaries while preserving the autonomy of the individual systems.

A key outcome of the integration activities is the validation of the federated and privacy-preserving communication model. Raw user, clinical, media, logistics, and interaction data remain within the corresponding partner environments, while only the information required for recommendation requests or model coordination is exchanged. Federated workflows enable locally trained models to participate in central aggregation and subsequently receive updated global models without requiring the transfer of raw partner data.

The testing activities provide further evidence of the technical readiness of the integration framework. Interface and connectivity testing verified API availability, authentication mechanisms, standardized responses, and API documentation, while end-to-end scenarios validated routing, partner communication, federated aggregation, and recommendation delivery. The platform also incorporates mechanisms for handling incompatible model updates, incremental partner participation, and temporary unavailability of the central federated server, contributing to the robustness and continuity of the overall solution.

Overall, D5.3 demonstrates that EARS has evolved into an interoperable, modular, secure, and extensible integration environment capable of connecting heterogeneous recommendation systems and external AI services through standardized mechanisms. The validated integration approach provides a solid technical foundation for extending the EARS ecosystem with additional partners, services, and application domains beyond the current project pilots and supports the future exploitation and sustainability of the platform.